Remerciement

Je tiens à remercier les professeurs de l'EPHEC pour leur engagement et le temps qu'ils ont consacrés aux partages de leurs connaissances, tout au long de ce bachelier. Les commentaires et conseils qu'ils m'ont fournis ont été d'une grande aide dans ma compréhension des matières étudies.

Je tiens également à remercier mes camarades de classe, mes amis et ma famille qui ont souvent été présents pour m'encourager et me soutenir pendant cette période de travail intense.

J'aimerai aussi remercier la Défense pour avoir contribué à la réalisation de ces études et pour m'avoir permis d'effectuer mes stages en internes.

Enfin, je tiens à remercier toutes les personnes qui ont participé à ce projet de quelque manière que ce soit.

Merci encore pour votre temps, votre soutien et votre encouragement tout au long de ce travail de fin d'étude.

Table des matières

Liste des abréviations

- **CQRS** (*Command and Query Responsibility Segregation*) : Ségrégation des responsabilités de commande et de requête
- **DB** (Data base) : Base de données
- **API** (Application Programming Interface) : Interface de programmation d'application
- **UI** (User Interface) : Interface utilisateur
- **IDE** (*integrated development environment*) : Environnent de développement intégré
- **MAUI** (Multi-platform App UI) : Framework pour application mobile et desktop, multiplateforme
- **SGBD** (système de gestion de base de données) : Permet la gestion, la manipulation et le stockage de données
- **XAML** (eXtensible Application Markup Language) : Langage extensible de description d'information
- **CSS** (Cascading Style Sheets) : Les feuilles de style en cascade permettent d'organiser les éléments de type HTML.
- **HTML** (HyperText Markup Language) : Les balises hypertexte permettent de définir un contenu à l'aide de ballise.
- **JSON** (JavaScript Object Notation) : C'est un format de donnée sous forme de texte.
- **SQL** (Structured Query Language) : C'est un langage de requêtes structuré qui permet de communiquer (écriture ou lecture) avec une base de données.
- **HTTP** (Hypertext Transfer Portocol) : Protocole de transfert hypertexte

1 Introduction

1.1 Introduction

1.1.1 Contexte

Ce travail de fin d'étude est réalisé dans le cadre du cursus informatique de gestion réalisé à l'EPHEC. Il a pour objet la démonstration de mes acquis en analyse et développement.

Le choix de thème découle de la volonté de créer une application proche du domaine dans lequel je travail. En effet je suis actuellement chef d'atelier et manager adjoint dans un garage à la Défense. Ce garage inclus également un bureau de transport qui s'occupe de la gestion des prêts de véhicule.

Crée une application de covoiturage me permet ainsi de faire le rapprochement entre mon environnement professionnel actuel et mes études.

1.2 Cahier des charges

1.2.1 Introduction

Le cahier des charges comprend une description générale des fonctionnalités et du type d'utilisateur. Les solutions proposées et les technologies principalement employés dans le cadre du projet, y sont également abordées. Enfin, la méthodologie de travail y est décrite en fin de chapitre.

1.2.2 Description

Dans la partie web et mobile :

L'application permettra aux utilisateurs de publier une annonce ou/et d'effectuer une demande de réservation pour une annonce.

L'utilisateur pourra rechercher un trajet en sélectionnant la date et/ou les localités, parcourir les annonces correspondantes, en afficher les détails et, effectuer, consulter ou/et accepter une réservation.

Dans la partie web :

Il pourra accéder à son compte afin de gérer son profil, gérer ces messages, gérer ces réservations et ces payements.

L'utilisateur pourra également gérer ces véhicules et ses annonces. Il pourra inscrire/supprimer plusieurs véhicules et définir un véhicule courant. Il pourra accepter des demandes de réservation et confirmer un payement pour les annonces qu'il aura publiées.

Un utilisateur admin pourra également gérer les utilisateurs et les bloquer en cas de signalement justifié. Il pourra aussi gérer les demandes d'inscription de véhicule des utilisateurs.

Tout utilisateur hormis un administrateur pourra être signalé et coté.

L'information relative à l'entreprise pourra être consultée dans les rubriques « Contact » et « FAQ ».

1.2.3 Application

CoivoitEco est disponible en deux versions, une version web comprenant l'intégralité des fonctionnalités de l'application et une version mobile ne permettant que l'accès simplifier à une

partie des fonctionnalités. Des cas d'utilisations supplémentaires sont disponibles pour les utilisateurs de type administrateur.

1.2.4 Intervenants

1.2.4.1 Utilisateur non-enregistré (guest)

Les utilisateurs non-enregistré sont des utilisateurs n'étant pas encore inscrit sur l'application ou n'étant pas encore connecté. Leur accès est limité aux fonctionnalités publiques de CovoitEco, c'est à dire ne nécessitant aucune connexion.

1.2.4.2 Utilisateur enregistré non-bloqué (defaultUser)

Un utilisateur une fois enregistré et connecté, à accès à l'intégralité des fonctionnalités (hormis les fonctionnalités réservées aux administrateur) pour autant qu'il n'ait pas été bloqué. Ces permissions sont attribuées par le rôle qui lui est assigné.

1.2.4.3 Utilisateur enregistré bloqué (bloquedUser)

Un utilisateur bloqué possède des droites limitées aux fonctionnalités privées de CovoitEco. Il a toujours accès a son profile mais ne peux plus interagir avec les autres utilisateurs de l'application hormis l'administrateur. Ces permissions sont attribuées par le rôle qui lui est assigné.

1.2.4.3 Administrateur (admin)

Un administrateur possède les mêmes droits qu'un utilisateur enregistré et non-bloqué. Il a en outre accès à des fonctionnalités supplémentaire concernant la gestion des autre utilisateurs (hormis un administrateur) et de leur(s) éventuelle(s) véhicule(s). Ces permissions sont attribuées par le rôle qui lui est assigné.

1.2.5 Fonctionnalités

Références	Description	Intervenant(s)	Version
UC-1	Créer un compte	guest	Web
UC-2	Supprimer un compte	admin, defaultUser, bloquedUser	Web
UC-3	S'authentifier	admin, defaultUser, bloquedUser	Mobile / Web
UC-4	Se déconnecter	admin, defaultUser, bloquedUser	Mobile / Web
UC-5	Rechercher une annonce	admin, defaultUser, bloquedUser	Mobile / Web
UC-6	Publier une annonce	admin, defaultUser	Web
UC-7	Gérer ces annonces	admin, defaultUser, bloquedUser	Web
UC-8	Soumettre une demande de réservation	admin, defaultUser	Mobile / Web
UC-9	Gérer ces réservations	admin, defaultUser, bloquedUser	Mobile / Web
UC-10	Consulter une facture	admin, defaultUser, bloquedUser	Web
UC-11	Gérer les véhicules utilisateur	admin	Web
UC-12	Gérer ces véhicules	admin, defaultUser, bloquedUser	Web
UC-13	Enregistrer un véhicule	admin, defaultUser	Web
UC-14	Gérer son profil utilisateur	admin, defaultUser, bloquedUser	Web
UC-15	Gérer les profils utilisateur	admin	Web

UC-16	Evaluer un utilisateur	admin, defaultUser	Web
UC-17	Gérer son mot de passe	admin, defaultUser, bloquedUser	Web
UC-18	Consulter les informations de contact	admin, defaultUser, bloquedUser,guest	Web
UC-19	Consulter le FAQ	admin, defaultUser, bloquedUser,guest	Web
UC-20	Gérer mes messages	admin, defaultUser, bloquedUser	Web

1.2.6 Méthode de travail

- L'architecture du projet est en CQRS.
- Le backend et le frontend sont développés dans Visual studio 2022 en C#.
- Le template sélectionné pour le backend est ASP.NET Core Web API.
- Le frontend pour la partie web est Blazor assembly (modèle : Blazor WebAssmeblyApp).
- Le frontend pour la partie mobile est MAUI (modèle : .NET MAUI App).
- Pour la base de données, le SGBD utilisé est SQL server.

1.2.9 Conclusion

Dans ce chapitre j'ai abordé les points principaux concernant le projet. Il permet de définir dans les grandes lignes et les caractéristiques principales de la solution proposée.

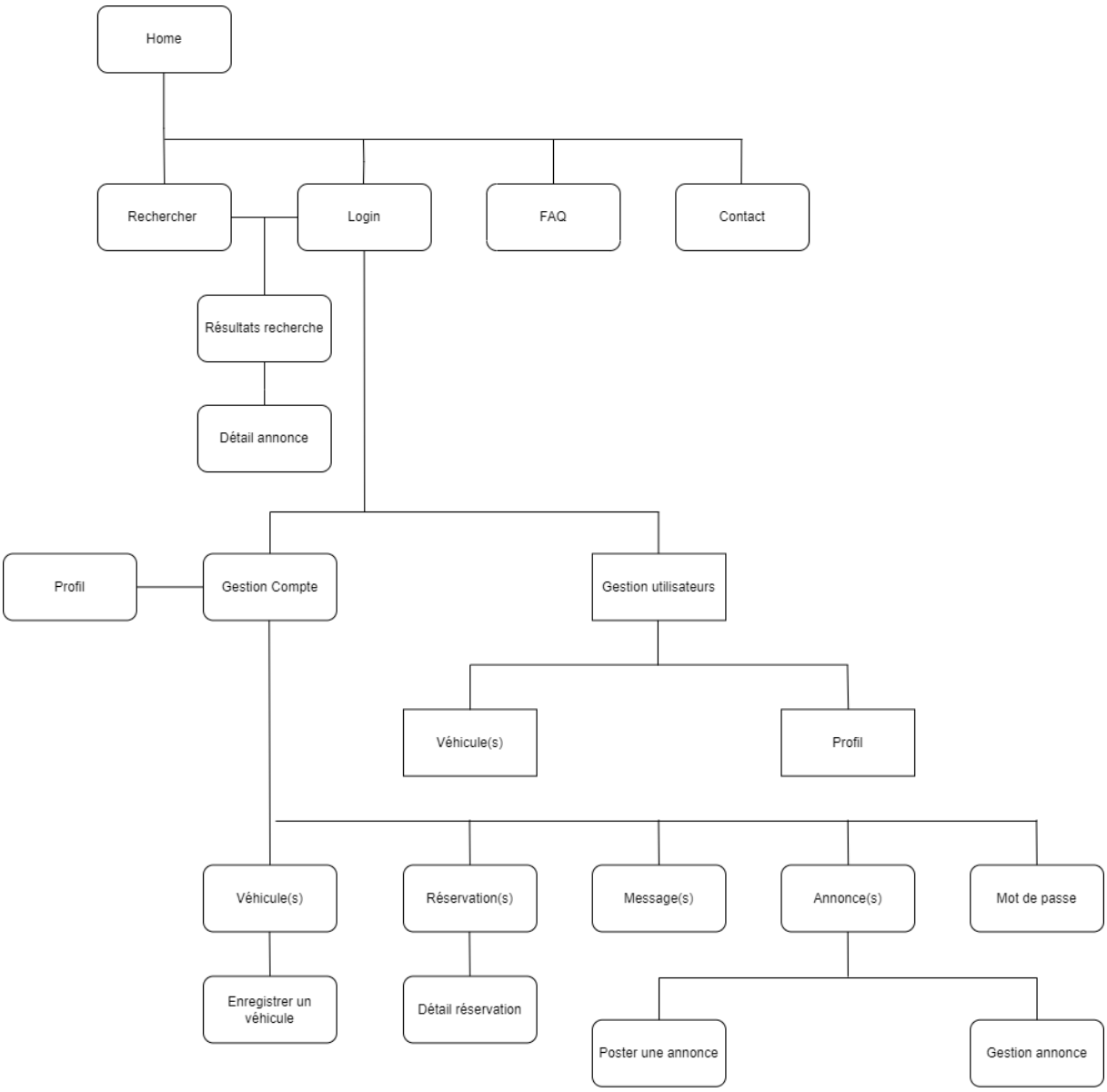
1.3 Analyse fonctionnelle

1.3.1 Introduction

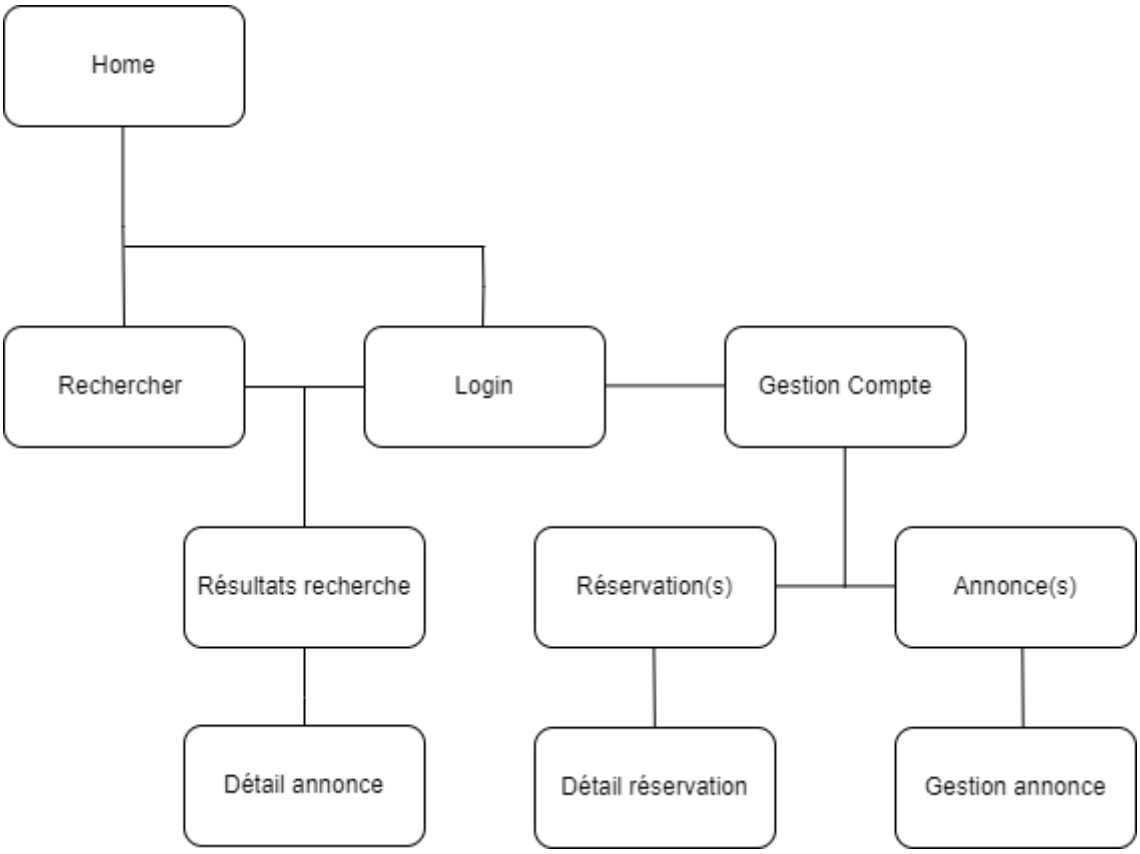
Ce chapitre contient les diagramme d'use cases et de navigations. Les use cases y sont notamment aborder de fonçons détaillés. Les contraintes (logique métier) y sont également définies, ainsi que différents schémas représentant la base de données.

1.3.2 Diagramme de navigation

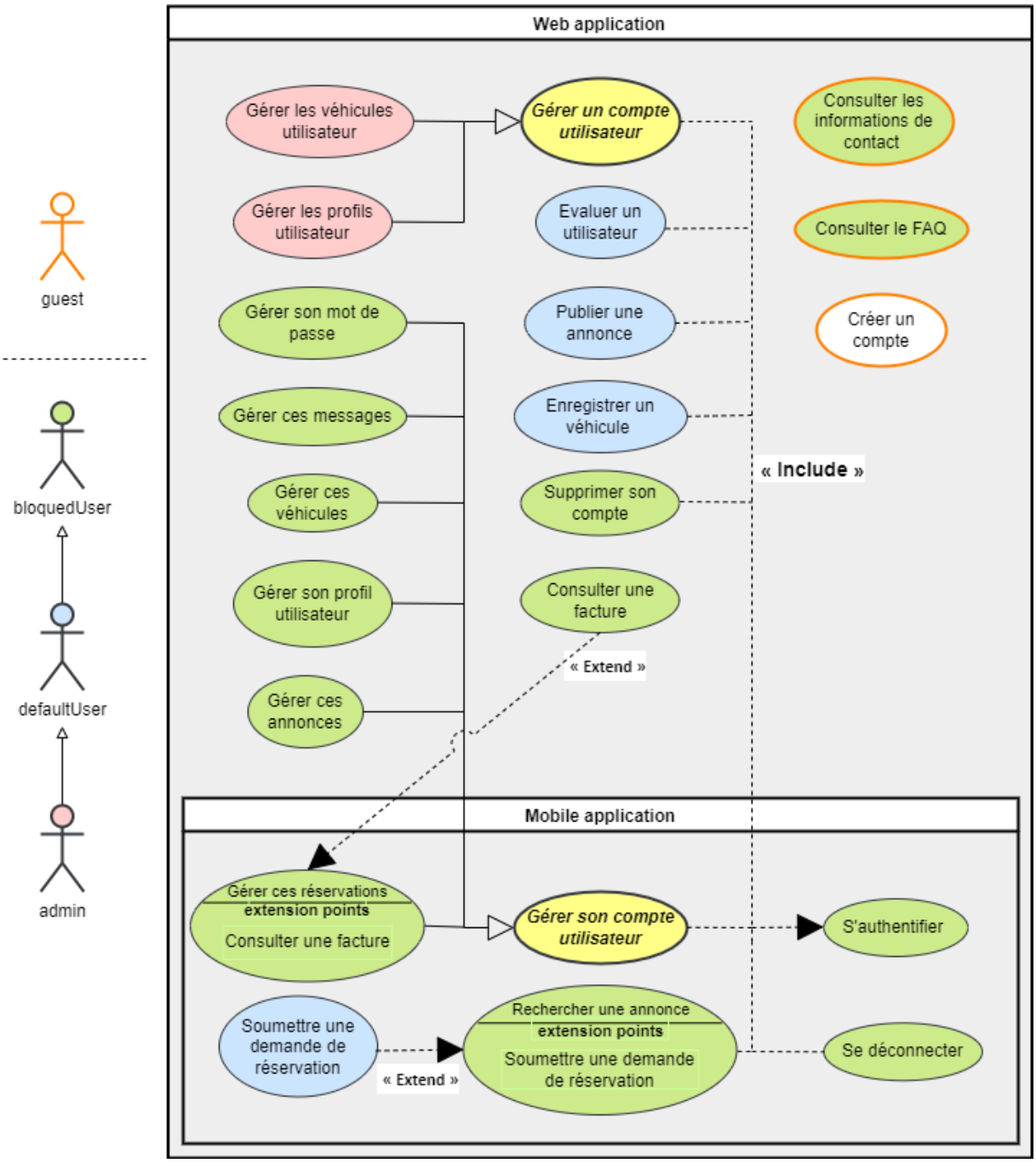
1.3.2.1 Web



1.3.2.2 Mobile


Home
Rechercher
Login
Gestion Compte
Résultats recherche
Réservation(s)
Annonce(s)
Détail annonce
Détail réservation
Gestion annonce

1.3.3 Diagramme des use cases



1.3.4 Description des use cases

1.3.4.1 Créer un compte

Titre : Créer un compte
Résumé : Un utilisateur crée un compte.
Acteur : utilisateur *(guest)*

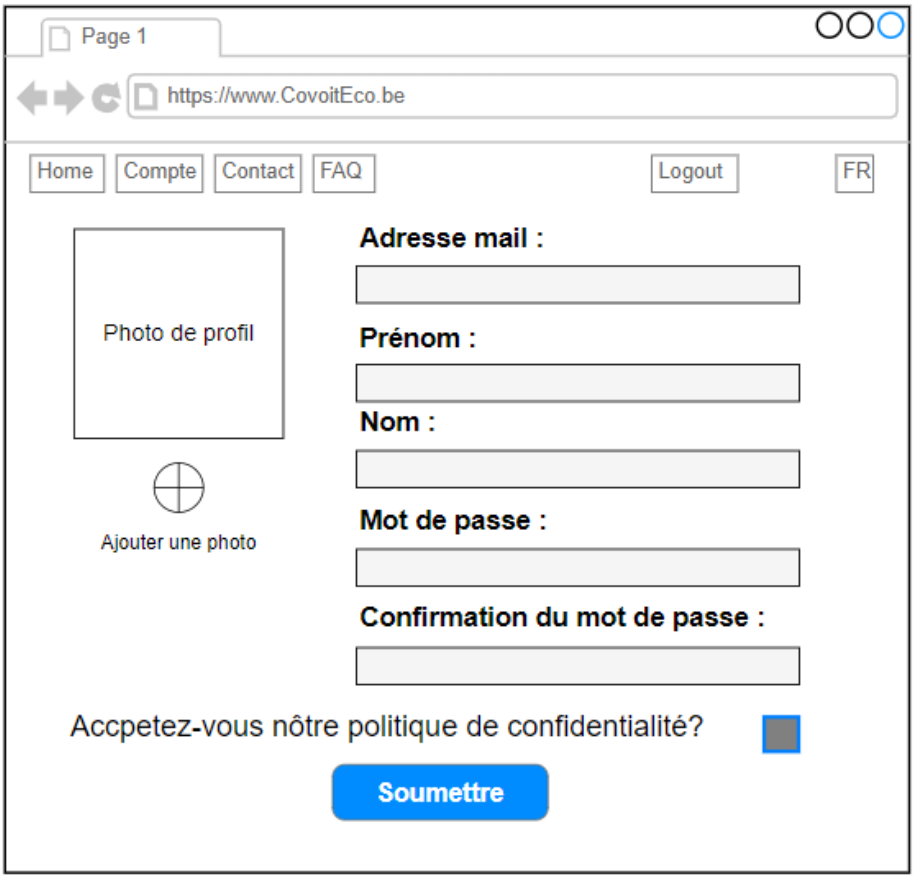
Préconditions :
- L'utilisateur n'est pas connecté.
- L'application Web est ouverte.

Scénario nominal :
1. L'utilisateur appuie sur le bouton « s'inscrire ».
2. Le système affiche le formulaire d'inscription.
3. L'utilisateur une fois le formulaire complété clique sur le bouton « soumettre ».
4. Le système confirme l'inscription.

Enchaînement d'erreurs :
E1. Le compte ne peut pas être crée car l'adresse mail est déjà utilisée.
Démarre au point 3 du scénario nominal.
4. Le système informe que la création à échoué car un compte est déjà associé à cette adresse.
Le scénario nominal prend fin.

Post-conditions :
- Un compte a été créer pour l'utilisateur.

Écrans :



1.3.4.2 Supprimer un compte

Titre : Supprimer un compte

Résumé : Un utilisateur supprime son compte.

Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :

- L'utilisateur est connecté sur la version Web.
- L'utilisateur se trouve dans la section compte.

Scénario nominal :

1. L'utilisateur appuie sur l'onglet « profil ».
2. Le système affiche le profil de l'utilisateur.
3. L'utilisateur appuie sur le bouton « supprimer ce compte ».
4. Le système demande confirmation à l'utilisateur.
5. L'utilisateur confirme.
6. Le système confirme la suppression et redirige l'utilisateur vers l'écran d'accueil.

Enchaînement d'erreurs :

E1. La suppression a échoué pour divers raison (annonce ouvertes, réservations ouverte, inscription de véhicule en cours).

Démarre au point 5 du scénario nominal.

6. Le système confirme l'échec de la suppression ainsi que la raison de cet échec et redirige l'utilisateur vers l'écran d'accueil.

Post-conditions :

- Le compte de l'utilisateur a été supprimé dans Auth0 et modifier dans la base de données.

Écrans :

1.3.4.3 S'authentifier

Titre : S'authentifier

Résumé : Un utilisateur s'authentifie sur l'application web/moblie.

Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :

- L'utilisateur n'est pas authentifié.
- L'utilisateur se trouve sur la page d'accueil.

Scénario nominal :
1. L'utilisateur appuie sur le bouton « Login ».
2. Le système redirige l'utilisateur vers l'interface d'Auth0.
3. L'utilisateur entre ses informations de connexion.
4. L'utilisateur appuie sur le bouton « continue ».
5. Auth0 valide la connexion et redirige l'utilisateur vers l'application.
6. Le système affiche la page d'accueil.

Enchaînement d'erreurs :
E1. Auth0 ne valide pas la connexion car le mot de passe ou l'adresse est incorrect.
Démarre au point 4 du scénario nominal.
5. Auth0 ne valide pas la connexion et signale à l'utilisateur que le mode de passe ou l'adresse est incorrecte.

Post-conditions :
- L'utilisateur est authentifier sur l'application.
- Le système a bien reçu le token Auth0.

Écrans :
(cf. *2.8.1 Auth0*)

1.3.4.4 Se déconnecter

Titre : Se déconnecter
Résumé : Un utilisateur se déconnecte de l'application web/mobile.
Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :
- L'utilisateur est authentifié

Scénario nominal :
1. L'utilisateur appuie sur le bouton « Logout ».
2. Le système confirme la déconnexion.
3. L'utilisateur est renvoyé vers la page d'accueil.

Post-conditions :
- L'utilisateur est déconnecté.

Écrans :

1.3.4.5 Rechercher une annonce

Titre : Rechercher une annonce
Résumé : L'utilisateur recherche une annonce.
Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :
- L'utilisateur est authentifié.
- L'utilisateur se trouve sur la page d'accueil.

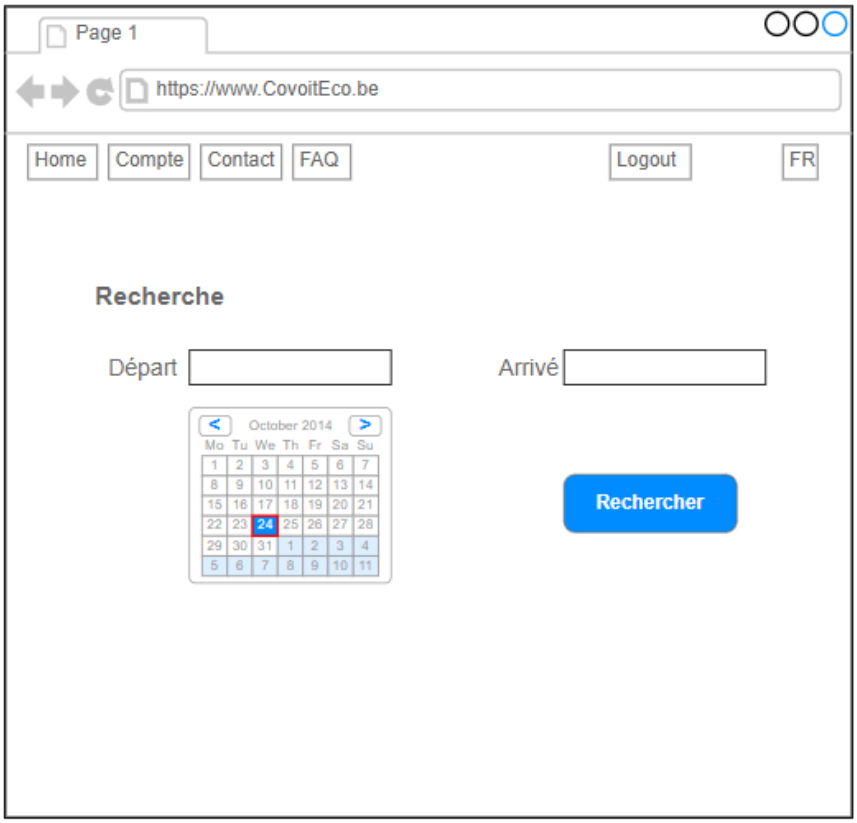
Scénario nominal :
1. L'utilisateur saisie les critères de recherche.
2. L'utilisateur appuie sur le bouton « rechercher ».
3. Le système affiche les annonces correspondantes.
4. L'utilisateur appuie sur le bouton « détail annonce ».
5. Le système affiche les détails de l'annonces.

Scénarios alternatifs :
A1. L'utilisateur ne saisit aucun critère de recherche.
Démarre au point 0 du scénario nominal.
1. L'utilisateur ne saisit pas les informations de recherche.
Le scénario nominal reprend au point 2.

A2. Aucune annonce ne correspond aux critères de recherches.
Démarre au point 2 du scénario nominal.
3. Le système n'affiche aucun résultat.
Le scenario nominal prend fin.

Post-conditions :
- N/A

Écrans :



1.3.4.6 Publier une annonce

Titre : Publier une annonce

Résumé : L'utilisateur publie une annonce.

Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :

- L'utilisateur a enregistré au moins un véhicule.
- Le véhicule est disponible.
- Le véhicule est autorisé par un admin.
- L'utilisateur est authentifié.

Scénario nominal :

1. L'utilisateur navigue vers la section « Annonce(s) ».
2. L'utilisateur appuie sur le bouton « publier ».
3. Le système redirige l'utilisateur vers le formulaire de publication d'annonce.
4. L'utilisateur appuie sur le bouton « soumettre » après avoir complété le formulaire.
5. Le système confirme la création de l'annonce et redirige l'utilisateur dans la section « Annonce(s) ».

Enchaînement d'erreurs :

E1. La création de l'annonce échoue pour divers raisons (véhicule associé à une autre annonce ou date choisis en conflit avec un annone déjà existant).

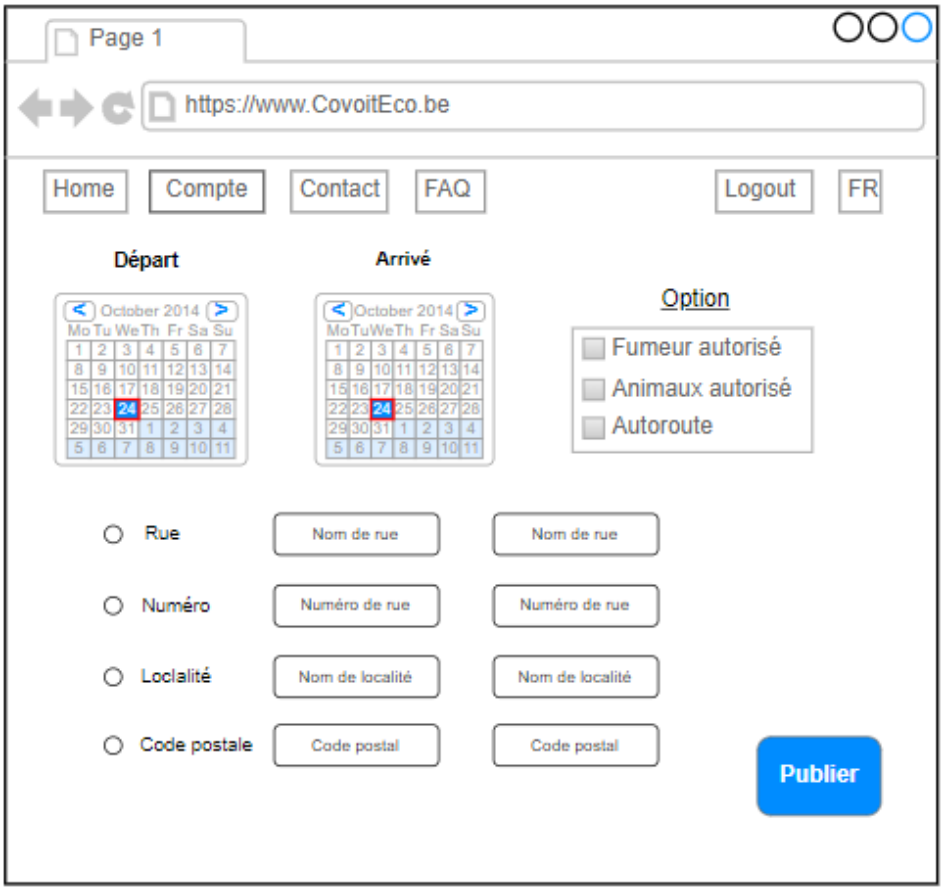
Démarre au point 4 du scénario nominal.

5. Le système invalide le formulaire et indique la raison de l'échec.

Post-conditions :

- L'annonce est affiché dans la section « Annonce(s) ».

Écrans :



1.3.4.7 Gérer ces annonces

Titre : Gérer ces annonces

Résumé : L'utilisateur gére ces annonces.

Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :

- L'utilisateur est authentifié.
- L'utilisateur possède au moins une annonce.

Scénario nominal :

1. L'utilisateur navigue vers la section « Annonce(s) ».
2. Le système affiche la liste de annonces de l'utilisateur.
3. L'utilisateur appuie sur le bouton « administrer » de l'annonce sélectionnée.
4. Le système affiche les détails de l'annonce et la liste des réservations correspondante.
5. L'utilisateur exécute l'action désiré.
    - Appuyer sur le bouton « accepter la demande ».
    - Appuyer sur le bouton « confirmer le payement ».
6. Le système confirme l'exécution de la commande.

Scénarios alternatifs :

A1. Il n'y a aucune réservation correspondante.

Démarre au point 3 du scénario nominal.

4. Le système affiche les détails de l'annonce

Le scénario nominal prend fin.

A2. Aucune action n'est exécutée ou exécutable.

Démarre au point 4 du scénario nominal.

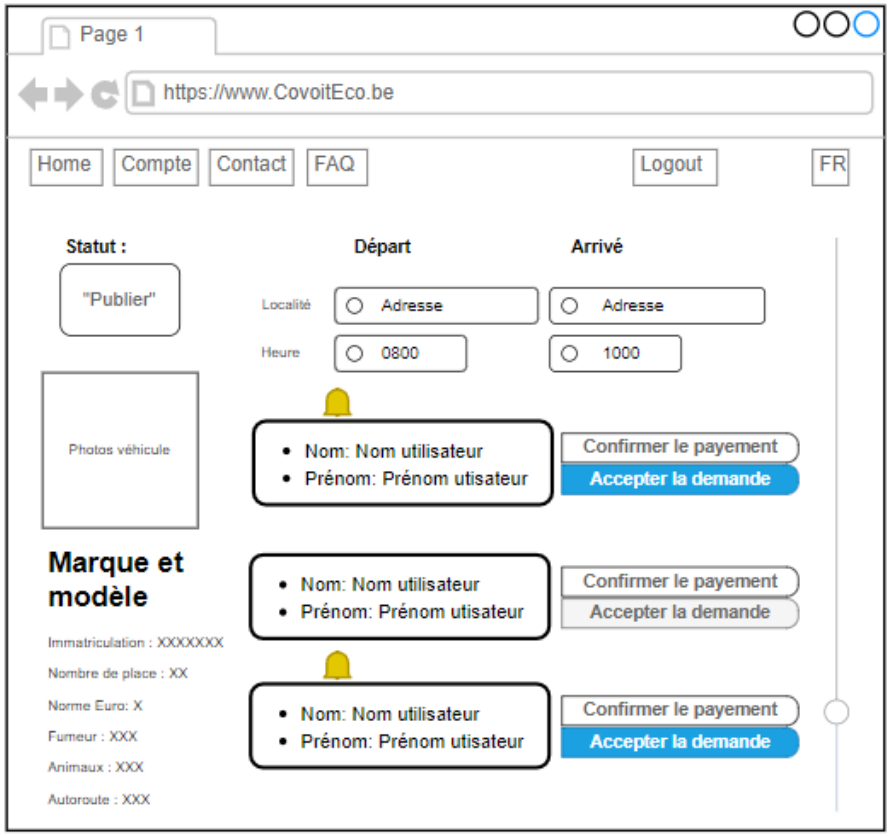
5. L'utilisateur n'effectue aucune action.

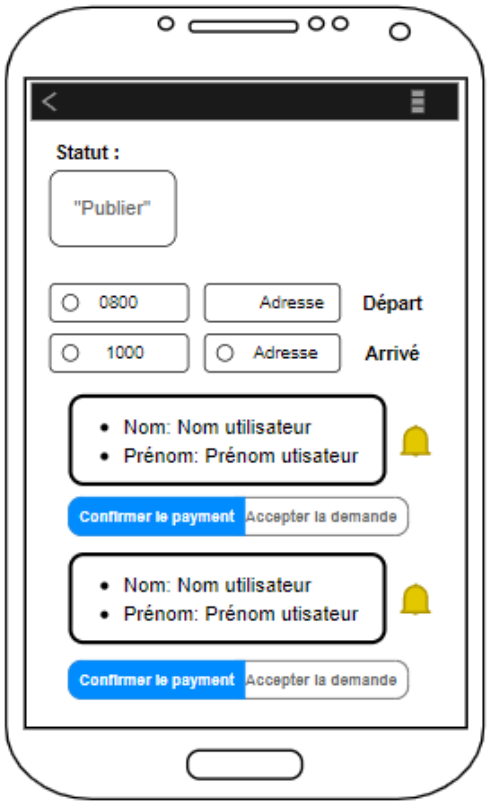
Le scénario nominal prend fin.

Post-conditions :

- Les données ont été mise à jour (scénario nominal uniquement).

Écrans :





1.3.4.8 Soumettre une demande de réservation

Titre : Soumettre une demande de réservation

Résumé : L'utilisateur soumet une demande de réservation pour une annonce.

Acteurs : utilisateur (*admin* ou *defautlUser*)

Préconditions :

- L'utilisateur est authentifié.
- L'utilisateur est dans le détail d'une annonce.

Scénario nominal :

1. L'utilisateur clique sur le bouton « soumettre vôtre demande ».
2. Le système confirme la création d'une réservation et redirige l'utilisateur vers la page d'accueil.

Enchaînement d'erreurs :

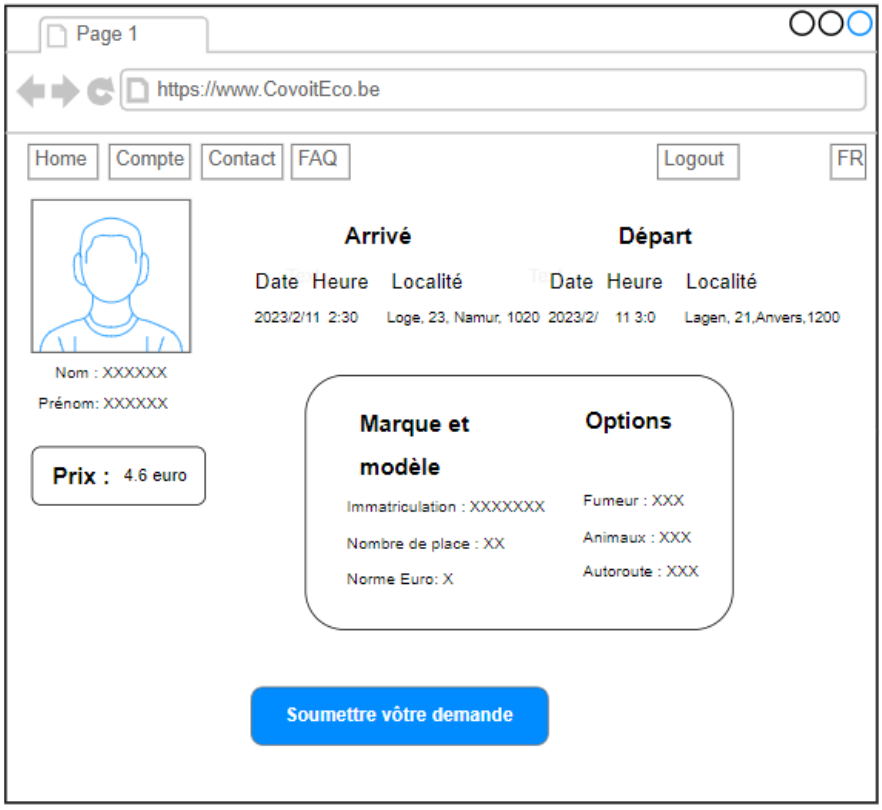
E1. La création d'une réservation à échoué pour diverses raisons (plus de place disponible ou délai de réservation dépassé).

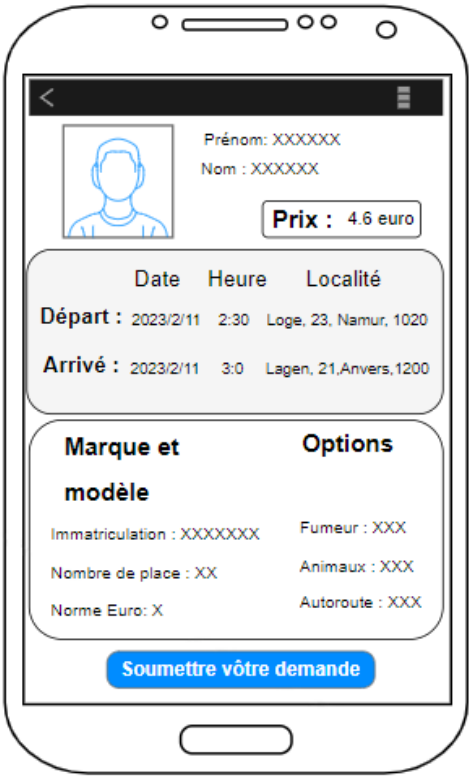
Démarre au point 1 du scénario nominal.

1. Le système invalide la création d'une réservation en indiquant les raisons, et l'utilisateur est redirigé vers la page d'accueil.

Post-conditions :
- Une réservation a été créé pour l'utilisateur et l'annonce correspondante.

Écrans :





1.3.4.9 Gérer ces réservations

Titre : Gérer ces réservations
Résumé : L'utilisateur gère ces réservations.
Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :
- L'utilisateur est authentifié.
- L'utilisateur possède au moins une réservation.

Scénario nominal :
1. L'utilisateur navigue vers la section « Réservation(s) ».
2. Le système affiche la liste des réservations de l'utilisateur.
3. L'utilisateur appuie sur le bouton « détail » de l'annonce sélectionnée.
4. Le système affiche les détails de l'annonce et la liste des réservations correspondante.
5. L'utilisateur exécute l'action désiré.
    - Appuyer sur le bouton « accepter la demande ».

- Appuyer sur le bouton « confirmer le payement ».

6. Le système confirme l'exécution de la commande.
7. Le système envoie une notification à l'utilisateur.

Scénarios alternatifs :

A1. L'utilisateur clique sur le bouton « détail ».

Démarre au point 4 du scénario nominal.

5. Le système affiche les détails de la réservation correspondante.

Le scénario nominal prend fin.

Enchaînement d'erreurs :

E1. L'utilisateur effectue une action (accepter/confirmer) sur une autre session.

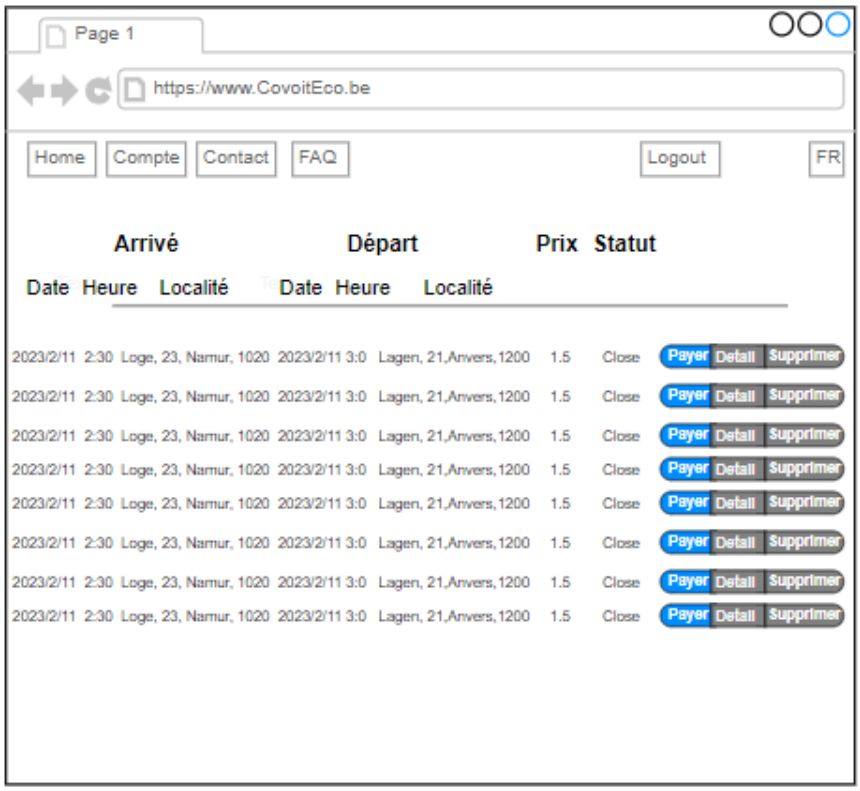
Démarre au point 5 du scénario nominal.

6. Le système invalide l'action et affiche la description du problème rencontré.

Post-conditions :

- Une réservation a été créé pour l'utilisateur et l'annonce correspondante.

Écrans :



1.3.4.10 Consulter une facture

Titre : Consulter une facture

Résumé : l'utilisateur consulte une facture.

Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :

- L'utilisateur est authentifié.
- L'utilisateur est sur une page « Détail de réservation »

Scénario nominal :

1. L'utilisateur appuie sur le bouton « exporter ».
2. Le système sauvegarde la facture dans le dossier « Downloads » du PC utilisateur, et affiche le PDF dans la barre des documents téléchargés.
3. L'utilisateur clique sur la facture.
4. Le système ouvre le PDF dans un nouvel onglet du navigateur.

Scénarios alternatifs :

A1. L'utilisateur ne sélectionne aucune facture

Démarre au point 2 du scénario nominal.

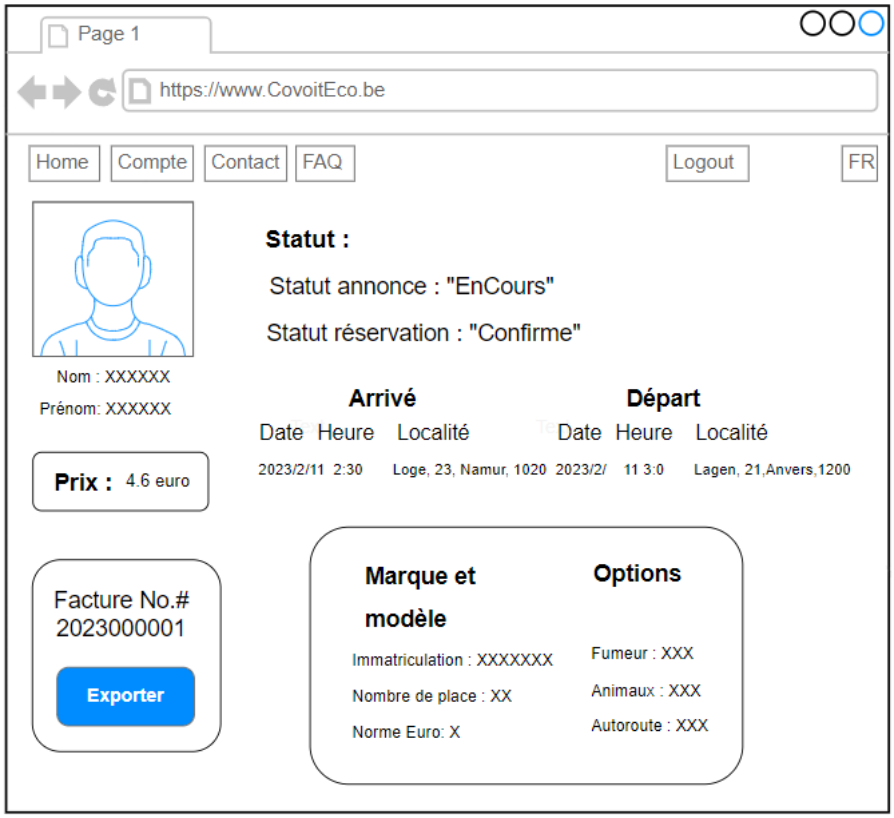
3. L'utilisateur n'ouvre aucune facture.

Le scénario nominal prend fin.

Post-conditions :

- La facture a été sauvegardé dans le dossier download.

Écrans :



1.3.4.11 Gérer les véhicules utilisateur

Titre : Gérer les véhicules utilisateur

Résumé : L'administrateur gère un ou plusieurs véhicule(s) utilisateurs.

Acteurs : administrateur

Préconditions :

- L'administrateur est authentifié.
- Au moins un utilisateur (admin compris) possède un véhicule non confirmé.

Scénario nominal :

1. L'administrateur navigue vers la section « Gestions utilisateurs ».
2. Le système affiche la liste de tous les utilisateurs.
3. L'administrateur appuie sur le bouton « véhicule » pour un utilisateur correspondant.
4. Le système affiche la liste du ou des véhicule(s) de l'utilisateur.
5. L'administrateur appuie sur le bouton « autoriser » pour un véhicule.
6. Le système confirme l'acception du véhicule.
7. Le système envoie une notification à l'utilisateur.

Scénarios alternatifs :

A1. Aucune liste de véhicule n'est sélectionnée.

Démarre au point 2 du scénario nominal.

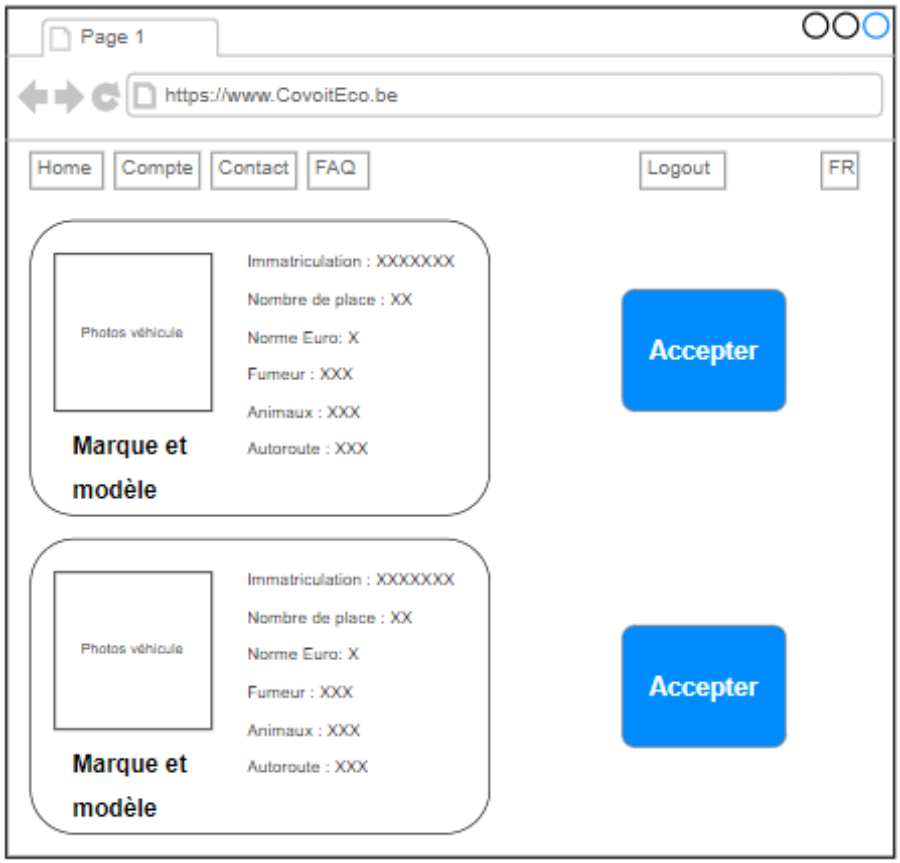
3. L'administrateur ne clique sur aucun bouton « véhicule »
Le scénario nominal prend fin.

A2. Aucun véhicule n'est autorisé.
Démarre au point 4 du scénario nominal.
5. L'administrateur ne clique sur aucun bouton « autoriser » pour un véhicule
Le scénario nominal prend fin.

Enchaînement d'erreurs :
E1. Le véhicule sélectionné a été validé par un autre administrateur.
Démarre au point 5 du scénario nominal.
6. Le système informe que l'véhicule est déjà confirmé.

Post-conditions :
- Le véhicule autorisé est maintenant enregistré comme disponible.
- Une notification a été envoyé à l'utilisateur (scénario nominal uniquement).

Écrans :



1.3.4.12 Gérer ces véhicules

Titre : Gérer ces véhicules

Résumé : L'utilisateur gère ces véhicules.

Acteurs : utilisateur (*admin*, *defautlUser* ou *bloquedUser*)

Préconditions :

- L'utilisateur est authentifié.
- L'utilisateur possède au moins une voiture.

Scénario nominal :

1. L'utilisateur navigue vers la section « Véhicule(s) ».
2. Le système affiche la liste des véhicules de l'utilisateur.
3. L'utilisateur clique sur un bouton « courant ».
4. Le système confirme l'exécution de la commande.

Scénarios alternatifs :

A1. Aucun bouton « courant » n'est sélectionné.

Démarre au point 3 du scénario nominal.

4. L'utilisateur ne clique sur aucun bouton « courant »

Le scénario nominal prend fin.

Enchaînement d'erreurs :

E1. Le véhicule courant actuel est lié à une annonce non close.

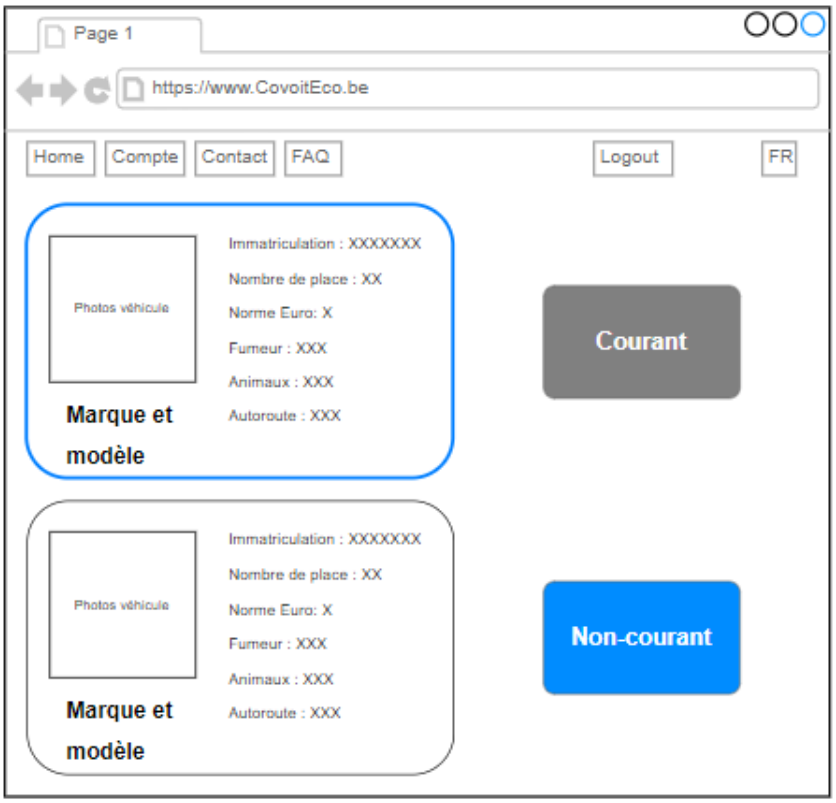
Démarre au point 3 du scénario nominal.

6. Le système invalide l'action et affiche la description du problème rencontré.

Post-conditions :

- Le statut des deux véhicules concernés a été modifié.

Écrans :



1.3.4.13 Enregistrer un véhicule

Titre : Enregistrer un véhicule

Résumé : L'utilisateur enregistre un nouveau véhicule.

Acteurs : utilisateur (*admin* ou *defautlUser*)

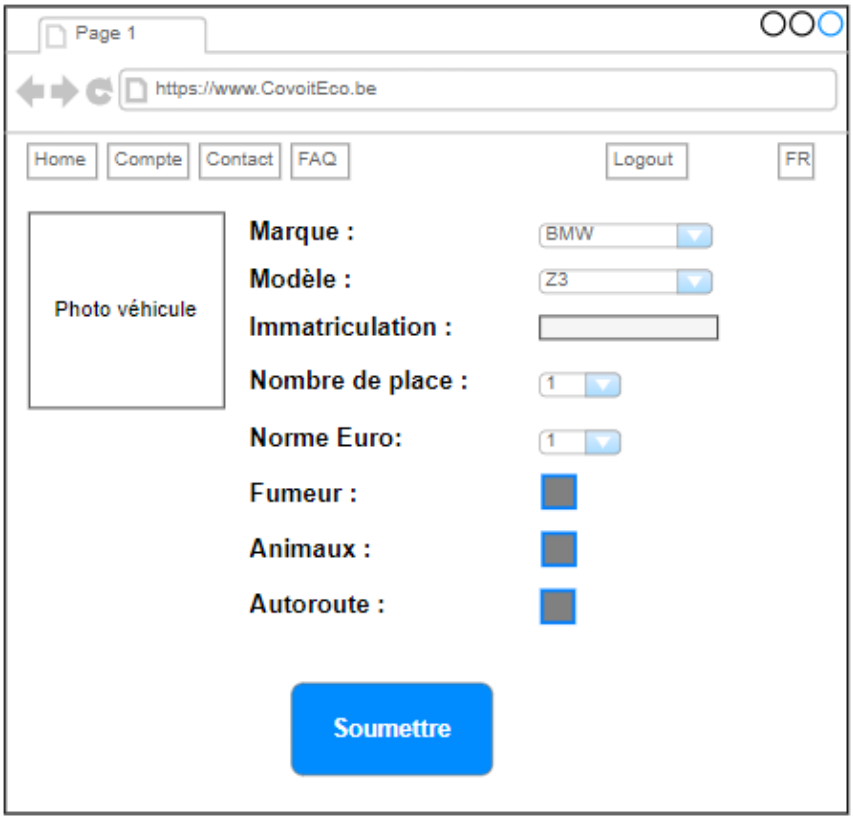
Préconditions :
- L'utilisateur est authentifié.
- L'utilisateur est dans la section « Véhicule(s) ».

Scénario nominal :
1. L'utilisateur clique sur le bouton « enregistrer ».
2. Le système affiche le formulaire d'enregistrement.
3. L'utilisateur clique sur le bouton « enregistrer ».
4. Le système confirme l'exécution de la commande.
5. Le système envoie une notification a tous les administrateurs.

Enchaînement d'erreurs :
E1. Un véhicule identique est déjà enregistré.
Démarre au point 3 du scénario nominal.
4. Le système invalide l'action et affiche la description du problème rencontré.

Post-conditions :
- Le véhicule a été enregistré dans la base de données.
- Une notification a été envoyé à tous les administrateurs.

Écrans :



1.3.4.14 Gérer son profil utilisateur

Titre : Gérer son profil utilisateur

Résumé : L'utilisateur gère son profil.

Acteurs : utilisateur (*admin* ou *defautlUser*)

Préconditions :

- L'utilisateur est authentifié.

Scénario nominal :

1. L'utilisateur navigue vers la section « Profil ».
2. Le système affiche la liste des données profil de l'utilisateur.
3. L'utilisateur effectue une modification de son profil.
    - Modifier l'adresse mail.
    - Modifier la photo de profil.
    - Modifier son N° de compte IBAN
    - Modifier son adresse de facturation
4. L'utilisateur appuie sur le bouton « sauvegarder ».
5. Le système lui demande confirmation.
6. Le système confirme la modification.

Enchaînement d'erreurs :

E1. L'adresse mail n'est pas disponible ou incorrect.

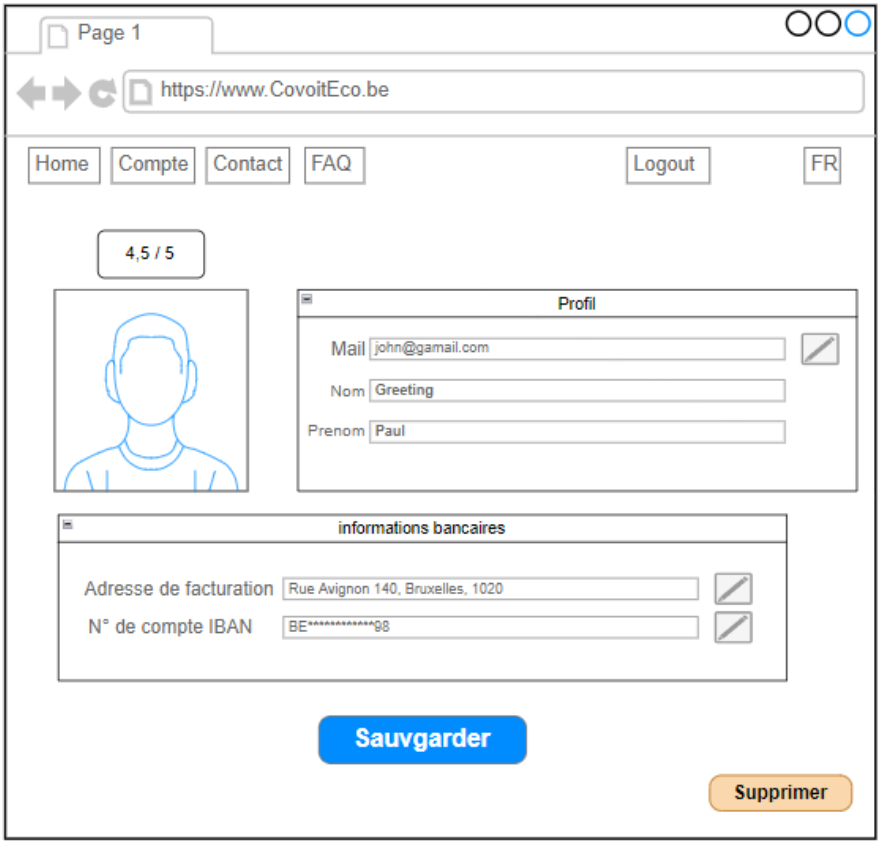
Démarre au point 5 du scénario nominal.

6. Le système invalide la modification et renvoie la raison du problème.

Post-conditions :

- La modification a été sauvegardé au niveau d'Auth0 et/ou de la base de données.

Écrans :



1.3.4.15 Gérer un profil utilisateur

Titre : Gérer un profil utilisateur

Résumé : L'administrateur gère un profil utilisateur.

Acteurs : administrateur

Préconditions :

- L'administrateur est authentifié.

Scénario nominal :

1. L'administrateur navigue vers la section « Gestions utilisateurs ».
2. Le système affiche la liste de tous les utilisateurs.
3. L'administrateur appuie sur le bouton « profil » pour un utilisateur correspondant.
4. Le système affiche les données profil de l'utilisateur correspondant.
5. L'administrateur sélectionne un rôle.
6. L'administrateur appuie sur le bouton « appliquer ».
7. Le système demande confirmation.
8. L'administrateur confirme son choix.
9. Le système confirme la modification.

Scénarios alternatifs :

A1. L'administrateur n'appuie sur aucun bouton « profil ».
Démarre au point 2 du scénario nominal.
3. L'administrateur ne clique pas sur un bouton « profil ».
Le scénario nominal prend fin.

A2. Un ou plusieurs signalement existe pour l'utilisateur
Démarre au point 4 du scénario nominal.
5. L'administrateur clique sur un bouton « supprimer » pour un signalement.
6. Une notification est envoyée à l'utilisateur fessant objet du signalement.
Le scénario nominal prend fin.

A3. L'administrateur n'appuie sur aucun bouton « appliquer ».
Démarre au point 5 du scénario nominal.
6. L'administrateur ne clique pas sur un bouton « appliquer ».
Le scénario nominal prend fin.

Enchaînement d'erreurs :

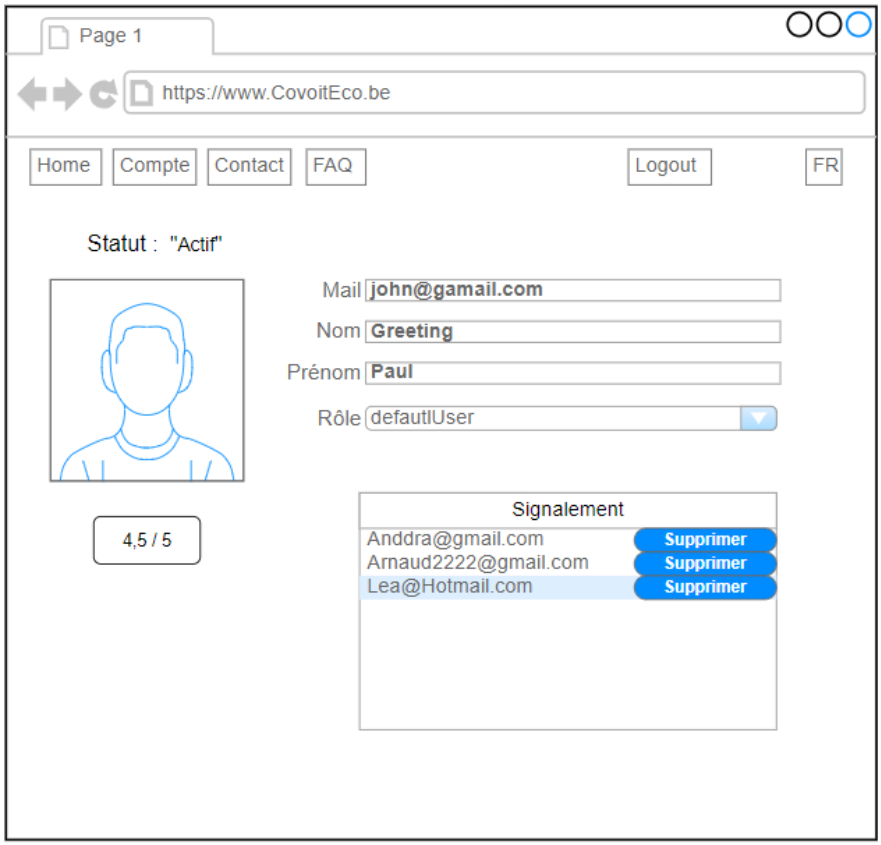
E1. Le système invalide la modification car l'utilisateur vient d'être supprimé.
Démarre au point 8 du scénario nominal.
5. Le système invalide la modification et renvoie la raison du problème.

E2. Un autre administrateur a annulé la suppression.
Démarre au point 8 du scénario alternatif **A2**.
9. Le système invalide la modification et renvoie la raison du problème.

Post-conditions :

- Le rôle de l'utilisateur a été adapté dans Auth0 et dans la base de données.

Écrans :



1.3.4.16 Évaluer un utilisateur

Titre : Évaluer un utilisateur

Résumé : L'utilisateur évalue un autre utilisateur.

Acteurs : utilisateur (*admin* ou *defautlUser*)

Préconditions :

- L'utilisateur est authentifié.
- L'utilisateur possède un message de type évaluation.

Scénario nominal :

1. L'utilisateur sélectionne un message de type évaluation.
2. Le système affiche le contenu du message.
3. L'utilisateur appuie sur le bouton « évaluer ».
4. Le système affiche un pop-up.
5. L'utilisateur sélectionne une note.
6. L'utilisateur appuie sur le bouton « valider ».
7. Le système ferme le pop-up et remercie l'utilisateur pour son vote.

Scénarios alternatifs :

A1. L'utilisateur n'appuie pas sur le bouton « évaluer ».
Démarre au point 2 du scénario nominal.
3. L'utilisateur ne clique pas sur le bouton « évacuer ».
Le scénario nominal prend fin.

A2. L'utilisateur ne sélectionne aucune note.
Démarre au point 4 du scénario nominal.
5. L'utilisateur ne clique pas sur une note.
Le scénario nominal prend fin.

A3. L'utilisateur n'appuie sur aucun bouton « valider ».
Démarre au point 5 du scénario nominal.
6. L'utilisateur ne clique pas sur le bouton « valider ».
Le scénario nominal prend fin.

A4. L'utilisateur à valider une note égale à 1.
Démarre au point 6 du scénario nominal.
7. Le système demande à l'utilisateur s'il souhaite signaler l'utilisateur évalué.
8. L'utilisateur appuie sur le bouton « oui ».
9. Le système envoie une notification à l'utilisateur évalué et aux administrateurs.
Le scénario nominal reprend au point 7.

A5. L'utilisateur ne souhaite pas signaler l'utilisateur évalué.
Démarre au point 7 du scénario alternatif **A4**.
8. L'utilisateur appuie sur le bouton « non ».
Le scénario nominal reprend au point 7.

Enchaînement d'erreurs :

E1. Le système invalide l'évaluation pour divers raisons (utilisateur déjà évalué ou compte de l'utilisateur évalué inactif).
Démarre au point 6 du scénario nominal.
7. Le système invalide l'évaluation, ferme le pop-up et renvoie l'origine du problème.

Post-conditions :

- La note de l'utilisateur évalué a été modifié.
- L'utilisateur évalué et les administrateurs ont été notifié du signalement.

Écrans :

1.3.4.17 Gérer son mot de passe

Titre : Gérer son mot de passe

Résumé : L'utilisateur modifie son mot de passe.

Acteurs : utilisateur (*admin* ou *defautlUser*)

Préconditions :

- L'utilisateur est authentifié.

Scénario nominal :

1. L'administrateur navigue vers la section « Mot de passe ».
2. Le système affiche un formulaire de modification de mot de passe.
3. L'utilisateur complète le formulaire.
4. L'utilisateur appuie sur le bouton « valider ».
5. Le système confirme la modification du mot de passe.

Scénarios alternatifs :

A1. L'utilisateur n'appuie pas sur le bouton « valider ».

Démarre au point 3 du scénario nominal.

4. L'utilisateur ne clique pas sur le bouton « valider ».

Le scénario nominal prend fin.

Enchaînement d'erreurs :

E1. Le mot de passe a été modifié sur une seconde cession.

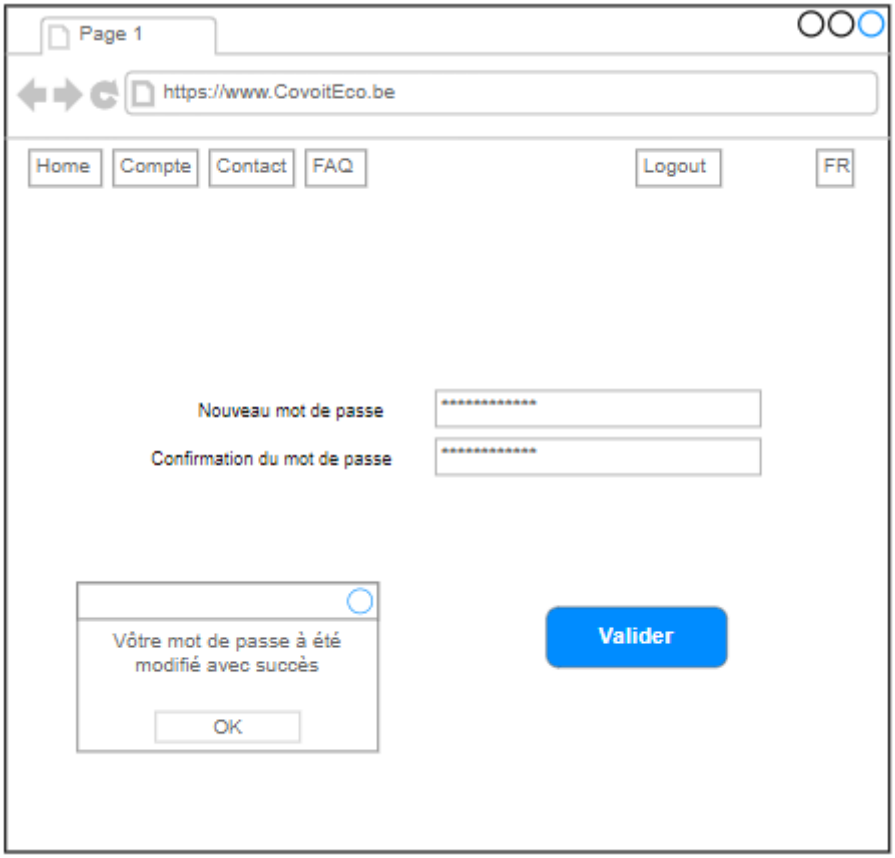
Démarre au point 4 du scénario nominal.

5. Le système invalide la modification et renvoie la raison du problème.

Post-conditions :

- Le mot de passe de l'utilisateur a été adapté dans Auth0 (scénario nominal uniquement).

Écrans :



1.3.4.18 Consulter les informations de contact

Titre : Consulter les informations de contact

Résumé : L'utilisateur consulte les informations de contact.

Acteurs : utilisateur (*admin*, *defautlUser, bloquedUser ou guest*)

Préconditions :

-N/A

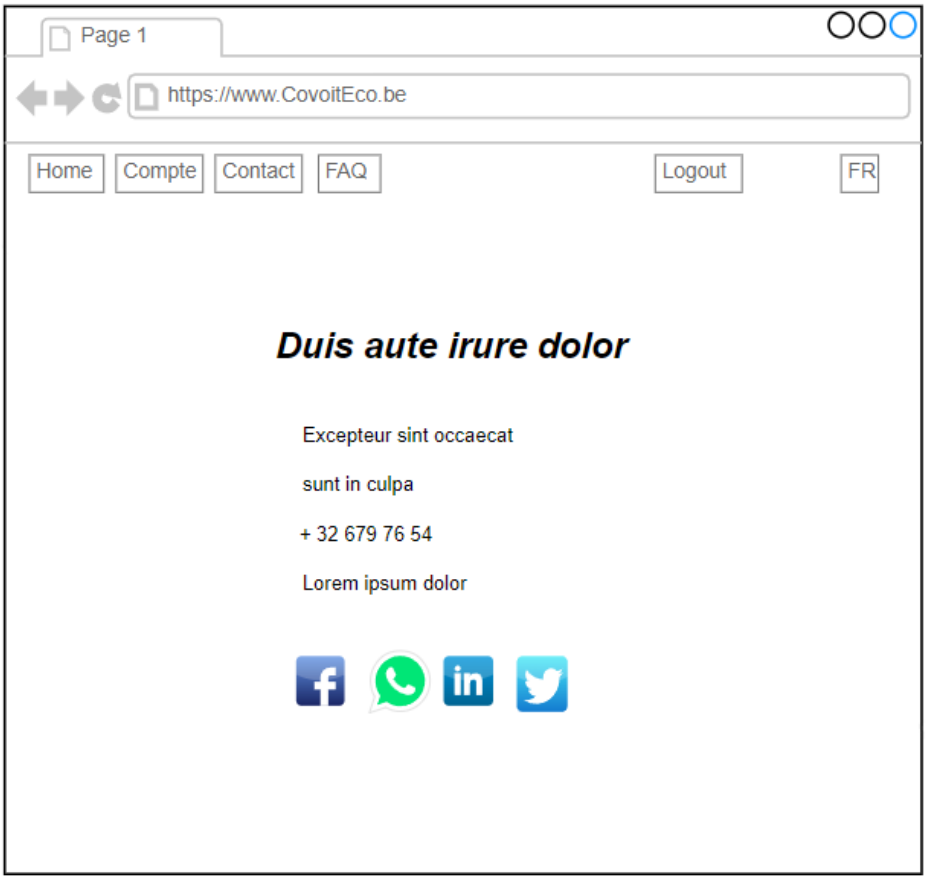
Scénario nominal :

1. L'utilisateur navigue vers la section « Contact ».
2. Le système affiche les informations de contact.

Post-conditions :

- N/A

Écrans :



1.3.4.19 Consulter le FAQ

Titre : Consulter le FAQ

Résumé : L'utilisateur consulte le FAQ.

Acteurs : utilisateur (*admin*, *defautlUser, bloquedUser ou guest*)

Préconditions :

- L'utilisateur est authentifié.

Scénario nominal :

1. L'administrateur navigue vers la section « FAQ ».
2. Le système affiche la liste des sujets.
3. L'utilisateur sélectionne une question.
4. Le système affiche les questions réponses correspondante.

Scénarios alternatifs :

A1. L'utilisateur ne sélectionne aucun sujet.

Démarre au point 2 du scénario nominal.

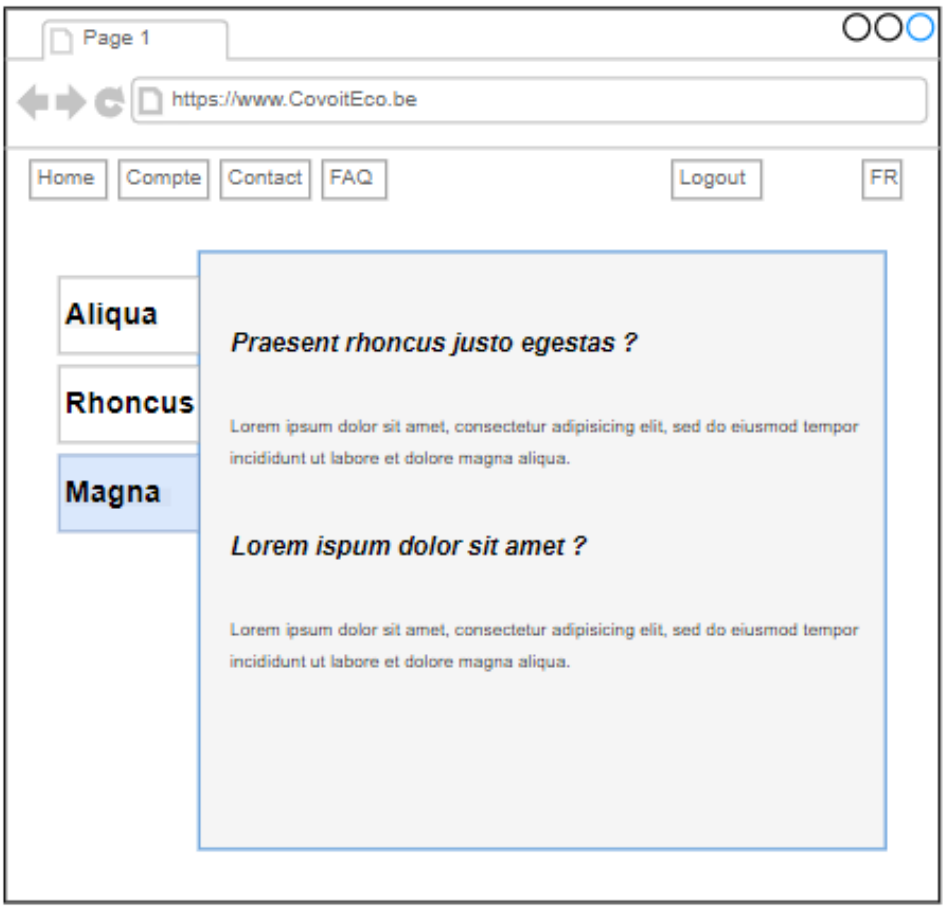
3. L'utilisateur ne sélectionne aucun sujet.

Le scénario nominal prend fin.

Post-conditions :

- N/A

Écrans :



1.3.4.20 Gérer ces messages

Titre : Gérer ces messages

Résumé : L'utilisateur gère ces messages.

Acteurs : utilisateur (*admin, defautlUser ou bloquedUser*)

Préconditions :

- L'utilisateur est authentifié.
- L'utilisateur possède au moins un message (notification).

Scénario nominal :

1. L'utilisateur navigue vers la section « Message(s) ».
2. Le système affiche la liste des messages.
3. L'utilisateur appuie sur un bouton « ouvrir ».
4. Le système affiche les contenus du message correspondant.
5. L'utilisateur appuie sur le bouton « supprimer » du message correspondant.
6. Le système supprime le message correspondant.

Scénarios alternatifs :

A1. L'utilisateur n'appuie pas sur le bouton « ouvrir ».
Démarre au point 2 du scénario nominal.
3. L'utilisateur ne clique pas sur le bouton « ouvrir ».
Le scénario nominal prend fin.

A2. L'utilisateur n'appuie pas sur le bouton « supprimer ».
Démarre au point 4 du scénario nominal.
3. L'utilisateur ne clique pas sur le bouton « supprimer ».
Le scénario nominal prend fin.

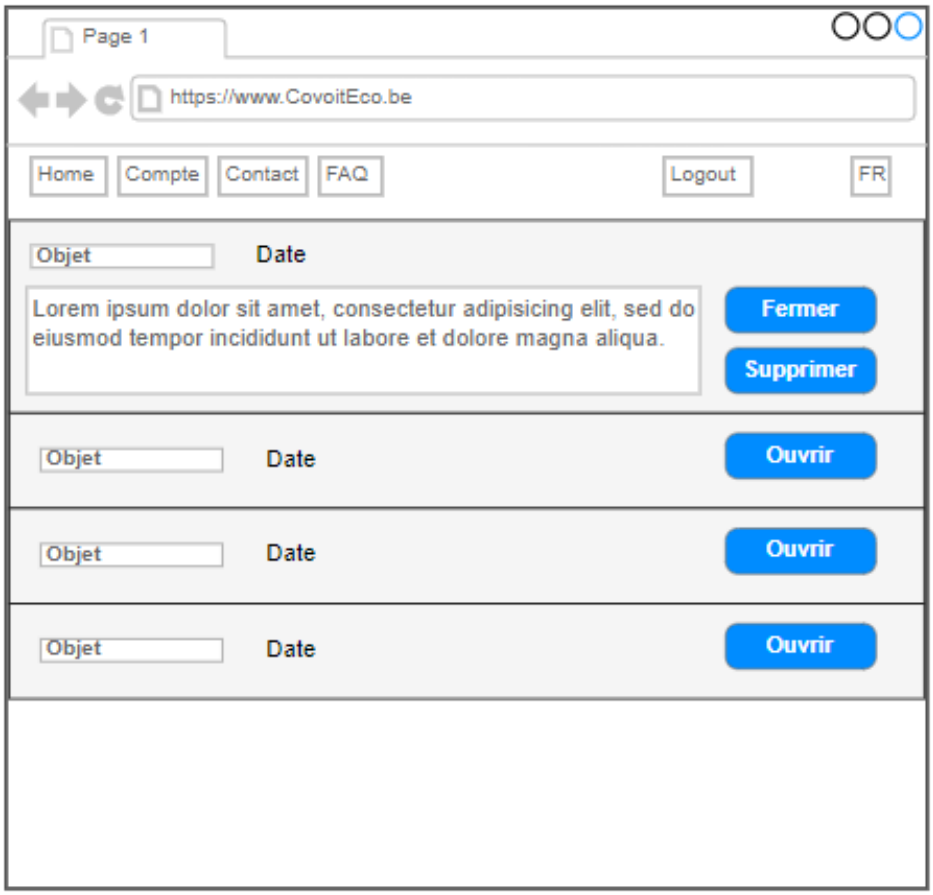
Enchaînement d'erreurs :

E1. Le message a été supprimé sur une seconde cession.
Démarre au point 5 du scénario nominal.
5. Le système invalide la modification et renvoie la raison du problème.

Post-conditions :

- Le message a été supprimé de la base de données (scénario nominal uniquement).

Écrans :



1.3.5 Contraintes

1.3.5.1 Règles de structure (RS)

Annonce :

Code	Description
RS-001	ANN_Id est obligatoire [clé primaire => auto-incrémentation]
RS-002	ANN_DatePublication est obligatoire
RS-003	ANN_Prix est obligatoire
RS-004	ANN_LocaliteDepart est obligatoire
RS-005	ANN_LocaliteArrive est obligatoire
RS-006	ANN_DateDepart est obligatoire
RS-007	ANN_DateArrive est obligatoire
RS-008	ANN_OptAutoroute est obligatoire (*false* par défaut)
RS-009	ANN_OptFumeur est obligatoire (*false* par défaut)
RS-010	ANN_OptAnimaux est obligatoire (*false* par défaut)
RS-011	ANN_VEH_Id est obligatoire
RS-012	ANN_STATANN_Id est obligatoire

RS-013	ANN_UTL_Id est obligatoire

Facture :

Code	Description
RS-014	FACT_Id est obligatoire [clé primaire => auto-incrémentation]
RS-015	FACT_DateCreation est obligatoire
RS-016	FACT_DatePayment n'est **PAS** obligatoire
RS-017	FACT_Resolus est obligatoire (*false* par défaut)
RS-018	FACT_RES_Id est obligatoire

Note :

Code	Description
RS-019	NOT_Id est obligatoire [clé primaire => auto-incrémentation]
RS-020	NOT_Cotation est obligatoire
RS-021	NOT_UTL_Id est obligatoire

Notification :

Code	Description
RS-022	NOTIF_Id est obligatoire [clé primaire => auto-incrémentation]
RS-023	NOTIF_IdAuteur n'est **PAS** obligatoire
RS-024	NOTIF_NOTIFLIBEL_Id est obligatoire
RS-025	NOTIF_UTL_Id est obligatoire

NotificationLibelle :

Code	Description
RS-026	NOTIFLIBEL_Id est obligatoire [clé primaire => auto-incrémentation]
RS-027	NOTIFLIBEL_Libelle est obligatoire

Reservation :

Code	Description
RS-028	RES_Id est obligatoire [clé primaire => auto-incrémentation]
RS-029	RES_DateReservation est obligatoire
RS-030	RES_ANN_Id est obligatoire
RS-031	RES_STATRES_Id est obligatoire
RS-032	RES_UTL_Id est obligatoire

StatutAnnonce :

Code	Description
RS-033	STATANN_Id est obligatoire [clé primaire => auto-incrémentation]
RS-034	STATANN_Libelle est obligatoire

StatutReservation :

Code	Description
RS-035	STATRES_Id est obligatoire [clé primaire => auto-incrémentation]
RS-036	STATRES_Libelle est obligatoire

Utilisateur :

Code	Description
RS-037	UTL_Id est obligatoire [clé primaire => auto-incrémentation]
RS-038	UTL_Nom est obligatoire
RS-039	UTL_Prenom est obligatoire
RS-040	UTL_Mail est obligatoire
RS-041	UTL_Actif est obligatoire (*true* par défaut)
RS-042	UTL_Image n'est **PAS** obligatoire
RS-043	UTL_Role est obligatoire
RS-044	UTL_IBAN est obligatoire
RS-045	UTL_AdresseFacturation est obligatoire

Vehicule :

Code	Description
RS-046	VEH_Id est obligatoire [clé primaire => auto-incrémentation]
RS-047	VEH_Immatriculation est obligatoire
RS-048	VEH_Couleur est obligatoire
RS-049	VEH_Courant est obligatoire (*false* par défaut)
RS-050	VEH_Disponible est obligatoire (*false* par défaut)
RS-051	VEH_NombrePlace est obligatoire
RS-052	VEH_NormeEuro est obligatoire
RS-053	VEH_Image est obligatoire
RS-054	VEH_UTL_Id est obligatoire
RS-055	VEH_MOD_Id est obligatoire

Modele : (Données de « base_auto_free » => E-BDD)

Code	Description
RS-056	MOD_Id est obligatoire [clé primaire => auto-incrémentation]
RS-057	MOD_Marque est obligatoire
RS-058	MOD_Modele est obligatoire

1.3.5.2 Règles de validation (RV)

Annonce :

Code	Description
RV-001	ANN_Prix = prix claculé automatiquement
RV-002	ANN_DateDepart > (date courante + 30min)
RV-003	(ANN_DateArrive - ANN_DateDepart) > 30 min
RV-004	ANN_STATANN_Id = 1 lors de la création => status "Publier"
RV-005	Chevauchement interdit (pour un même UTL_Id)
RV-006	Espacement entre annonces > 60 min (pour un même ANN_UTL_Id)

Facture :

Code	Description
RV-007	FACT_DateCreation = date courante
RV-008	FACT_DatePayment = date de paiement
RV-009	FACT_Resolus passe de *false* à *true* => quand paiement effectué

Note :

Code	Description
RV-011	0 <= NOT_Cotation <= 5

Notification :

Code	Description
RV-012	NOTIF_Type n'est **PAS** modifiable
RV-013	NOTIF_Type est un nombre de 1-10 correspondant à un message (voire **RMS**)

NotificationLibelle :

Code	Description
RV-014	NOTIFLIBEL_Libelle n'est **PAS** modifiable
RV-015	NOTIFLIBEL_Libelle correspond à un message (voire **RMS**)

Reservation :

Code	Description
RV-016	RES_DateReservation = date courante < (ANN_DateDepart - 15min)
RV-017	RES_STATRES_Id = 1 lors de la création => status "EnAttente"
RV-018	RES_UTL_Id != ANN_UTL_Id
RV-019	Chevauchement interdit (pour un même RES_UTL_Id)
RV-020	VEH_NombrePlace = nombre maximum de réservations par annonce
RV-021	Max 1 réservation/util. par annonce => (statuts "Annule" non comptabilisé)

StatutAnnonce :

Code	Description

RV-022	STATANN_Libelle => ("Publier", "EnCours" ou"Close")

StatutReservation :

Code	Description
RV-023	STATRES_Libelle => ("EnAttente", "Confirme", "EnOrdre" ou "Annule")

Utilisateur :

Code	Description
RV-024	UTL_IdAuth0 n'est **PAS** modifiable
RV-025	UTL_Actif passe de *true* à *false* => quand utilisateur supprimé
RV-026	ULT_Nom n'est **PAS** modifiable
RV-027	UTL_Prenom n'est **PAS** modifiable

Vehicule :

Code	Description
RV-028	VEH_Courant => 1 veh courant par utilisateur
RV-029	VEH_Disponible passe de *false* à *true* => quand veh autorisé
RV-030	VEH_Disponible passe de *false* à *true* => quand veh autorisé

Modele :

Code	Description
RV-031	MOD_Marque n'est **PAS** modifiable
RV-032	MOD_Modele n'est **PAS** modifiable

1.3.5.3 Règles de calculs (RC)

Annonce :

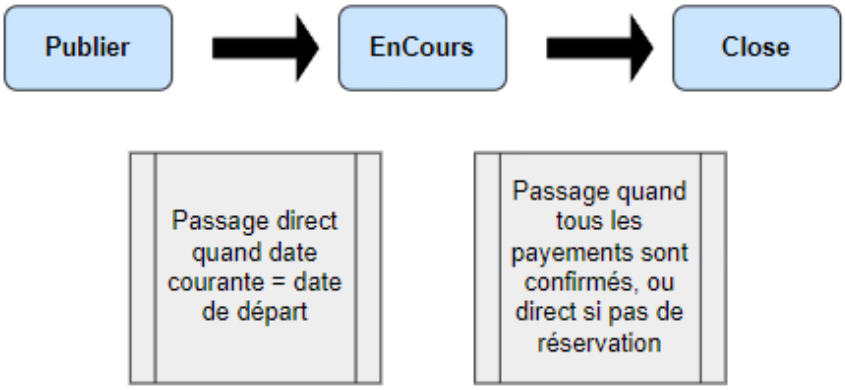
Code	Description
RC-001	ANN_Prix correspond à ANN_DateArrive - ANN_DateDepart en minutes, multiplié par 0.10 (10 cents) => le résultat est en euro (2 chiffres après la virgule).
RC-002	La commission est de 1 cent (Gestion externe => Banque)

Note :

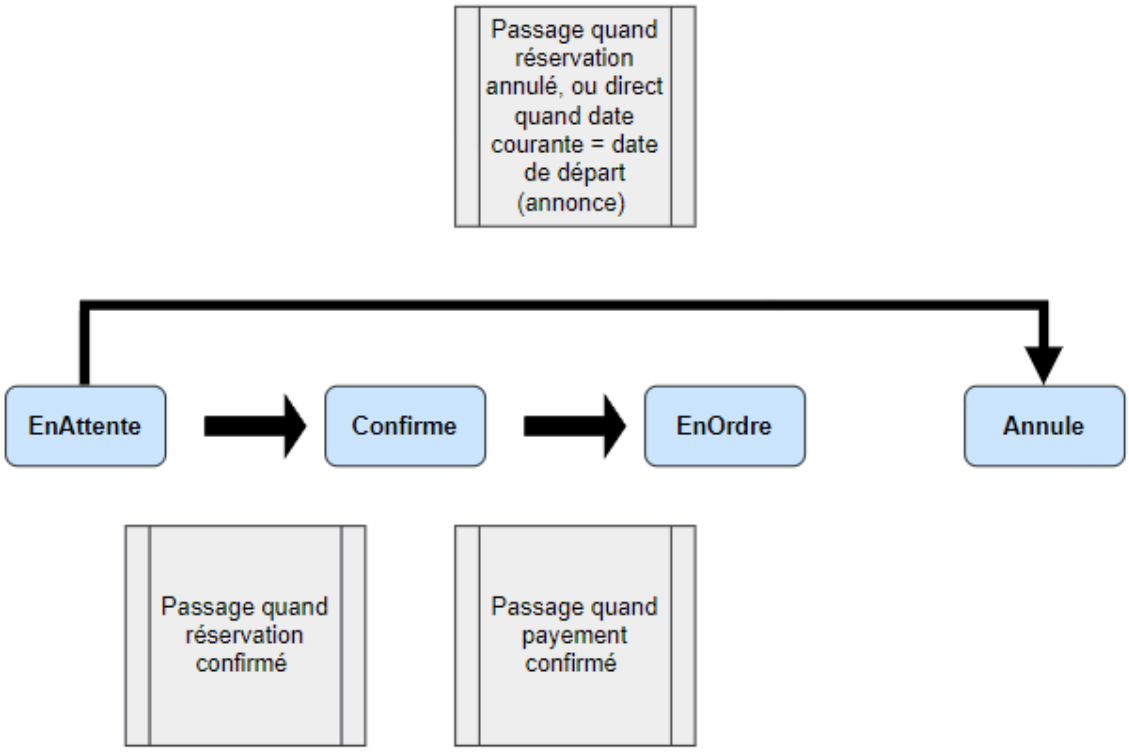
Code	Description
RC-003	NOT_Cotation correspond à la moyenne des notations

1.3.5.4 Règles de statuts

StatutAnnonce :



StatutReservation :



1.3.5.5 Règles de messagerie système (RMS)

Véhicule :

Code	Message
RMS-001	Le(s) véhicule(s) a/ont été confirmé par un administrateur.
RMS-002	De nouveaux véhicules sont en attentes de confirmation.

Utilisateur :

Code	Message
RMS-003	Vous faites l'objet d'un signalement par un ou plusieurs utilisateur(s).
RMS-004	Par suite d'un signalement vos droits ont été restreints.
RMS-005	Vos droits ont été rétablis.
RMS-006	Vous ne faites plus l'objet d'un signalement.

Réservation :

Code	Message
RMS-007	Votre demande de réservation est acceptée.
RMS-008	Vous avez des demandes de réservation en attentes.
RMS-009	Vous avez une ou plusieurs réservations(s) en attente de paiement.
RMS-0010	Vôtre paiement a été confirmé

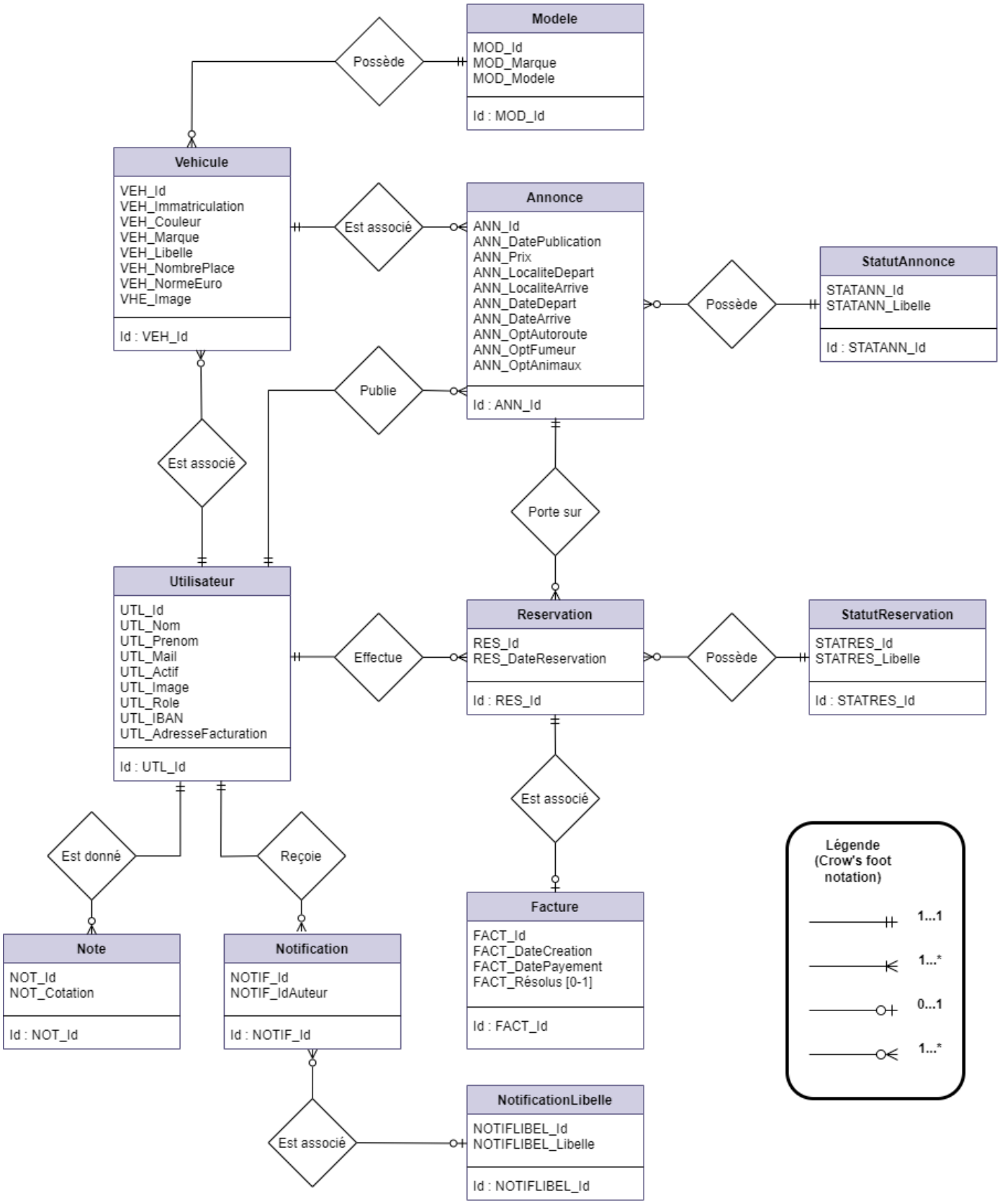
Table des événements :

Code (basée)	Événement A (création de la notification)	Événement B (fin de la notification)	Destinataire (User/Admin)	Auteur requis (Oui/Non)	Type
RMS-001	Véhicule confirmé par un admin	Bouton « supprimer » utilisé	User	Non	1
RMS-002	Véhicule non disponible	Plus de véhicule non disponible	Admin	Non	2
RMS-003	Signalement d'un utilisateur effectué	Signalement annuler par un admin	User	Oui	3
RMS-004	Rôle => bloquedUser	Bouton « supprimer » utilisé	User	Non	4
RMS-005	Rôle => bloquedUser à defaultUser	Bouton « supprimer » utilisé	User	Non	5
RMS-006	Signalement supprimé par un admin	Bouton « supprimer » utilisé	User	Non	6
RMS-007	Demande de réservation acceptée	Bouton « supprimer » utilisé	User	Non	7

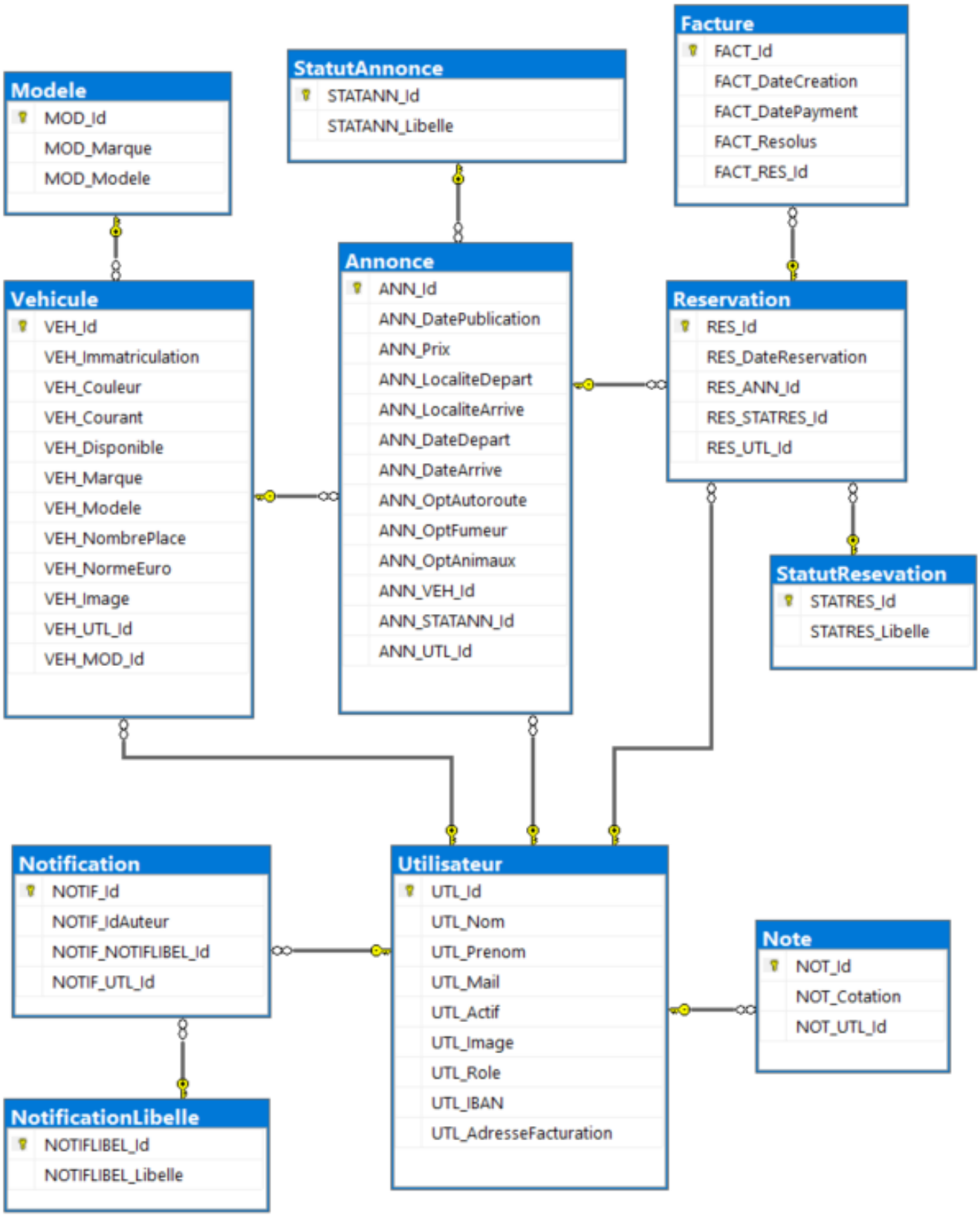
RMS-008	Demande de réservation en statut « EnAttente »	Plus de demande de réservation en statut « EnAttente »	User	Non	8
RMS-009	Date d'arrivée pour une annonce échue + réservation en statut « Confirme »	Plus de réservation en statut « Confirme » et une date d'arrivée échue	User	Non	9
RMS-010	Paiement confirmé par un annonceur	Bouton « supprimer » utilisé	User	Non	10

1.3.6 Base de données

1.3.6.1 Schéma E-A



1.3.6.2 Schéma DB



1.3.7 Conclusion

Dans ce chapitre j'ai abordé en détails les différentes fonctionnalités que propose CovoitEco. J'ai également défini les différentes contraintes (logique métier) qui régissent les différentes entités ainsi que leurs interactions. Enfin j'ai présenté en détaille la structure de la base de données exploitée par l'application.

2 Développement

2.1 Introduction

Ce chapitre contient les technologies employées pour le développement et l'architecture de l'application. Les différentes *commands* et *queries* au sein du *backend* y sont détaillés. Ce chapitre contient également une brève description pour les outils de développement principalement employés ainsi que les API. Le chapitre se termine par des propositions de développement futur.

2.2 Technologie & Architecture

2.2.1 Templates

- ASP.NET Core Web API (.NET 6) => Backend
- Blazor WebAssmeblyApp (.NET 6) => Frontend
- .NET MAUI App (.NET 6) => Frontend
- Class Library => librairies

2.2.2 Langages

- C#
- XAML
- CSS (+ Framework Bootstrap)
- HTML (+ Framework Bootstrap)
- JSON
- SQL

2.2.3 Gestionnaire de base de données

- Microsoft SQL Server (+ MySQL)

2.2.4 Architecture global (CQRS)

2.2.4.1 Description

L'architecture du projet est de type CQRS. Dans une architecture CQRS les requêtes (*queries*) sont séparer des commandes (*commands*). C'est-à-dire que l'accès aux données est séparé en lecture (requêtes) et écriture (commandes). L'architecture CQRS est également organisée en différents couches.

Frontend (Web, Mobile) :

Cette couche correspond à la partie du code coté client (navigateur). Elle se présente à l'utilisateur sous la forme d'un site web ou application mobile interactive. Elle communique avec la partie server via des appels HTTP.

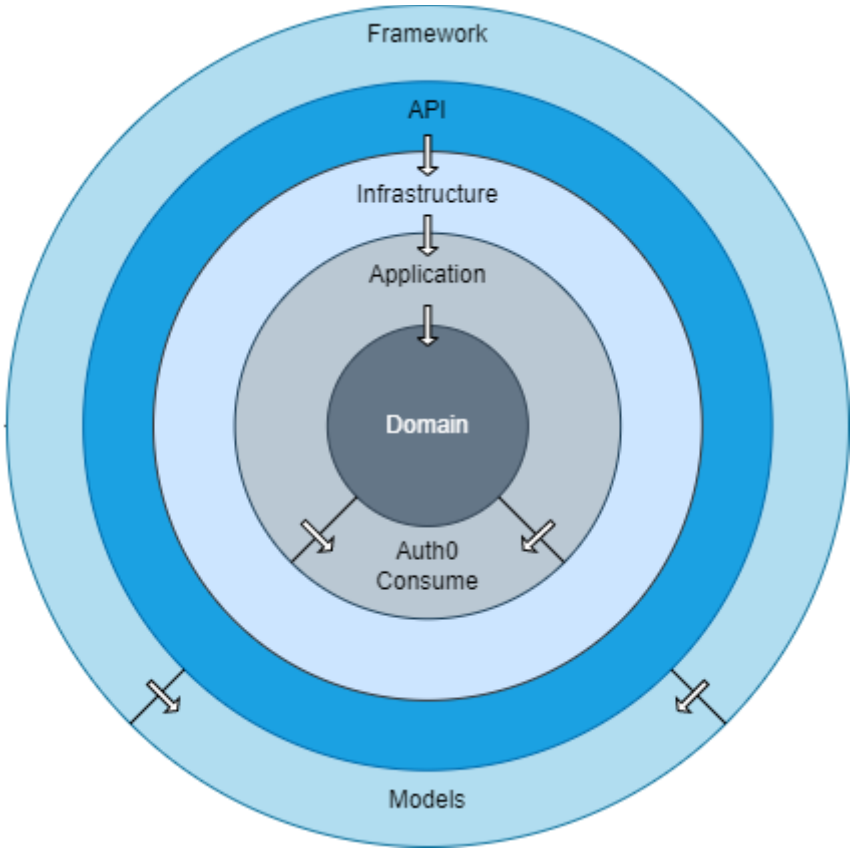
Backend :

Cette couche correspond à la partie du code coté serveur. Elle est invisible pour l'utilisateur. Cette partie interagit avec la base de données et contient la logique métier. La communication entre le backend la DB est réalisé à l'aide d'Entity FrameWork (couche d'accès aux données de la DB).

Base de données :

C'est dans cette partie que sont stocké les données de CovoitEco. Cette couche n'est également pas visible pour l'utilisateur.

2.2.4.2 Schéma



Domain :

Contient les entités qui sont manipulées par l'API.

Application + Auth0 consume :

Contient les interfaces, validators, commands, queries, models, exceptions, mappings et DTO's.

Infrastructure :

Contient la partie defaultContext, migration et logger.

API :

Contient les endpoints de l'API (contrôleurs), les middlewares et handler.

Framework + models :
Constitue l'interface user (page, components), service et handler.

2.3 Appels API

2.3.1 Description

Les divers services fournis par le backend ou API externe, sont accessible via des requêtes de type HTTP pour les clients. Ces services requièrent la permission adéquate afin d'être exploité. Ces requêtes nécessitent également parfois un ou plusieurs paramètres.

2.3.2.1 Annonce

Nom	Paramètre(s)	Permission	Description	Méthode
GetAllAnnonceProfile	N/A	read:annonce	Renvoie la liste des annonces d'un utilisateur	GET
GetAnnonceProfile	ANN_Id	read:annonce	Renvoie une annonce	GET
GetAnnonceReservation	ANN_Id	read:annonce	Renvoie toutes les reservations pour une annonce	GET
CreateAnnonce	CreateAnnonceCommand (objet)	create:annonce	Crée une annonce	POST
GetAnnonceRecherche	departureDate, departureCity,arrivalCity	read:annonce	Renvoie une liste d'annonces correspondante	GET
UpdateStatutAnnonce	ANN_Id	update:annonce	Update le statut d'une annonce	PUT
GetDistanceTime	Language, mode, region, key, originis, destinations	Read:annonce	Renvoie la distance et le temps entre origins et destinations (**Distance Matrix**)	GET
GetPrediction	Language, region, types, key, input	Read:annonce	Rouvoie une liste de prédiction (**Place Autocomplete**)	GET

2.3.2.2 Facture

Nom	Paramètre(s)	Permission	Description	Méthode
GetIdFactureReservation	RES_Id	read:facture	Renvoie le FACT_Id correspondant	GET
CreateFacture	RES_Id	create:facture	Crée une facture pour une reservation	POST
UpdateFacturePayment	FACT_Id	update:facture	Ajoute une date de paiement	PUT

2.3.2.3 Réservation

Nom	Paramètre(s)	Permission	Description	Méthode
GetAllReservationProfile	ANN_Id	read:reservation	Renvoie la liste des réservations pour une annonce	GET
GetAllReservationUserProfile	N/A	read:reservation	Renvoie la liste des réservations pour un utilisateur	GET
GetReservationUserProfile	RES_Id	read:reservation	Renvoie la réservation correspondante	GET
GetIdReservationUserProfile	ANN_Id, UTL_Id	read:reservation	Renvoie le RES_Id pour l'annonce et l'utilisateur correspondants	GET
CreateReservation	CreateReservationCommand (objet)	create:reservation	Crée une réservation	POST
UpdateAccepterReservation	RES_Id	update:reservation	Update le statut d'une réservation	PUT
UpdateConfirmePayment	RES_Id	update:reservation	Confirme la résolution du paiement	PUT
UpdateStatutReservation	RES_Id	update:reservation	Update le statut d'une réservation	PUT
DeleteReservation	RES_Id	delete:reservation	Update le statut d'une réservation	PUT

2.3.2.4 Utilisateur

Nom	Paramètre(s)	Permission	Description	Méthode
CreateUser	CreateUserCommand (objet)	create:profile	Crée un compte utilisateur (**CoiviteEco/Auth0**)	POST
GetUserProfile		read:profile	Renvoie le profile utilisateur	GET
GetIDUserProfile	UTL_Mail	read:profile	Renvoie l'UTL_Id correspondant	GET
DeleteUser	N/A	delete:profile	Update le statut d'un utilisateur	PUT
GetAllUser	N/A	read :user	Renvoie la liste de tous les utilisateurs	GET
UpdateUser	UserUpdate (objet), UTL_IdAuth0	update:profile	Update un profile utilisateur (**CoiviteEco/Auth0**)	PATCH
GetUserInfo	accessToken	read:profile	Renvoie les informations de l'utilisateur (**Auth0**)	GET

2.3.2.5 Véhicule

Nom	Paramètre(s)	Permission	Description	Méthode
GetVehiculeProfile	N/A	read:vehicule	Renvoie le profil du véhicule courant d'un utilisateur	GET
GetVehicule	ANN_Id	read:vehicule	Renvoie le profil du véhicule pour l'annonce correspondante	GET
GetAllVehiculeProfile	N/A	read:vehicule	Renvoie la liste de véhicule d'un utilisateur	GET
CreateVehiculeProfile	CreateVehiculeProfileCommand (objet)	create:vehicule	Crée un véhicule	POST
UpdateVehiculeProfile	UpdateVehiculeProfileCommand (objet)	update:vehicule	Update le statut et la disponibilité d'un véhicule	PUT
UpdateAccepterVehicule	VEH_Id, UTL_Id	update:user	Update la disponibilité d'un véhicule	PUT
GetNewVehiculeUser	UTL_Id	read :user	Renvoie la liste de tous véhicule utilisateur non confirmé	GET

2.3.2.6 Note

Nom	Paramètre(s)	Permission	Description	Méthode
CreateNote	UTL_Id	create:note	Crée une note pour un utilisateur	POST
UpdateAverageNote	NOT_Id	update:note	Update la cotation	PUT
GetNote	UTL_Id	read:note	Renvoie la note d'utilisateur	GET

2.3.2.7 Notification

Nom	Paramètre(s)	Permission	Description	Méthode
CreateNotification	UTL_Id, NOTIF_Type	create:notification	Crée une notification pour un utilisateur	POST
DeleteNotification	NOTIF_Id	delete:notification	Supprime définitivement une notification	DELETE
UpdateNotification	NOTIF_Id	update:notification	Update la notification	PUT
GetNotification	N/A	read:notification	Renvoie la liste des notifications d'un utilisateur	GET
GetNumbReproting	UTL_Id	read :user	Renvoie le nombre de signalement pour un utilisateur	GET

GetAllInfoReporting	UTL_Id	Read :user	Renvoie la liste des nom, prénom et adresse des auteurs de signalements	GET

2.3.2.8 Rôle

Nom	Paramètre(s)	Permission	Description	Méthode
AssignRole	UserRole (objet), idRole	update:user	Update (ajoute) le rôle d'un (ou plusieurs) utilisateur (**Auth0**)	POST
RemoveRole	RemoveRoleUser, UTL_IdAuth0	update:user	Update (retire) un (ou plusieurs) rôle d'un utilisateur (**Auth0**)	DELETE
GetRoleUser	UTL_Id	read:user	Renvoie le rôle d'un utilisateur	GET

2.7 Outils de développement

2.7.1 Swagger

Swagger un outil professionnel gratuit d'aide au développement d'API's. Swagger permet de visualiser et de tester facilement des API grâce à des formulaire pré-complété. Swagger génère automatiquement de la documentation pour une API.

Dans le cadre de ce projet Swagger UI a été principalement utilisé pour :

- Vérifier si les permissions sont actives niveau backend
- Visualiser rapidement l'ensemble des services du backend
- Effectuer des requêtes HTTP et visualiser les réponses
- Tester la gestion des erreurs

2.7.2 Postman

Postman est une API d'aide au développement d'API's. Postman permet de tester des requêtes groupées, de faciliter la construction de documentation, d'établir un historique des résultats…

Dans le cadre de ce projet Postman a été principalement utilisé pour :

- Tester la validité d'un token
- Tester la gestion des permissions
- Enregistrer des requêtes réutilisables
- Tester des requêtes pour les API's externes (Auth0, calendrier et paiement)

2.7.3 GitHub

GitHub est une plateforme de stockage en ligne. Elle permet d'organiser et structurer les différentes versions d'un projet et favorise le suivis et le partage du code entres développeur. GitHub facilite également le travail collaboratif et la gestion des projets.

Dans le cadre de ce projet GitHub a été principalement utilisé pour :

- Stocker les différentes versions du projet
- Offrir un suivi et faciliter le partage du projet
- Créer un backup online

2.7.4 Visual Studio 2022

Visual studio est environnement de développement intégré (IDE). Il comprend les outils nécessaires à l'écriture du code, sa compilation et son éventuel débogage. Visual studio supporte de nombreux langages en plus de C# (C++, JavaScript, TypeScript, Python, etc). Il comprend également de nombreux type de templates pour diverses plateformes.

Dans le cadre de ce projet Visual Studio a principalement été utilisé pour :

- Éditer du code
- Déboguer
- Tester le code
- Faciliter la gestion des versions sur GitHub

2.7.5 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) est un IDE dédié à la gestion de l'infrastructure SQL. Il comprend l'outil nécessaire à la gestion, configuration et administration des composants de SQL Server, Azure SQL Databases et Azure Synapse Analytics.

Dans le cadre de ce projet SSMS a principalement été utilisé pour :

- Visualiser la base de données CovoitEco
- Créer la base de données CovoitEco
- Interroger la base de données CovoitEco (requêtes)
- Gérer la sécurité d'accès aux instances de la base données CovoitEco
- Éditer des scripts

2.8 API's

2.8.1 Auth0

2.8.1.1 Description

Auth0 est une plate-forme prenant en charge l'authentification et l'autorisation (permissions/rôles) pour une application, un intranet ou un site.

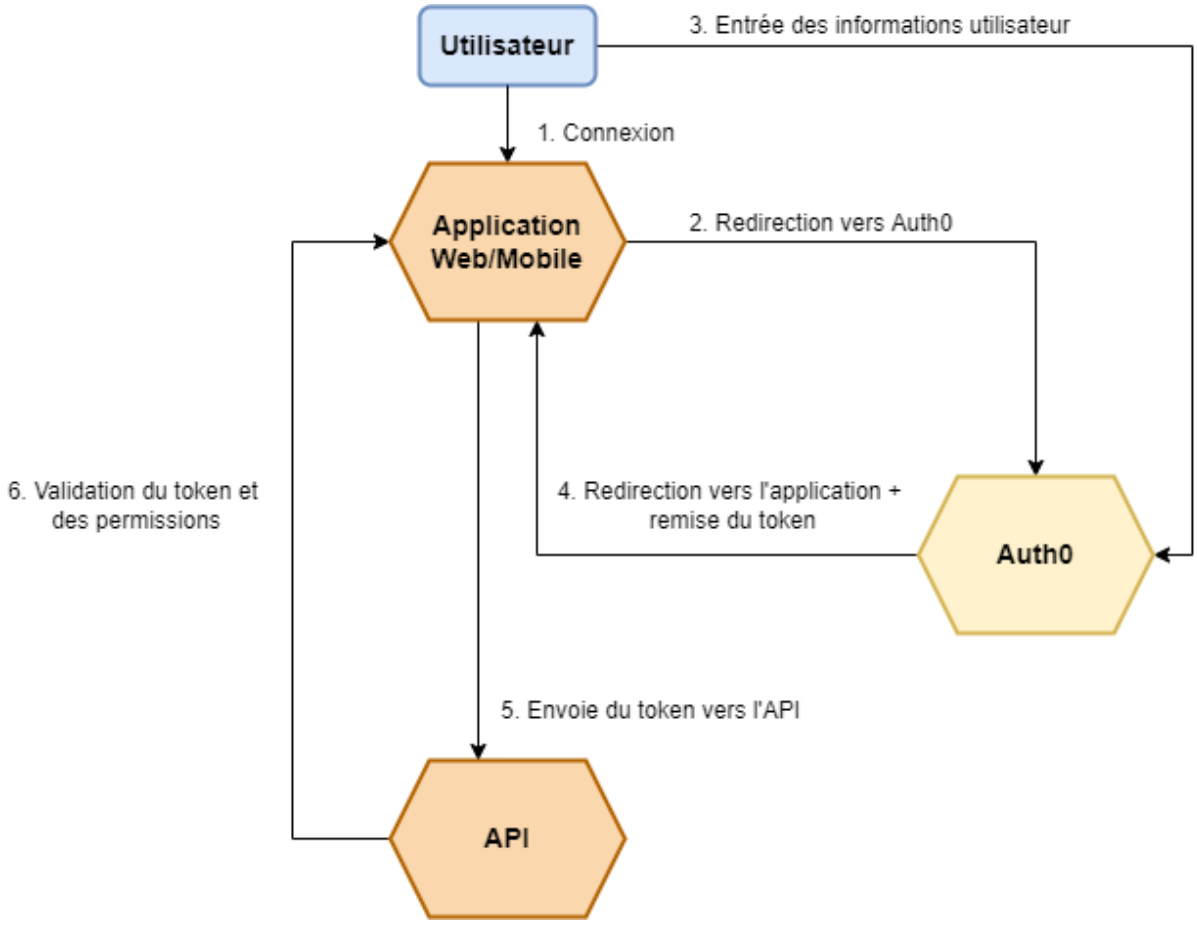
Dans le cadre de ce projet Auth0 a principalement été utilisé pour :

- Inclure une interface d'authentification personnalisé
- Gérer les permissions et les rôles pour CovoitEco
- Intégrer des fonctionnalités de gestion des rôles et des utilisateurs (Auth0 Management)
- Sécuriser les routes d'accès

2.8.1.2 Fonctionnement

Lorsqu'un utilisateur souhaite se connecter sur CovoitEco il est redirigé vers l'interface de connexion Auth0. Une fois les informations de connexion (mail + mot de passe) reconnus, Auth0 redirige l'utilisateur vers la page d'accueil de CovoitEco. A travers cette connexion est délivrée un token (au standard JSON Web Token) comprenant entre autres les permissions et données utilisateur.

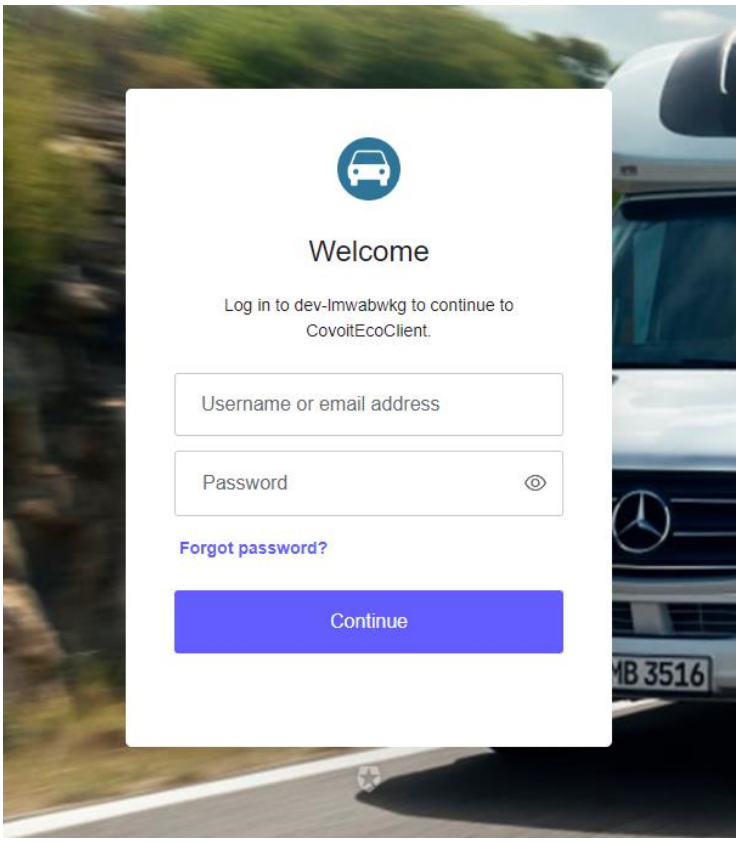
Schéma :



2.8.1.3 Portail d'authentification

L'utilisateur renseigne son identifiant (mail ou prénom) ainsi que son mot de passe et clique sur continuer pour transmettre ces informations de connexion. En cas d'oublie du mot de passe il est possible pour l'utilisateur de le modifier.

Capture :



2.8.1.4 Tokens

Divers tokens sont employés au travers du processus de fonctionnement de Auth0. Ils permettent d'échanger des informations entre entités (information utilisateur, permissions…).

ID tokens :

Les tokens JWT (JSON Web Token) sont des tokens suivent les standards RFC 7519 (structure, contenus...). Ils sont employés pour **l'authentification** d'un utilisateur. Ils contiennent divers informations utilisateurs (name, email, picture etc…).

Access tokens :

Les access tokens sont employé pour **l'autorisation** d'un utilisateur. Il contient des informations relatives aux droits, permissions, l'ID utilisateur et la validité du token. Ce sont également des tokens JWT.

Ils existent de plus des jetons dit spécialisés :

- Refresh tokens
- IDP access tokens
- Auht0 Management API access tokens

2.8.2 Google Places API

2.8.2.1 Présentation

L'API Places permet d'obtenir des données de localisations via de requêtes de type http. Ces données peuvent être des descriptions de lieux, des coordonnées de géolocalisations, des photos, des avis, des horaires, des niveaux de prix, etc. Les paramètres incomplets ou ambigus sont prises en charge par l'API qui propose également des fonctionnalités de saisie-automatique ou autocomplété.

2.8.2.1 Place Autocomplete

Ce service renvoie en réponse à une requête de type http, une prédiction. Dans CovoitEco ce sont uniquement des localités qui sont proposées à l'utilisateur lors de la saisie d'un caractère.

Les seuls paramètres obligatoires sont l'***input*** utilisateur et la ***key***. Des paramètres supplémentaires peuvent également être ajouté.

Pour CovoitEco les paramètres employés sont :

- ***language*** : permet de sélectionner dans quelle langue sont exprimé les résultats
- ***region*** : spécifie le *country code* du top-level domain
- ***types*** : spécifie la nature de la prédication (restaurant, pharmacie, localité,…)

Format de l'URL (requête) :

```
https://maps.googleapis.com/maps/api/place/autocomplete/output?parameters
```

Une réponse se présente sous la forme d'une liste de prédictions comprenant diverses informations ainsi que le mot suggérer dans « description ».

Exemple de réponse (JSON) **:**


```
"predictions": [
    {
        "description": "Bruxelles",
        "matched_substrings": [
            {
                "length": 1,
                "offset": 0
            }
        ],
        "place_id": "ChIJZ2jHc-2kw0cRpwJzeGY6i8E",
        "reference": "ChIJZ2jHc-2kw0cRpwJzeGY6i8E",
        "structured_formatting": {
            "main_text": "Bruxelles",
            "main_text_matched_substrings": [
                {
                    "length": 1,
                    "offset": 0
                }
            ]
        },
```

2.8.3 Google API Distance Matrix

Distance Matrix permet via une requête http d'obtenir la distance et la durée entres deux (ou plus) localisations.

Dans la requête il faut obligatoirement une ***origins***, une ***destinations*** et une ***key***. Des paramètres supplémentaires peuvent également être ajoutés, en plus des paramètres par défaut (***traffic_model*** et ***mode***).

Pour CovoitEco les paramètres employés sont :

- ***language*** : permet de sélectionner dans quelle langue sont exprimé les résultats
- ***region*** : spécifie le *country code* du top-level domain
- ***mode*** : spécifie le type de locomotion (driving [default], walking, bicycling,...)
- ***traffic_model*** : spécifie si prédiction (temps) est optimiste ou non (optimistic > best_guess [default] > pessimistic) => non utilisé dans la requêtes

Format de l'URL (requête) :

```
https://maps.googleapis.com/maps/api/distancematrix/outputFormat?parameters
```

Une réponse se présente sous la forme d'une adresse de destination et d'origines autocomplété ainsi que la distance et la durée entre deux points.

Exemple de réponse (JSON) **:**


```
{
   "destination_addresses": [
      "Strombeeksesteenweg, 1800 Vilvoorde, Belgique"
   ],
   "origin_addresses": [
      "Sint-Annalaan 275, 1853 Grimbergen, Belgique"
   ],
   "rows": [
      {
         "elements": [
            {
               "distance": {
                  "text": "1,1 km",
                  "value": 1088
               },
               "duration": {
                  "text": "2 minutes",
                  "value": 137
               },
               "status": "OK"
            }
         ]
      }
   ],
   "status": "OK"
}
```

2.9 Syncfusion (Trial version)

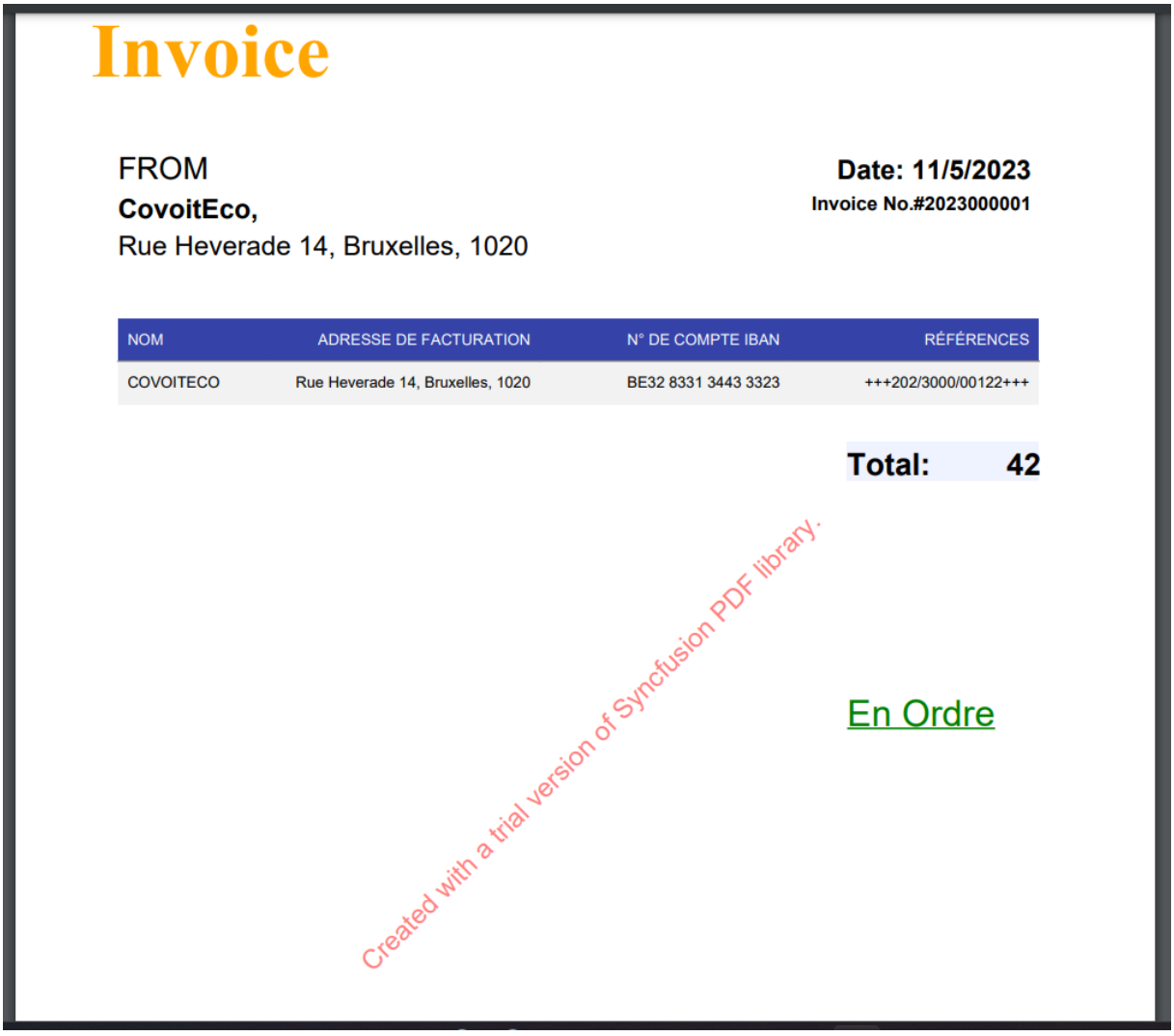
2.9.1 Présentation

Syncfusion est un logiciel de support au développement Web, Desktop et application Mobile. Il offre en une multitude de composants et frameworks pour l'environnement .NET, Flutter, JavaScript, Angular, Vue, et React. Il propose également des outils de test et d'entraiment ainsi que la documentation et les tutoriels nécessaire. De plus, il inclut diverses fonctionnalités permettant la création ou la modification de documents de type Excel, Word, PDF et PowerPoint.

2.9.2 Gestion des paiements

Les factures sous format pdf sont générés grâce aux fonctionnalités apportées par Syncfusion. Ces factures contiennent les informations de paiement nécessaire à l'établissement d'un virement classique sur un compte de l'entreprise. CovoitEco sous traite en partie la gestion des paiements et du prélèvement des commissions, chez une banque.

Exemple de facture :



Invoice

FROM
CovoitEco,
Rue Heverade 14, Bruxelles, 1020

Date: 11/5/2023
Invoice No.#2023000001

NOM	ADRESSE DE FACTURATION	N° DE COMPTE IBAN	RÉFÉRENCES
COVOITECO	Rue Heverade 14, Bruxelles, 1020	BE32 8331 3443 3323	+++202/3000/00122+++

Total: 42

Created with a trial version of Syncfusion PDF library.

En Ordre

2.9.3 Téléchargement du ficher dans le navigateur

Le téléchargement est exécuté grâce à l'ajout de la classe **FileUtil.**

Code :

```
0 references
public static class FileUtil
{
    1 reference
    public static ValueTask<object> SaveAs(this IJSRuntime js, string filename, byte[] data)
        => js.InvokeAsync<object>(
            "saveAsFile",
            filename,
            Convert.ToBase64String(data));
}
```

La fonction « **saveAsFile** » est définit au niveau de index.html dans dossier wwwroot.

Code :

```
<script type="text/javascript">
    function saveAsFile(filename, bytesBase64) {
        if (navigator.msSaveBlob) {
            //Download document in Edge browser
            var data = window.atob(bytesBase64);
            var bytes = new Uint8Array(data.length);
            for (var i = 0; i < data.length; i++) {
                bytes[i] = data.charCodeAt(i);
            }
            var blob = new Blob([bytes.buffer], { type: "application/octet-stream" });
            navigator.msSaveBlob(blob, filename);
        }
        else {
            var link = document.createElement('a');
            link.download = filename;
            link.href = "data:application/octet-stream;base64," + bytesBase64;
            document.body.appendChild(link); // Needed for Firefox
            link.click();
            document.body.removeChild(link);
        }
    }
</script>
```

2.10 Développement futur

- Il serrait intéressant que les utilisateurs puissent s'envoyer des messages. Ces messages serraient également envoyés directement sur la boite mail des utilisateurs.
- Pouvoir affiner la filtration ou utiliser une carte lors d'une recherche d'annonce(s) serrait une amélioration intéressante pour les utilisateurs.
- Actuellement l'application n'est utilisable que pour la Belgique, il faudrait activer le blocage d'accès par emplacement afin de limiter son utilisation au territoire belge.
- Il serrait souhaitable d'inclure en plus du virement classique, plusieurs solutions de paiement exécutable depuis l'application.
- Il faudrait inclure des rubriques concernant la politique de gestion des données, les copyrights, et les informations légales.

2.11 Conclusion

Dans ce chapitre j'ai abordé les différentes technologies employées dans le cadre du projet ains que son architecture globale. Les différents services du backend ont été également détaillés. Les différents outils ayant été utilisés au cours du développement et leurs apports, ont été brièvement présenté. Les API externe et leurs rôles ont été également décris. Enfin j'ai présenté une liste de réflexion concernant les possibilités de développement futur.

3 Conclusion

Par l'intermédiaire de se travail j'ai pu acquérir de nouvelles compétences mais aussi affiné celles que je possédais. C'est durant la réalisation de ce projet que j'ai pour la première fois réalisé le développement d'une application mobile. Rien que par l'intermédiaire de cette réalisation ce sont des nouveaux concept, langage et technologie qui ont été explorer. Mais aussi de façon plus générale c'est dans la réalisation de CoivoitEco que j'ai affinée ma méthodologie du travail, ma compréhension de concepts pourtant familier et l'exploitation de ressource externes comme les librairies ou les API's.

Concevoir et développer un projet plus grand et plus complexe que ce que j'avais réalisé auparavant, a représenté un défi. Devoir dans un premier fournir une analyse étendue a demandé un effort dans ma capacité à anticiper les ressources dont j'aurai besoin et les outils que je devrai utiliser. J'ai dû égalent eus plus de difficulté à définir la logique métier et les fonctionnalités d'un projet plus complexe qu'a mon habitude. Pour le développement c'est surtout l'utilisation de Framework pour lequel je n'avais pas d'expérience ou pour lequel j'ai peux d'expérience, qui à concentrer mes efforts. J'ai dû également dédier beaucoup de temps la lecture de documentation et à l'apprentissage en générale de nouveaux concepts, langages et composants.

Les limites essentiellement liées au temps dont je disposais pour l'ensemble du projet, a dicté en partie mon choix concernant les technologies employés et les fonctionnalités.

Les fonctionnalités les plus attendues sur une simple application de covoiturage sont, je pense, en bonne partie reprisent pour CovoitEco. Il y a encore cependant de nombreuse amélioration possible ou à effectuer.

Elles concernent principalement :

- L'ajout de solutions de payement et la gestion des paiements en générale, afin de correspondre au standard actuel pour une application de ce type.
- L'amélioration de la recherche d'annonce qui pour l'instant reste limité et qui pourrait inclure une carte, filtres ou autre aide à la recherche.
- L'ajout d'une messagerie utilisateur qui permettrait d'internaliser les communications, plus proches des standards actuels.
- L'ajout d'un politique de gestion des données, les copyrights, et les informations légales ou toutes autres rubriques obligatoire pour une application de covoiturage.

De manière général CovoitEco propose une bonne base pour la création d'une version plus professionnel et/ou plus complète d'une application d'une application de covoiturage web et mobile.

4 Bibliographie

- « Modèle CQRS », *Microsoft*, https://learn.microsoft.com/fr-fr/azure/architecture/patterns/cqrs (consulté le 25.04.2023).

- « Swagger UI », *Swagger.io*, https://swagger.io/tools/swagger-ui/ (consulté le 25.04.2023).

- « What is Postman? », *Postman*, https://www.postman.com/product/what-is-postman/ (consulté le 25.04.2023).

- ANAND Megha, MILLER Paula, RICHARDS Dania, et col. 2023. « Qu'est-ce que Visual Studio ? », *Microsoft*, 05.05.2023, https://learn.microsoft.com/fr-fr/visualstudio/get-started/visual-studio-ide?view=vs-2022 (consulté le 07.05.2023).

- GAHNAYEM Mark, WEST Randolph, ROTH Jason, et col. 2023. « What is SQL Server Management Studio (SSMS)? », *Microsoft*, 31.03.2023, https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16 (consulté le 02.05.2023).

- COTTREZ Gaëten. « Réaliser un système d'authentification avec Auth0 », *apprendre-la-programmation*, https://apprendre-la-programmation.net/authentification-auth0/ (consulté le 05.05.2023).

- « Tokens », *Auth0*, https://auth0.com/docs/secure/tokens (consulté le 06.05.2023).

- « Why you need Essential Studio », *Syncfusion*, https://www.syncfusion.com/ (consulté le 02.05.2023).

- « Présentation », *Google*, https://developers.google.com/maps/documentation/places/web-service/overview?hl=fr (consulté le 10.05.2023).

- « Présentation », *Google*, https://developers.google.com/maps/documentation/places/web-service/overview?hl=fr (consulté le 10.05.2023).

- « Présentation de l'API Distance Matrix », *Google*, https://developers.google.com/maps/documentation/distance-matrix/overview?hl=fr (consulté le 10.05.2023).

– « Base de données de marques et modèles de voitures », *E-BDD*, https://www.e-bdd.com/base-de-donnees-voitures.html
(consulté le 08.05.2023).

Tables des Annexes

5 Annexes