



HONORIS UNITED UNIVERSITIES

2021 / 2022

SPÉCIALITÉ :

Systèmes Informatiques et Mobiles

Application Mobile : SPN – Cars

Réalisé par : Alaa Abdelbaki

Encadré par : Swiss Premium Negoce

Encadrant ESPRIT : Mme. Nouha Samet

Encadrant Entreprise : Mr. Ghaith Ben Abdessalem

SPN

Swiss Premium Negoce
GENÈVE

Remerciements

Au terme de ce projet, je tiens à adresser mes remerciements les plus sincères et à exprimer ma gratitude envers mon encadrant pédagogique Madame Nouha Samet dont l'assistance et la disponibilité étaient présentes tout au long du stage.

Je la remercie aussi pour ses directives et les conseils qu'elle m'a prodiguée pour atteindre les objectifs du stage dans les délais convenus.

Je dois aussi une grande partie de mon travail à Monsieur Ghaith Ben Abdessalem. Je le remercie pour sa disponibilité, ses remarques constructives qui m'ont aidé à surmonter beaucoup de difficultés et à améliorer les fonctionnalités de l'application ainsi que ses qualités humaines d'écoute et de compréhension tout au long de ce travail.

J'exprime également ma gratitude envers tous les membres de la société Swiss Premium Negoce qui m'ont apporté leur soutien et leur savoir faire. J'ai découvert dans ce service une équipe jeune, dynamique et conviviale qui est devenue, pour moi, une famille. Je remercie également la directrice de département IT de la société Madame Jihene Ben Abderrazek pour sa disponibilité et ses conseils qui ont été très bénéfiques tout au long de ce stage.

Je tiens également à remercier tous les enseignants qui ont participé à mon évolution scientifique durant ces quatre années que j'ai passé au sein de ESPRIT. Je remercie finalement les membres du jury en espérant qu'ils apprécieront ce rapport.

Alaa Abdelbaki

Table des matières

Introduction générale	1
1 Cadre du projet	3
1.1 Présentation de l'organisme d'accueil	4
1.1.1 Présentation générale	4
1.1.2 Activités	4
1.2 Présentation du projet	5
1.2.1 Présentation générale	5
1.2.2 Problématique	5
1.2.3 Étude de l'existant	5
1.2.4 Critique de l'existant	7
1.2.5 Solution Proposée	8
1.3 Méthodologie de travail	9
2 Conception du projet	13
2.1 Spécification des besoins	14
2.1.1 Besoins fonctionnels	14
2.1.2 Besoins non fonctionnels	15
2.2 Diagrammes UML	15
2.2.1 Diagramme de cas d'utilisation	15
2.2.2 Créer réservation	17

2.2.3	Gérer réservation	17
2.2.4	Diagramme de classes	19
2.2.5	Diagrammes de séquences	20
2.3	UI/UX Design	28
2.4	Technologies et outils utilisés	31
2.4.1	Flutter	31
2.4.2	Express JS	32
2.4.3	JSON Web Token (JWT)	32
2.4.4	MongoDB	33
2.4.5	Firebase	33
2.4.6	Visual Studio Code	34
2.4.7	Postman	35
2.4.8	Adobe Xd	35
2.4.9	Git	36
2.4.10	Swagger	36
3	Réalisation de l'application	39
3.1	Création de compte	40
3.1.1	Création de compte avec email et mot de passe	40
3.1.2	Création de compte avec Google ou Apple	41
3.2	Authentification	41
3.3	Page d'accueil	43
3.4	Gestion de profil	44
3.5	Demander un service	44
3.6	Affichage des voitures disponibles	46
3.7	Signature numérique de contrat de location	46
3.8	Paiement	49
3.9	Localiser une voiture	49
3.10	Messagerie instantanée	50
3.11	Documentation des API	51
Conclusion		55

Table des figures

1.1	Logo Swiss Premium Negoce	4
1.2	Logo Sixt	5
1.3	Logo Blacklane	6
1.4	Logo Uber	6
1.5	Sélection de voiture avec Uber	8
1.6	Sélection de voiture avec Blacklane	8
1.7	Positionnement de l'application SPN-Cars par rapport aux alternatives.	9
1.8	Méthode en cascade	11
2.1	Diagramme de cas d'utilisation généralisé	16
2.2	Diagramme de cas d'utilisation spécifique : Gestion de profil.	17
2.3	Diagramme de cas d'utilisation spécifique : Créer réservation.	17
2.4	Diagramme de cas d'utilisation spécifique : Gérer réservation.	18
2.5	Diagramme de classe	19
2.6	Diagramme de séquence : Crédit de compte avec E-mail et mot de passe.	20
2.7	Diagramme de séquence : Crédit de compte avec Un compte Google / Apple.	21
2.8	Diagramme de séquences : Authentification.	22
2.9	Diagramme de séquences : Page d'accueil.	23
2.10	Diagramme de séquences : Demander une location.	24

2.11	Diagramme de séquences : Demander un transfert.	25
2.12	Diagramme de séquences : Choisir une voiture	26
2.13	Diagramme de séquences : Envoi du contrat de location.	27
2.14	Diagramme de séquences : Paiement des services.	28
2.15	Première page.	29
2.16	Page de connexion.	29
2.17	Page d'accueil.	29
2.18	Page d'accueil lors d'un transfert en cours.	29
2.19	Sélection de type de transfert.	30
2.20	Suivi de la position actuelle du chauffeur avec la voiture.	30
2.21	Liste de voitures disponibles.	30
2.22	Détails de la voiture sélectionnée.	30
2.23	Messagerie instantanée avec le chauffeur.	31
2.24	Logo Flutter	31
2.25	Logo Express	32
2.26	Logo JWT	32
2.27	Logo MongoDB	33
2.28	Logo Firebase	33
2.29	Logo Git	34
2.30	Logo Postman.	35
2.31	Logo Adobe Xd	35
2.32	Logo Visual Studio Code.	36
2.33	Logo Swagger	36
3.1	Formulare de création de compte : 1ère étape.	41
3.2	Formulare de création de compte : 2ème étape.	41
3.3	Page de Login.	42
3.4	Validation des champs de Login.	42
3.5	Login avec Google.	43
3.6	Formulaire de demande de transfert.	45
3.7	Utiliser le bouton «Localiser» pour choisir la position actuelle.	45
3.8	Rechercher un emplacement à l'aide de Google Places.	45
3.9	Afficher la meilleure route entre le point de départ et le point d'arrivée.	45

TABLE DES FIGURES

7

3.10	Liste des voitures disponibles selon le point de départ sélectionné.	46
3.11	Affichage si aucune voiture n'est disponible.	46
3.12	Création de réservation.	47
3.13	Page de vérification de signature	47
3.14	Vérification de signature avec un document non signé.	48
3.15	Email contenant les documents à signer.	48
3.16	Choix de signature.	48
3.17	confirmation de signature.	48
3.18	Interface de paiement.	49
3.19	Détails de paiement.	49
3.20	Affichage détails de la réservation.	50
3.21	Liste de messages de l'utilisateur.	51
3.22	Conversation avec utilisateur.	51
3.23	Schéma de documentation des API.	52
3.24	Documentation des API.	53
3.25	Détails API ajout voiture.	54

Introduction générale

Le marché mondial de location de voitures continue, chaque année, à se développer et à s'amplifier, et le besoin d'atteindre un nombre maximal de clients est devenu une nécessité. Surtout avec l'avancement technologique qui a permis à tous le monde d'avoir un smartphone. Cet appareil capable de réaliser plusieurs fonctionnalités en simples clics.

Pendant les années précédentes, avec la pandémie mondiale de COVID-19, qui a forcé des milliards de personnes à rester chez eux, il est devenu indispensable de digitaliser plusieurs services afin de les rendre plus accessible, plus fiable, et continuer à exister dans un marché qui continue à évoluer et innover. Plusieurs personnes ont pu se bénéficier de ses services sans quitter leurs maisons et risquer d'être atteints par la COVID, grâce aux différentes entreprises qui ont choisi de continuer à servir leurs clients à l'aide des applications mobiles.

L'entreprise Swiss Premium Negoce a vu cet événement mondial comme une opportunité pour s'introduire dans le monde vaste de nouvelles technologies et présenter leurs différents services sous forme numérique à l'aide des sites web et applications mobiles. Cette étape qui les permettra de rester toujours proche de sa clientèle, et continuer à fonctionner sans tenir compte de la pandémie.

L'un des services offerts par Swiss Premium Negoce est le service de location de voitures de luxe. Ce service permet aux utilisateurs de trouver leurs voitures de rêves et la louer même pour une courte durée avec la possibilité d'avoir un chauffeur personnel. Tout cela sera possible à l'aide de l'application Swiss Premium Negoce : Cars ou tout simplement : SPN Cars. Le présent rapport qui présentera cette application, s'étalera sur quatre chapitres :

- Le premier chapitre présentera l'organisme d'accueil, ainsi le projet à réaliser et une étude sur les différents aspects de ce projet.
- Le deuxième chapitre fera l'objet de présenter les différents besoins qui seront achevés par cette application.

- Le troisième chapitre sera à propos la conception de l'application, et les technologies choisies pour réaliser le projet suite à cette application.
- Le quatrième et dernier chapitre présentera la réalisation de ce projet, qui est une explication détaillée de chacune des fonctionnalités offerts par l'application avec des captures d'écran pour illustrer le travail réalisé.

Le projet sera clôturé par une conclusion générale.

Chapitre 1

Cadre du projet

Contents

1.1 Présentation de l'organisme d'accueil	4
1.1.1 Présentation générale	4
1.1.2 Activités	4
1.2 Présentation du projet	5
1.2.1 Présentation générale	5
1.2.2 Problématique	5
1.2.3 Étude de l'existant	5
1.2.4 Critique de l'existant	7
1.2.5 Solution Proposée	8
1.3 Méthodologie de travail	9

Introduction

Vu que ce projet de fin d'études sera réalisé au sein d'une entreprise accueillante, il s'avère être indispensable d'avoir une idée sur cette dernière. Par suite, nous allons mettre l'accent sur le sujet et les motivations derrière ce projet.

1.1 Présentation de l'organisme d'accueil

Ce projet a été réalisé au sein du tout nouveau département IT de ***Swiss Premium Negoce*** [1].

Ce département ce compose d'une équipe très innovante, très talentueuse qui est responsable des différents applications et sites web de tous les différents services proposés par SPN.

1.1.1 Présentation générale



FIGURE 1.1 – Logo Swiss Premium Negoce

Swiss Premium Negoce SA (Société Anonyme) est une société de conciergerie basée à Genève, Suisse. Créée en 2011 avec passion et enthousiasme, l'entreprise se spécialise en services de luxe. En 2022, SPN a inauguré SPN Tunis avec son nouveau département IT, un département qui permettra d'atteindre plus d'audience à travers la digitalisation des différents services offerts par SPN.

1.1.2 Activités

SPN offre plusieurs services de luxe tel que :

- Conciergerie
- Hospitalité de luxe
- Location de voitures
- Santé et services médicaux
- Camps d'été

- Éducation
- Jets privés
- Yachting
- Events

1.2 Présentation du projet

1.2.1 Présentation générale

L'application SPN-Cars est la solution trouvée par SPN pour offrir l'un de ses services les plus demandés plus facilement et convenablement grâce à l'utilisation de plusieurs nouvelles technologies qui permettront à l'entreprise de mieux performer et atteindre plus d'utilisateurs. On doit alors définir ce projet, son objectif, ses concurrents, les problèmes qu'ils créent et comment l'application SPN-Cars pourra les corriger et offrir un service meilleur que les autres applications disponibles sur le marché.

1.2.2 Problématique

L'application SPN-Cars vise à devenir l'un des leaders dans les domaines de location de voiture en ligne, réservation de chauffeurs et des services taxis. Ce qui pose un grand défi : **Comment fournir aux utilisateurs de l'application ses services en tenant compte de la rapidité de l'application ainsi que la facilité des procédures ?**

1.2.3 Étude de l'existant

Il existe déjà plusieurs applications qui offrent des services similaires, chacune de ses applications présente des avantages et des inconvénients qu'on les explorera dans les chapitres suivants.

Sixt



FIGURE 1.2 – Logo Sixt

Sixt [2] est un fournisseur international de voitures de location, avec plus de 2000 véhicules dans 110 différents pays.

Sixt offre plusieurs services en rapport avec la location des voitures :

- Location des voitures.
- Covoiturage.
- Service taxi.
- Location de voiture par abonnement mensuel.

Blacklane



FIGURE 1.3 – Logo Blacklane

Blacklane [3] est une entreprise allemande qui offre à ses clients la possibilité de réserver un chauffeur avec une voiture luxueuse et vivre une expérience VIP.

L'entreprise offre principalement trois services :

- Trajets longue distance.
- Chauffeurs à la demande.
- Transfert aéroport.

Uber



FIGURE 1.4 – Logo Uber

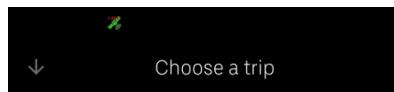
Uber [4] est une entreprise de technologies américaine qui propose à ses clients une solution de trouver des chauffeurs pour les transporter vers leurs destination choisie grâce à son application mobile.

Uber a commencé en tant qu'une application qui joue le rôle d'intermédiaire entre ses utilisateurs et ses chauffeurs indépendants qui offrent leurs services.

1.2.4 Critique de l'existant

Même si les exemples mentionnés ci-dessus sont les leaders mondiaux dans le domaine de services de transport, ils sont tous différents. Surtout en terme de qualité de services, du marché visé.

Uber, par exemple, est à la recherche d'attirer un nombre maximal d'utilisateurs, c'est pourquoi l'application offre une sélection très variée de moyens de transport (voire fig. 5). Blacklane, par contre, vise une clientèle plus exclusive, une clientèle qui est prête à payer pour avoir un service de luxe. Ce qui est visible lors de la sélection de voitures à l'aide de l'application Blacklane comme le montre la figure ci-dessous (voir fig. 6)



Popular

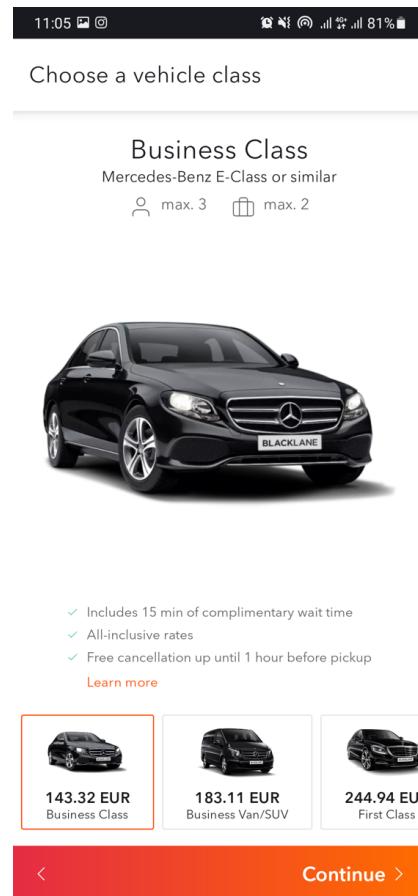


FIGURE 1.5 – Sélection de voiture avec Uber

FIGURE 1.6 – Sélection de voiture avec Blacklane

1.2.5 Solution Proposée

L’objectif de l’application SPN-Cars est de fournir à ses utilisateurs des expériences luxueuses en simples clics.

L’application doit avoir un

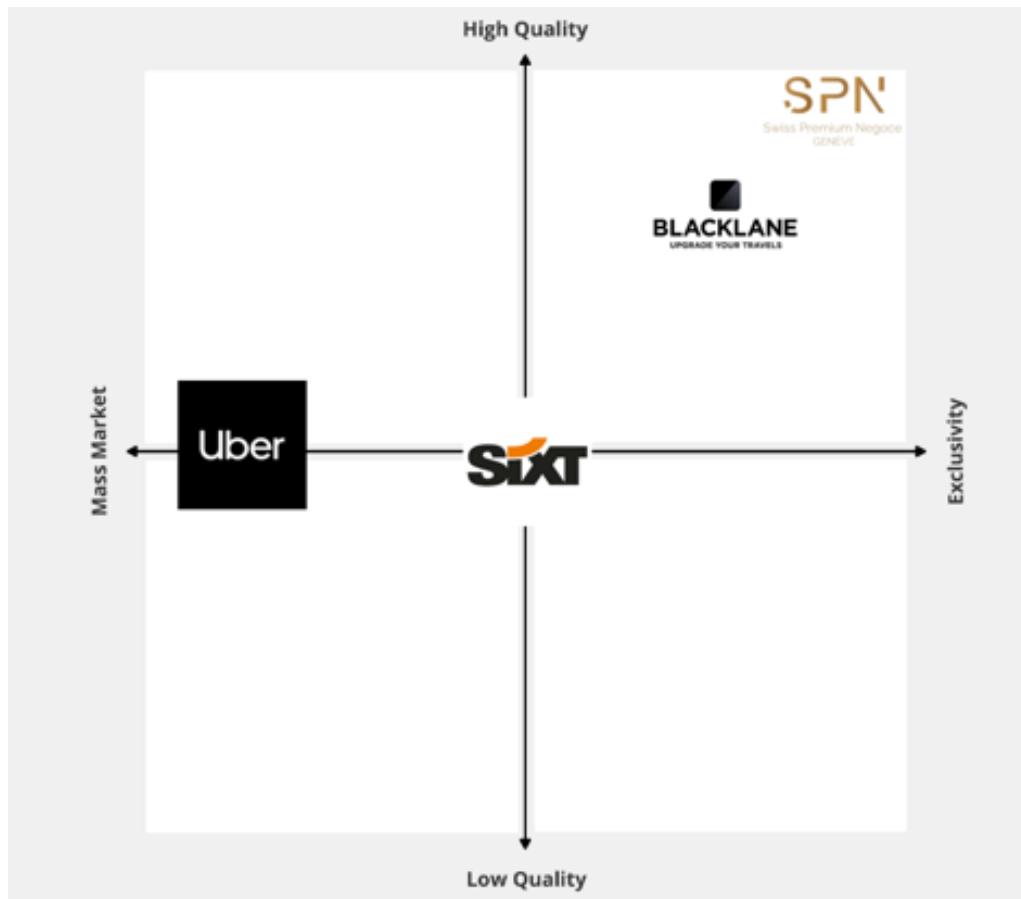


FIGURE 1.7 – Positionnement de l’application SPN-Cars par rapport aux alternatives.

1.3 Méthodologie de travail

L’objectif est de pouvoir mener ce projet en respectant les délais et les budgets alloués. Pour atteindre cet objectif, il faut adopter une méthode de gestion de projet. Il existe de nombreuses méthodes à choisir dont on cite :

- **Les méthodes traditionnelles :** Ces méthodes sont les plus utilisées en gestion de projet. Elles sont appelées également «En cascade» car chaque étape, car chaque étape doit être terminée pour passer à la suivante. En appliquant cette méthodologie, l’équipe projet suit le cahier de charges à la lettre et travaille sur la totalité du projet jusqu’à sa livraison. Il n’y a pas d’interaction avec le client qui recevra son projet une fois que celui-ci est terminé.
- **Les méthodes agiles :** Ces méthodes sont moins rigides que les méthodes traditionnelles, elles placent les besoins du client au centre des priorités du projet. Elles offrent une plus grande flexibilité et une meilleure visibilité dans la gestion du projet, ce qui permet à l’équipe d’être plus réactives aux attentes de client. Le projet sera découpé en mini-

projets chacun nécessitant la validation du client pour passer au suivant.

Comparaison des méthodes

Chaque méthode présente des avantages et des inconvénients, il faut alors les comparer pour identifier la méthode appropriée pour ce projet.

Dans le tableau suivant on présente une comparaison des méthodes :

	Méthodes classiques	Méthodes agiles
Cahier de charges	Un cahier de charge complet doit être disponible.	Se concentrent sur la phase de développement, le cahier de charges est souvent incomplet.
Temps passé	Le client n'est impliqué qu'au phases de conception et livraison de l'application.	L'équipe de développement et le donneur d'ordre communiquent constamment pour échanger sur le projet et planifier les sprints.
Type de projet	Ces méthodes sont préférées pour les projets complexes où rien n'est laissé au hazard.	Cette méthode convient aux projets incertains et innovants.

Méthode retenue

Comme pour ce projet il existe un cahier de charges bien défini, et un temps de développement très stricte, le choix d'une méthode classique, en particulier la méthode en cascade, et imminent.

La méthode en cascade est une méthode linéaire des différentes phases du projet nécessaires pour la création du livrable.

La méthode en cascade est composée de six étapes :

- **Étape des exigences** : C'est la première étape où on définit les exigences et les besoins.
- **Étape de l'analyse** : Faire une analyse plus approfondie des solutions qui répondent aux besoins et écarter celles qui ne correspondent pas aux exigences.
- **Étape de conception** : Dans cette étape on va définir d'une façon détaillée les différents besoins.

- **La mise en œuvre** : Une fois tout est défini et accepté par tous les participants, on commence la partie de mise en œuvre où on développe les différentes fonctionnalités.
- **La validation** : Cette étape consiste à tester les fonctionnalités développées dans l'étape précédente une par une et corriger les problèmes rencontrés.
- **La mise en service** : À la fin de l'étape de validation le projet est mis en service.

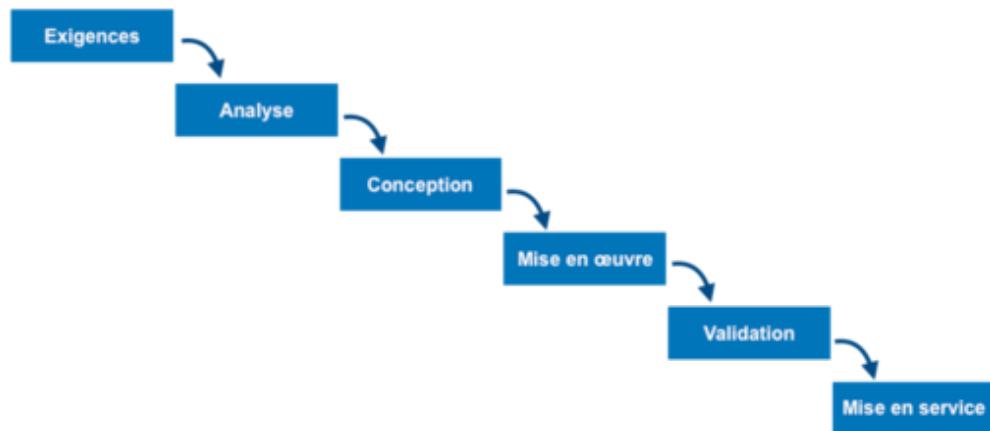


FIGURE 1.8 – Méthode en cascade .

[5]

Conclusion

Dans ce chapitre on a commencé par la présentation de l'entreprise accueillante et ses différentes activités. Puis on a présenté le projet offert et la problématique qu'il pose. Suite à la problématique, il fallait faire une étude de l'existant où on a pris l'exemple de différentes applications qui offrent un service similaires et déduire leurs faiblesses. Une fois ces défauts dégagés, on a pu passer à la présentation de la solution proposée qui permettra de résoudre les points manquants des autres applications. Finalement on a présenté la méthodologie de travail suivie qui assurera la fluidité au niveau de la gestion de ce projet.

Suite à ce chapitre, on passera vers l'étape de conception où on définira les différents aspects du projet.

Chapitre 2

Conception du projet

Contents

2.1	Spécification des besoins	14
2.1.1	Besoins fonctionnels	14
2.1.2	Besoins non fonctionnels	15
2.2	Diagrammes UML	15
2.2.1	Diagramme de cas d'utilisation	15
2.2.2	Créer réservation	17
2.2.3	Gérer réservation	17
2.2.4	Diagramme de classes	19
2.2.5	Diagrammes de séquences	20
2.3	UI/UX Design	28
2.4	Technologies et outils utilisés	31
2.4.1	Flutter	31
2.4.2	Express JS	32
2.4.3	JSON Web Token (JWT)	32
2.4.4	MongoDB	33
2.4.5	Firebase	33
2.4.6	Visual Studio Code	34
2.4.7	Postman	35
2.4.8	Adobe Xd	35
2.4.9	Git	36
2.4.10	Swagger	36

Introduction

Afin de créer une application respectant les besoins définis dans le cahier de charges, il faut d'abord passer par l'étape de conception. Cette étape permettra de décortiquer le cahier de charges et définir les acteurs, les cas d'utilisation, et finalement les différentes classes qui définiront les différents attributs de chaque entité qui interagit avec l'application.

2.1 Spécification des besoins

La première étape de la conception est de dégager les différents besoins de l'utilisateur. Ces besoins sont divisés en deux catégories : fonctionnels et non fonctionnels.

2.1.1 Besoins fonctionnels

L'authentification

Le client doit s'authentifier pour utiliser les différents services de l'application.

Consulter les voitures disponibles

Le client peut consulter les voitures disponibles selon sa position actuelle.

Créer une réservation

Le client peut créer une réservation en choisissant une voiture.

Paiement en ligne

Le client peut payer pour sa réservation à l'aide des méthodes de paiement en ligne.

Signature numérique de contrat de location

Le client peut signer son contrat de location en ligne en toute sécurité.

Suivre la position de la voiture

Le client peut suivre la position de la voiture louée en temps réel.

Contacter le chauffeur

Le client peut contacter le chauffeur de sa voiture louée par appel téléphonique ou messagerie instantanée.

2.1.2 Besoins non fonctionnels

Interfaces graphiques

L'utilisateur peut rapidement commencer à utiliser l'application sans difficulté à l'aide des interfaces graphiques claires et simples.

Performances de l'application

L'application ne doit pas présenter des imperfections en terme de rapidité et de fluidité en cours de son exécution.

Maintenabilité et scalabilité

L'application doit être simple à maintenir et facilement scalable dans le futur.

Sécurité

Toutes les informations traitées dans l'application doivent être protégées, et une vérification de l'utilisateur est impérative avant chaque opération.

2.2 Diagrammes UML

2.2.1 Diagramme de cas d'utilisation

L'application SPN-Cars a un seul acteur qui est l'utilisateur, ce dernier utilisera toutes les fonctionnalités offertes par l'application.

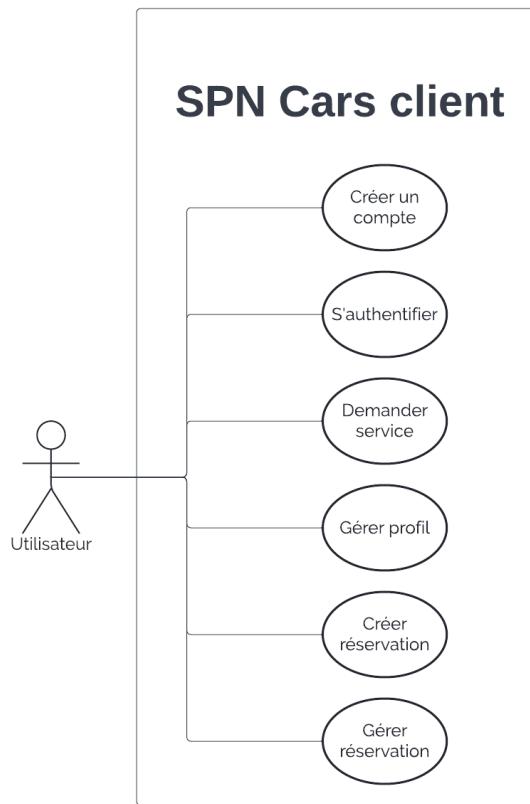


FIGURE 2.1 – Diagramme de cas d'utilisation généralisé

Gérer profil

L'utilisateur peut gérer son profil : Il peut changer sa photo de profil et ses informations personnels.

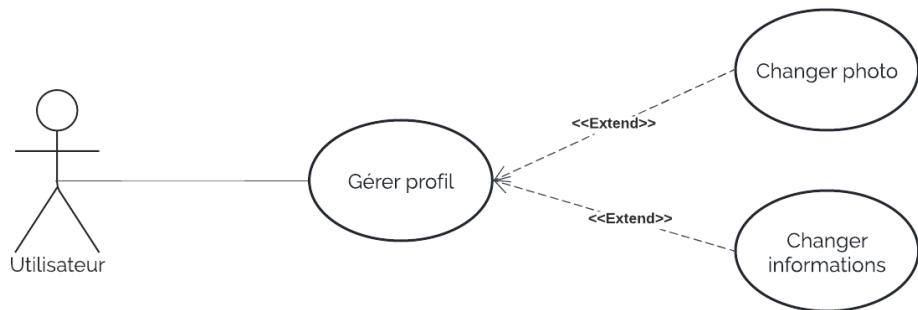


FIGURE 2.2 – Diagramme de cas d'utilisation spécifique : Gestion de profil.

2.2.2 Créer réservation

Pour créer sa réservation, l'utilisateur doit Louer une voiture, effectuer un paiement et signer le contrat numérique de location.

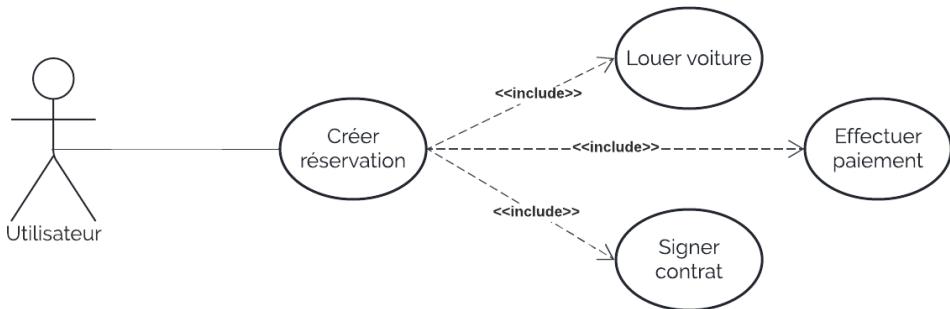


FIGURE 2.3 – Diagramme de cas d'utilisation spécifique : Créer réservation.

2.2.3 Gérer réservation

L'utilisateur peut gérer sa réservation, il peut suivre la position de la voiture louée en temps réel, et il peut aussi annuler sa réservation.

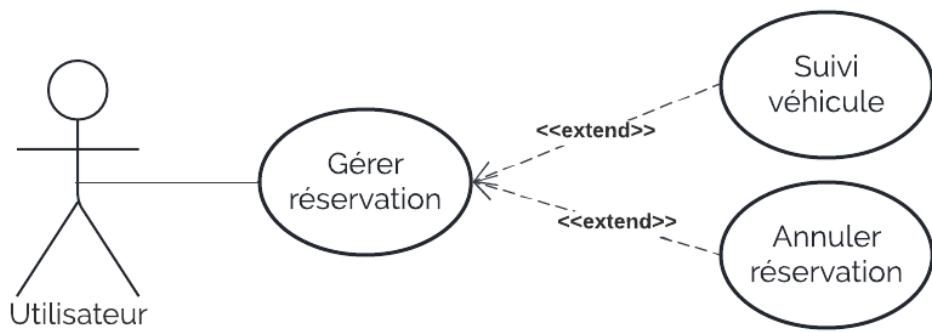


FIGURE 2.4 – Diagramme de cas d'utilisation spécifique : Gérer réservation.

2.2. DIAGRAMMES UML

2.2.4 Diagramme de classes

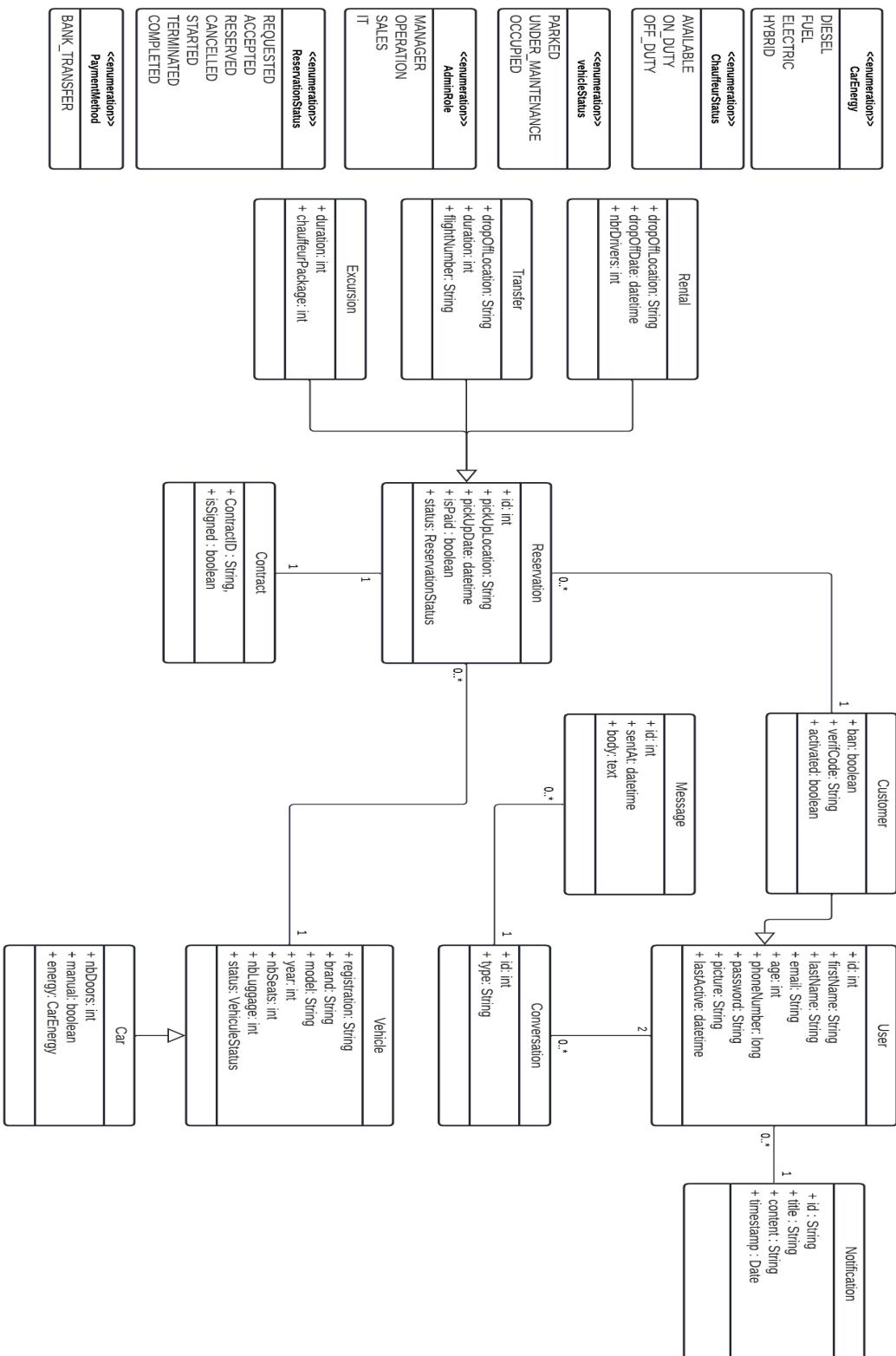


FIGURE 2.5 – Diagramme de classe

2.2.5 Diagrammes de séquences

Les diagrammes de séquences sont le moyen qui permettent de décrire d'une manière détaillée les différents cas d'utilisation

Création de compte

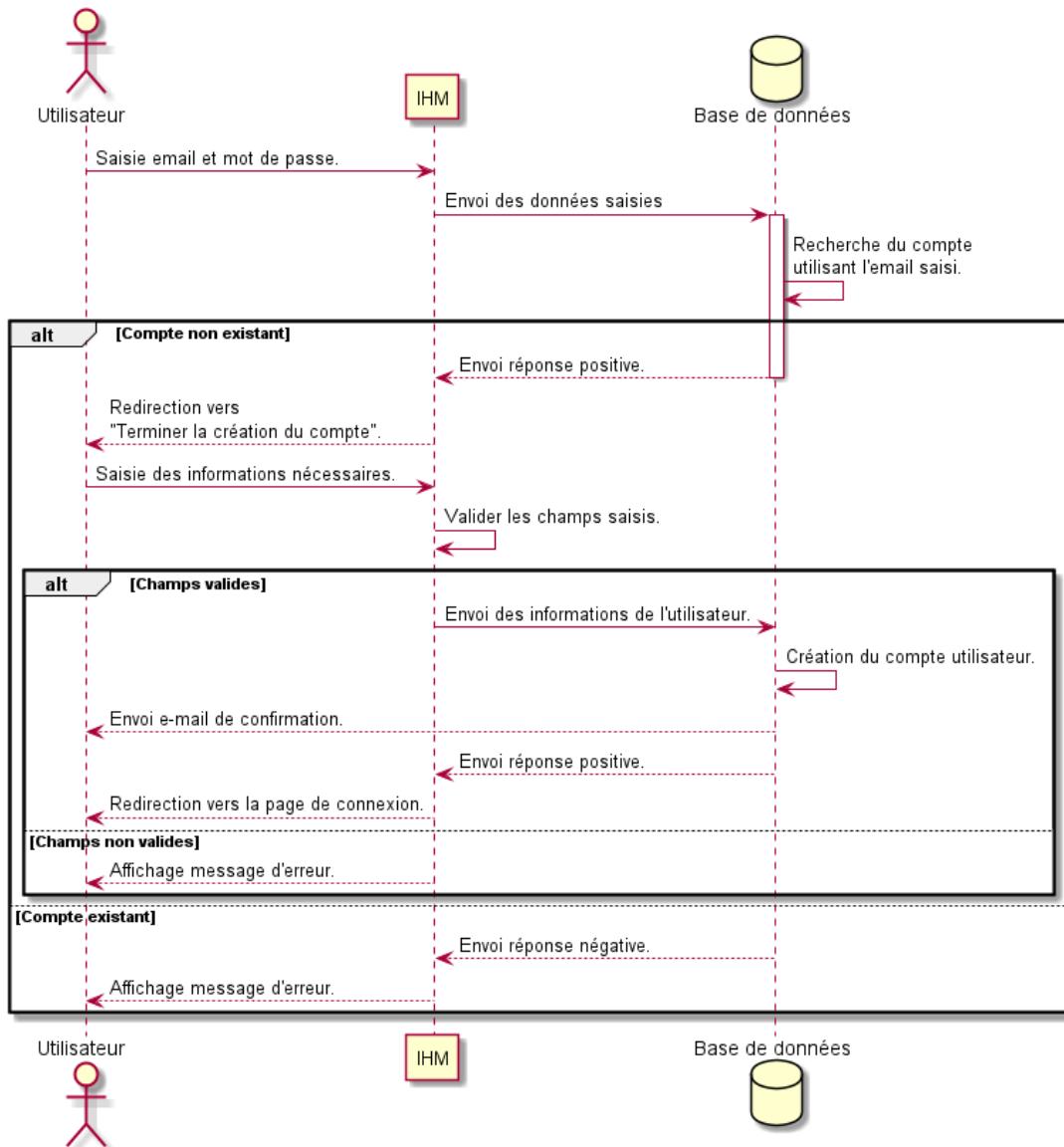


FIGURE 2.6 – Diagramme de séquence : Crédit de compte avec E-mail et mot de passe.

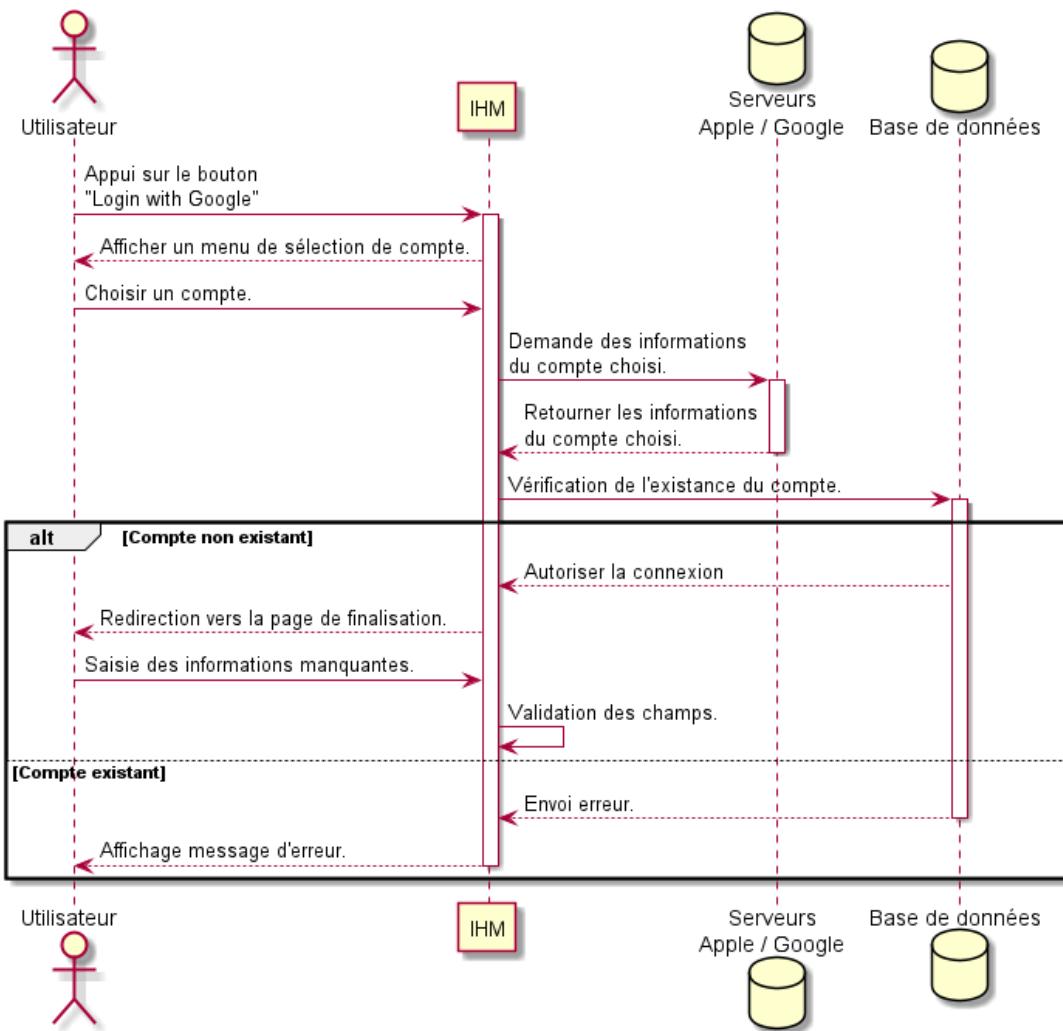


FIGURE 2.7 – Diagramme de séquence : Crédit de compte avec Un compte Google / Apple.

Authentification

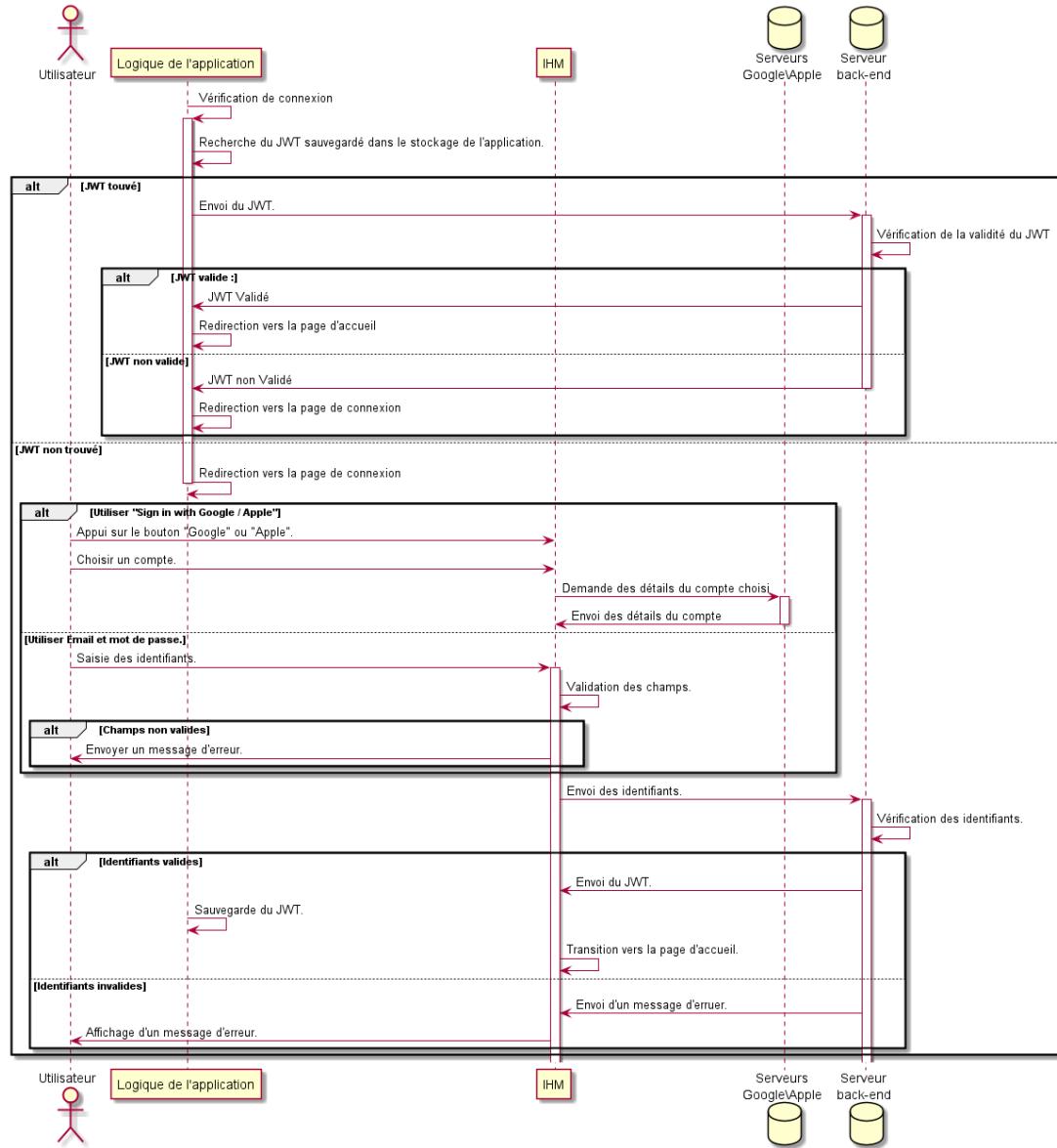


FIGURE 2.8 – Diagramme de séquences : Authentification.

Page d'accueil

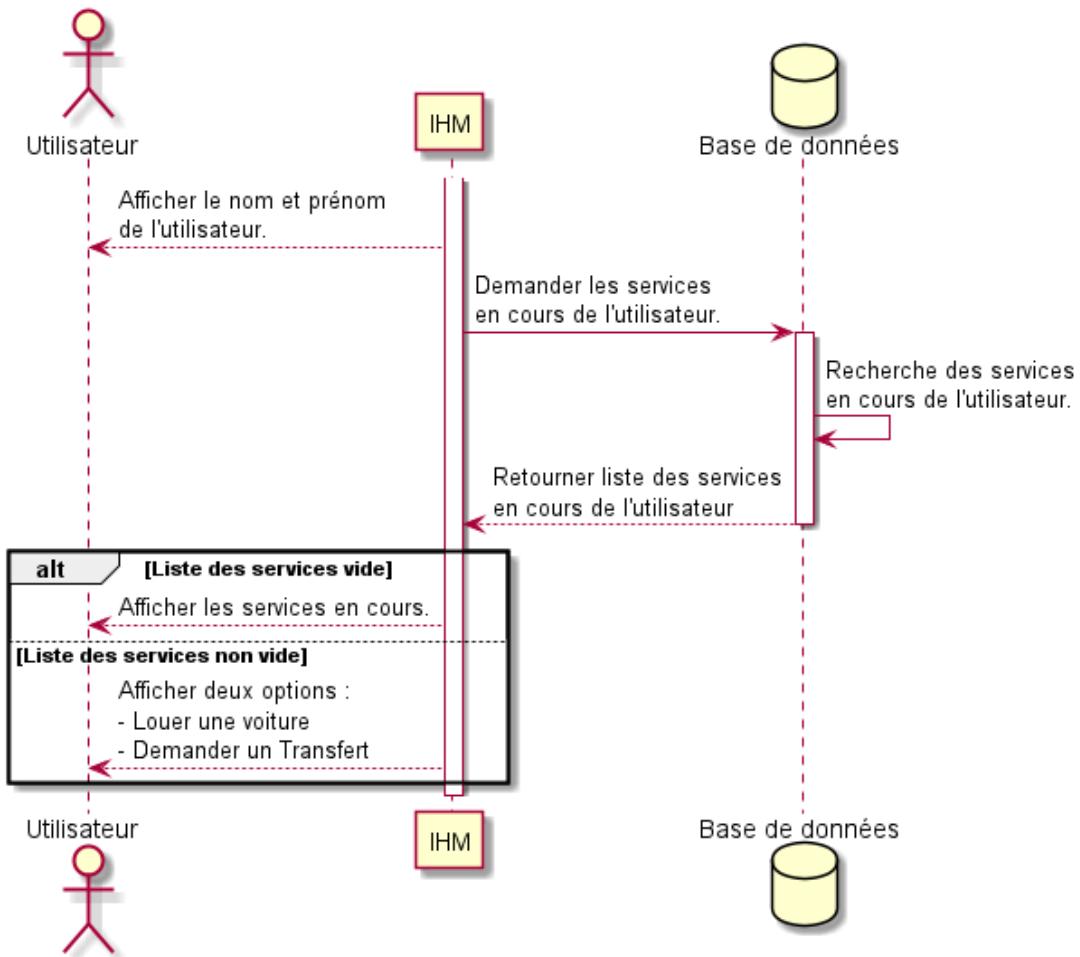


FIGURE 2.9 – Diagramme de séquences : Page d'accueil.

Demander un service

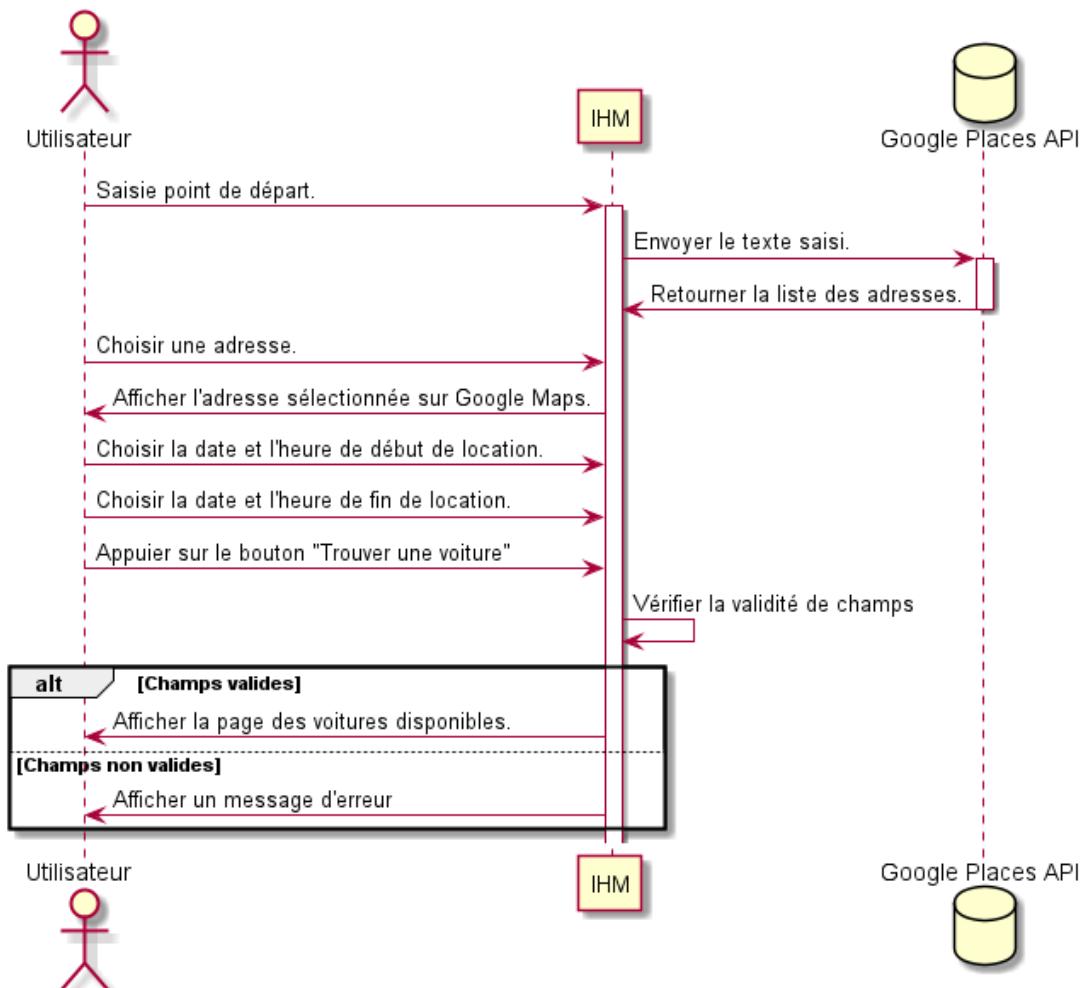


FIGURE 2.10 – Diagramme de séquences : Demander une location.

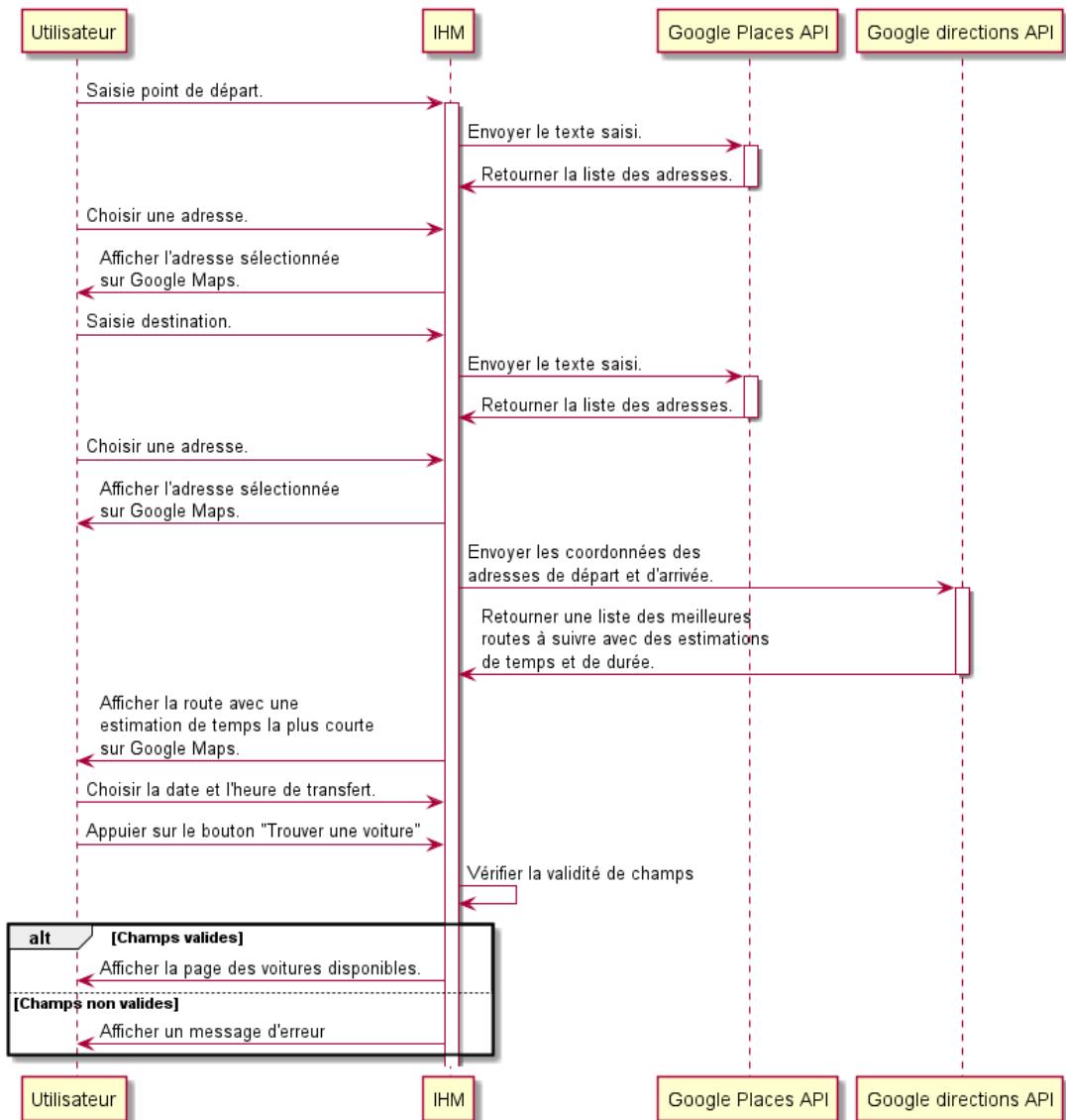


FIGURE 2.11 – Diagramme de séquences : Demander un transfert.

Affichage des voitures disponibles

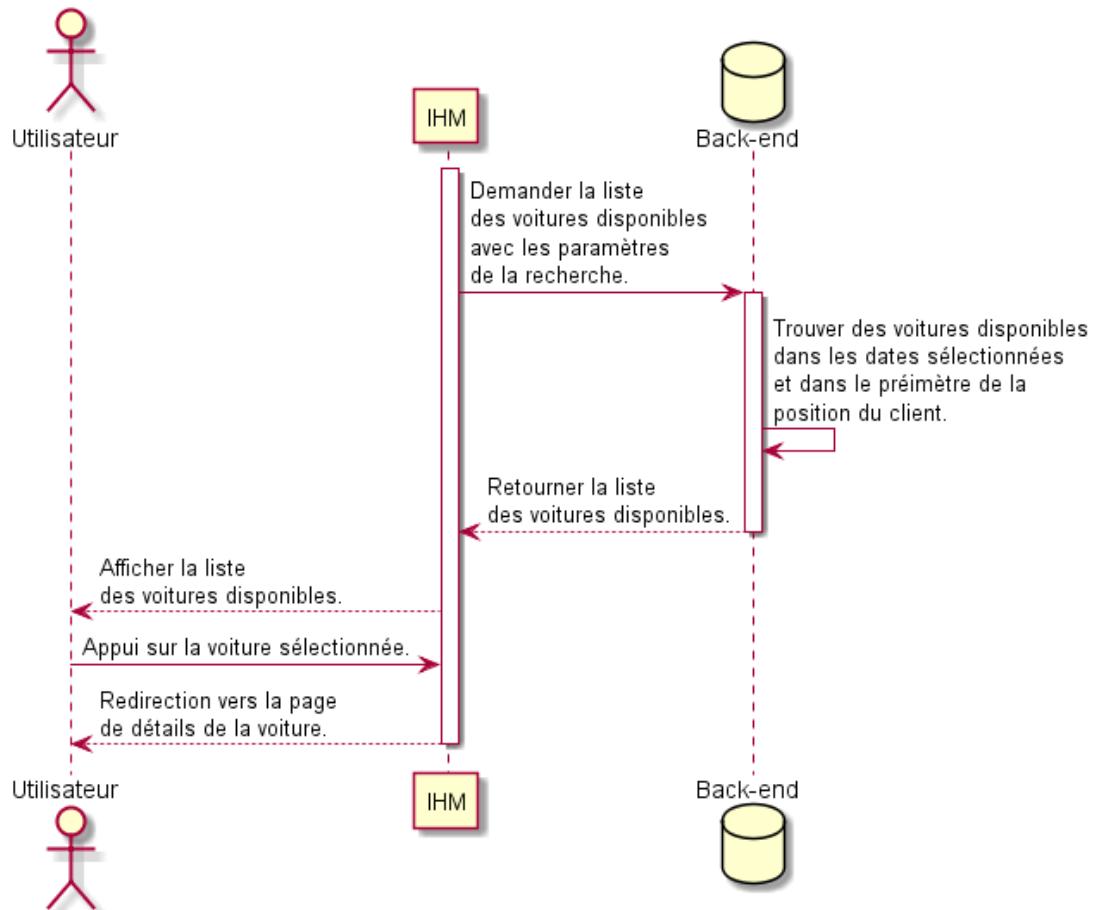


FIGURE 2.12 – Diagramme de séquences : Choisir une voiture

Signature numérique de contrat de location

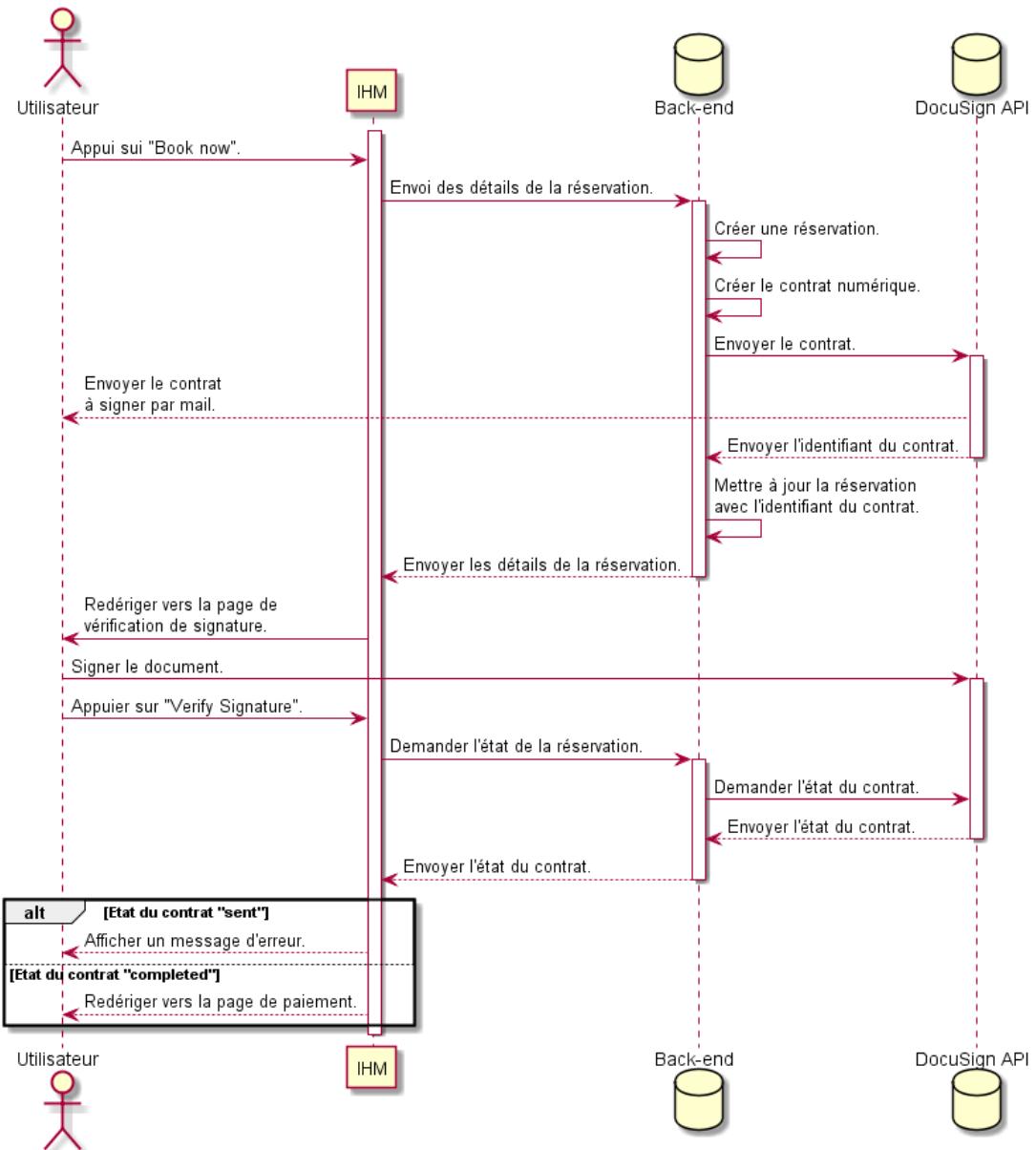


FIGURE 2.13 – Diagramme de séquences : Envoi du contrat de location.

Paiement en ligne

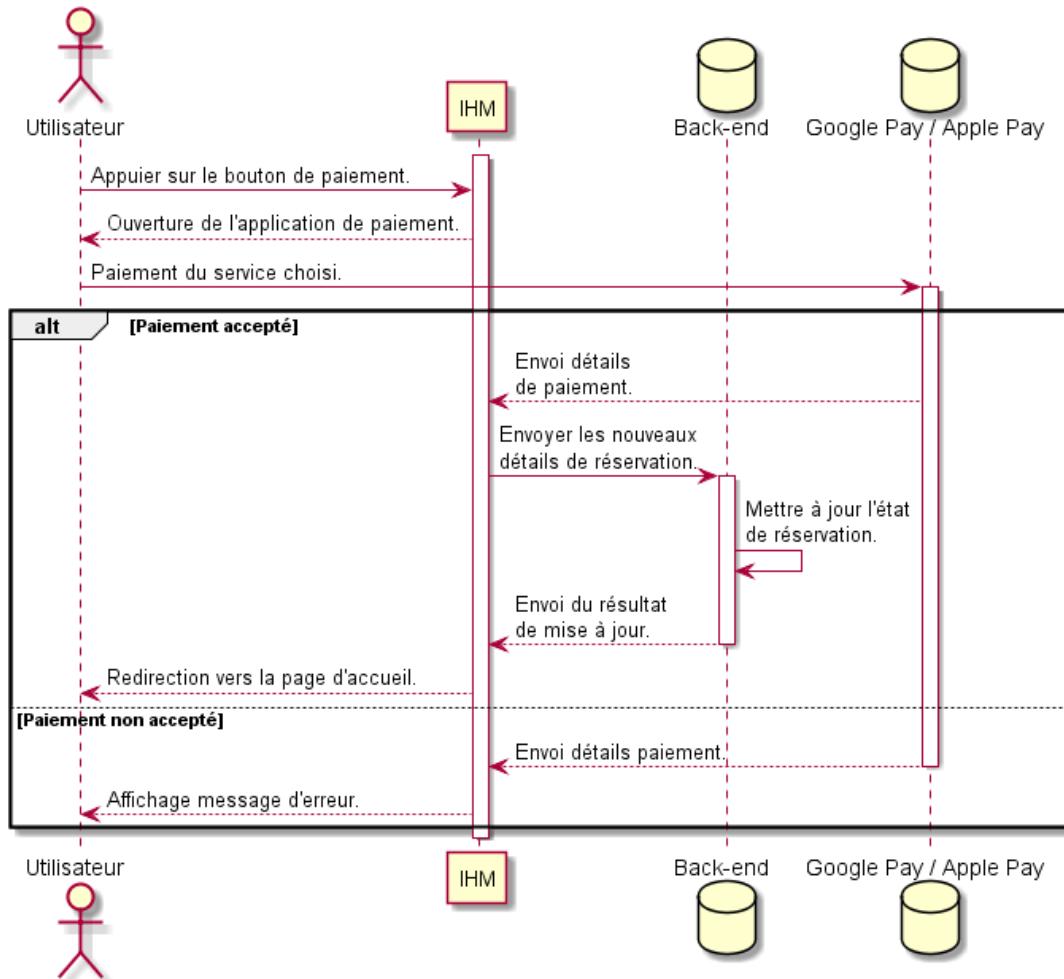


FIGURE 2.14 – Diagramme de séquences : Paiement des services.

Localiser une voiture

2.3 UI/UX Design

Avant de passer au développement de l'application, il faut créer d'abord les prototypes des interfaces utilisateur.

Cette étape est nécessaire pour tester plusieurs approches dans les interfaces de l'application, s'assurer d'offrir une expérience optimale pour l'utilisateur.

Suite à une collaboration avec l'équipe de design UI/UX, les interfaces suivantes ont été créées.



Welcome to SPN
luxury cars.
Fast service, highly skilled and
professional chauffeurs.

Get started

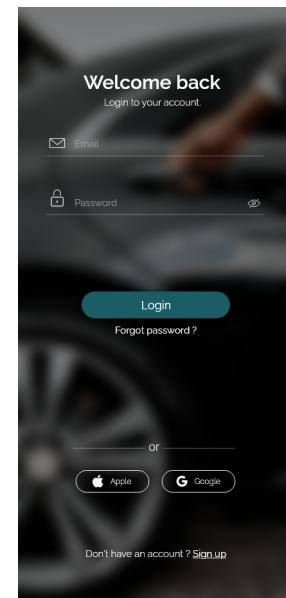


FIGURE 2.15 – Première page.

FIGURE 2.16 – Page de connexion.

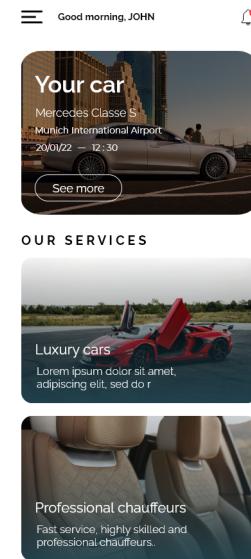
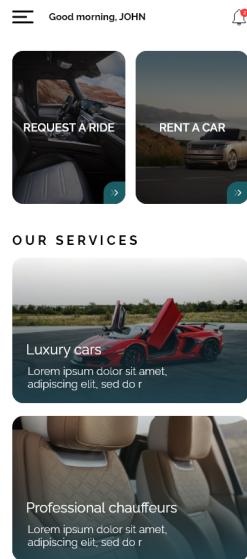


FIGURE 2.17 – Page d'accueil.

FIGURE 2.18 – Page d'accueil lors d'un transfert en cours.

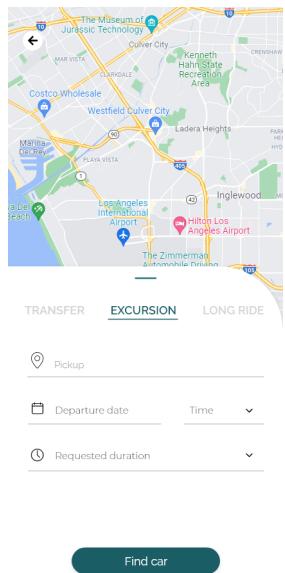


FIGURE 2.19 – Sélection de type de transfert.

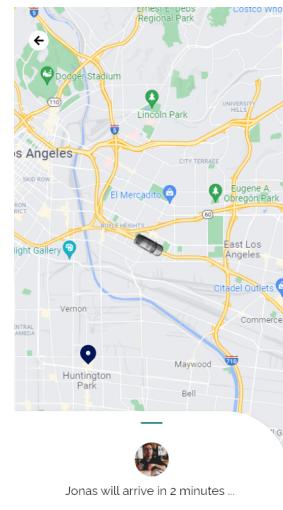


FIGURE 2.20 – Suivi de la position actuelle du chauffeur avec la voiture.



FIGURE 2.21 – Liste de voitures disponibles.



FIGURE 2.22 – Détails de la voiture sélectionnée.

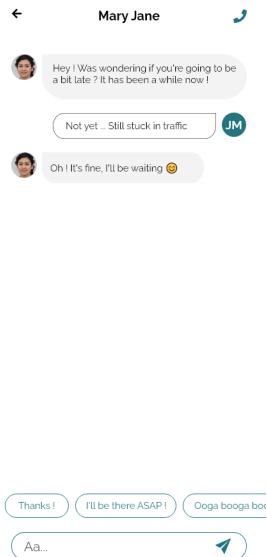


FIGURE 2.23 – Messagerie instantanée avec le chauffeur.

2.4 Technologies et outils utilisés

Comme cette application sera une application mobile, il est nécessaire que la performance soit la priorité lors du développement. C'est pourquoi on a choisi les technologies suivantes pour offrir une application rapide, performante, facile à utiliser.

Ses différentes technologies sont classifiées dans trois domaines principalement : Conception des interfaces graphiques, développement de l'application, et développement du back-end de l'application.

2.4.1 Flutter



FIGURE 2.24 – Logo Flutter

Flutter [6] est un kit de développement (SDK) open-source créé par Google et publié en 2017.

Flutter permet de créer des applications mobiles (Android / iOS), web et même desktop (Windows / Linux / MacOS), avec une seule base de code en Dart, un langage de programmation développé aussi par Google.

Flutter présente plusieurs avantages qui permettent de créer des applications mobiles performantes et réduit aussi le coût et le temps de développement nécessaires, grâce au langage de programmation utilisé **Dart** qui est très facile à maîtriser et qui offre plusieurs avantages, dont le plus important la fonctionnalité de «**Hot Reload**» qui permet de recharger l'application et afficher les changements sur l'écran sans passer par la recompilation du code source.

2.4.2 Express JS



FIGURE 2.25 – Logo Express

Express JS [7] est un framework back-end gratuit et open-source pour NodeJS. Créé par TJ Holowaychuk, la première version publique d'Express JS a été introduite au public en 2010. Express JS est un framework minimaliste, très léger pour garantir une performance optimale et une exécution rapide. Ce framework est aussi très flexible, même s'il fournit que quelques fonctionnalités, grâce à **NPM**, le gestionnaire de paquets de NodeJS, il peut être complété par plusieurs librairies disponibles.

Grâce à son minimalisme et facilité d'implémentation, ExpressJS est utilisé par plusieurs par nombreuses sociétés dans le monde, pour développer tout type d'applications, parmi ses sociétés il y a des géants de technologies tels que **IBM**, **Uber**, et plusieurs autres.

2.4.3 JSON Web Token (JWT)



FIGURE 2.26 – Logo JWT

Josn Web Token ou tout simplement connu comme "JWT" est un standard créé en 2015, qui permet d'encapsuler des données dans des jetons et l'échanger entre plusieurs parties en

toute sécurité.

Un jeton est composé principalement de trois parties :

- Un en-tête qui décrit le jeton en format JSON.
- Le contenu de ce jeton qui est également en format JSON.
- Une signature numérique.

2.4.4 MongoDB



FIGURE 2.27 – Logo MongoDB

MongoDB [8], est un système de gestion de base de données NoSQL, orientée documents. Une base de données NoSQL est utilisée pour le stockage de volumes massifs de données, elle se distingue des bases de données relationnelles par sa flexibilité et ses performances.

Le système MongoDB est développé par la société qui porte le même nom en 2007. Cette entreprise travaillait sur un système de cloud computing à données largement réparties.

Il est depuis devenu l'un des systèmes de gestion de base de données les plus utilisées, notamment pour des sites web très populaires tels que : *SourceForge.net*, *eBay* et *The New York Times*.

Contrairement à une base de données relationnelle SQL traditionnelle, MongoDB ne repose pas sur des tableaux et des colonnes. Les données sont stockées sous forme de collections et de documents. Les documents sont des paires de valeurs / clés servant d'unité de données de base. Les collections quant à elles contiennent des ensembles de documents et de fonctions. Elles sont l'équivalent des tableaux dans les bases de données relationnelles classiques.

2.4.5 Firebase



FIGURE 2.28 – Logo Firebase

Firebase [9] est une plateforme créée en 2011, puis acquise et développée par Google en 2014. Firebase facilite la création de back-end à la fois scalable et performant.

L'objectif de firebase est d'offrir aux professionnels et aux particuliers un moyen d'éviter l'engagement dans un processus complexe de création et de maintenance d'une architecture serveur. Firebase offre des API intuitives regroupées dans un SDK unique. Ces API permettent de gagner du temps et de réduire le nombre d'intégrations qu'on doit gérer dans l'application.

Firebase offre plusieurs services que tout le monde peut utiliser gratuitement grâce à sa politique «pay as you go» qui nécessite le paiement de ses services seulement si l'utilisation des ressources dépasse le quota du plan gratuit offert. Les services de Firebase les plus utilisés sont :

- **Firestore :**

Une base de données NoSQL, bénéficiant d'un hébergement cloud et permettant le stockage et la synchronisation de données des utilisateurs.

- **Fireabse Authentification :**

Un SDK prêt et facile à exploiter qui permet de d'authentifier les utilisateurs en offrant plusieurs méthodes d'authentification tels que Google, Apple, Facebook, Email et mot de passe, numéro de téléphone et plusieurs d'autres méthodes pour assurer l'authentification de l'utilisateur.

- **Firebase Cloud Messaging :**

Permet de connecter plusieurs périphériques au serveur dans les meilleures conditions (fiabilité et économie de batterie). Ce service permet de recevoir et envoyer des notification sur les différentes plateformes (Web / iOS / Android). Avec Firebase Cloud Messaging, il est possible aussi d'assurer un service de messagerie instantanée entre les utilisateurs.



FIGURE 2.29 – Logo Git

2.4.6 Visual Studio Code

Visual Studio Code [10] est un éditeur de texte open-source développé par Microsoft [11] pour Windows MacOS et Linux en 2015. Riche en fonctionnalités tels que le support de débogage, complétion intelligente du code, intégration de système de contrôle de version Git, et la capacité d'installation d'extensions qui permettent d'améliorer l'expérience de l'utilisateur, Visual Studio Code est devenu l'un des premiers choix pour plusieurs développeurs pour travailler sur leurs différents projets.

2.4.7 Postman



FIGURE 2.30 – Logo Postman.

Postman [12] est une plateforme permettant de concevoir, développer, et tester des API. Crée en 2012 comme une extension pour Google Chrome, Postman a gagné une grande popularité, qui l'a permis de migrer vers une application pour Windows Mac OS, et Linux. Postman maintenant est le premier choix pour la plupart des développeurs, avec plus de 20 millions d'utilisateurs [13].

2.4.8 Adobe Xd



FIGURE 2.31 – Logo Adobe Xd

Adobe Xd [14] est un outil de conception et modélisation des interfaces utilisateur des applications web et mobiles, développé par Adobe Inc.

Grâce aux outils fournis par Adobe Xd, la conception, l'amélioration et la rectification des interfaces graphiques et l'expérience de l'utilisateur de l'application sera plus facile, plus rapide et plus efficace. Dans le cadre de ce projet, Adobe Xd a été utilisé pour la création des prototypes des interfaces graphiques, qui seront, par la suite, construits en application mobile à l'aide de Flutter.

2.4.9 Git

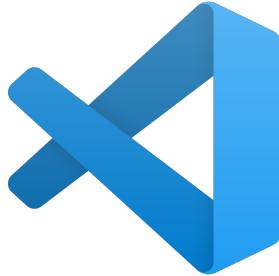


FIGURE 2.32 – Logo Visual Studio Code.

Git [15] est un système de contrôle de version open-source créé en 2005 par **Linus Torvalds**, le créateur du noyau du système d’exploitation Linux.

Git permet de gérer les ajouts et changements apportés au code source de manière tracée. Ainsi, si une erreur est commise, les développeurs peuvent revenir en arrière et comparer les versions antérieures du code, ce qui leur permet de corriger l’erreur tout en minimisant les perturbations pour tous les membres de l’équipe.

2.4.10 Swagger



FIGURE 2.33 – Logo Swagger

Swagger [16] est un langage permettant de créer une description des API Restful à l'aide de JSON. A l'aide de ses outils Open-Source, Swagger permet de concevoir, créer et décrire des API REST.

Swagger est créé en 2011 par Tony Tam, cofondateur du site de dictionnaires Wordnik, suite à un besoin d’automatisation de documentation de l’API qui est devenue de plus en plus frustrante. Juste après sa création, le projet Swagger est devenu open-source en septembre 2011.

Swagger est maintenant maintenu par la société **SmartBear Software** qui, en novembre 2015, a créé une organisation appelée **OpenAPI initiative** dont diverses entreprises tels que Google,

IBM et Microsoft sont les membres fondateurs.

Conclusion

A travers ce chapitre, on a établi la conception de l'application SPN-Cars : On a dégager les besoins fonctionnels et non fonctionnels, qui, à travers eux on a pu créer les diagrammes de cas d'utilisation, les diagrammes de classes, et les diagrammes de séquences. Suite au développement de ces diagrammes, on a passé vers l'étape de conception des interfaces utilisateur. Toutes ces différentes étapes ont permis de choisir les technologies et outils qui seront utilisés pour la création de cette application.

Une fois les différents aspects de l'application bien définis, on passera vers la phase de réalisation où on appliquera les diagrammes et interfaces développés dans ce chapitre pour produire cette application.

Chapitre 3

Réalisation de l'application

Contents

3.1	Création de compte	40
3.1.1	Création de compte avec email et mot de passe	40
3.1.2	Création de compte avec Google ou Apple	41
3.2	Authentification	41
3.3	Page d'accueil	43
3.4	Gestion de profil	44
3.5	Demander un service	44
3.6	Affichage des voitures disponibles	46
3.7	Signature numérique de contrat de location	46
3.8	Paiement	49
3.9	Localiser une voiture	49
3.10	Messagerie instantanée	50
3.11	Documentation des API	51

Introduction

Dans ce chapitre, on décrira toutes les fonctionnalités de l'application, leurs fonctionnement, les différents scénarios d'exécution avec des captures d'écran de chaque interface d'utilisateur avec un diagramme de séquences qui décrit en détail la fonctionnalité mentionnée

3.1 Crédation de compte

La première étape pour utiliser l'application SPN-Cars est de créer un compte. Ce compte permettra aux utilisateurs de bénéficier de tous les services offerts par l'application.

Pour créer un compte l'utilisateur a la possibilité de choisir trois méthodes : Créer un compte avec son email et choisir un mot de passe, ou créer un compte tout simplement en utilisant l'option de création de compte avec son compte Google ou Apple.

3.1.1 Crédation de compte avec email et mot de passe

C'est la méthode la plus basique qui existait depuis toujours. Pour créer un compte, l'utilisateur doit tout d'abord entrer son adresse email, un mot de passe, et une confirmation de cette mot de passe. Lors de l'appui sur le bouton de «Créer un compte», l'application envoie une requête vers le serveur back-end afin de vérifier l'existance d'un compte d'utilisateur utilisant la même adresse e-mail. Après une recherche effectuée sur les utilisateurs, le serveur back-end envoie une réponse positive s'il n'a trouvé aucun compte d'utilisateur utilisant l'adresse e-mail entrée par l'utilisateur, sinon une réponse négative sera envoyée. Selon la réponse retournée par le serveur l'application redirigera l'utilisateur vers une page pour compléter l'étape de création de compte si la réponse du serveur est positive, ou affichera un message d'erreur avec l'erreur adéquate si la réponse du serveur est négative.

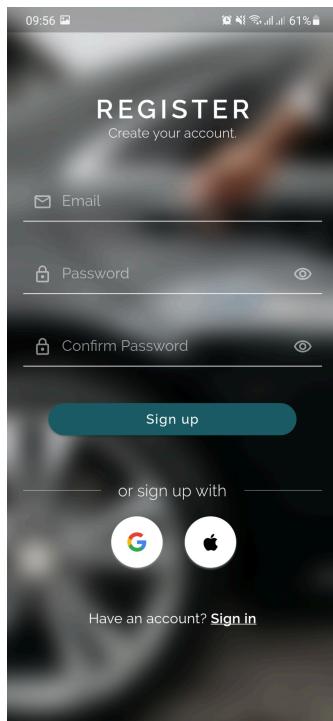


FIGURE 3.1 – Formulare de création de compte : 1ère étape.

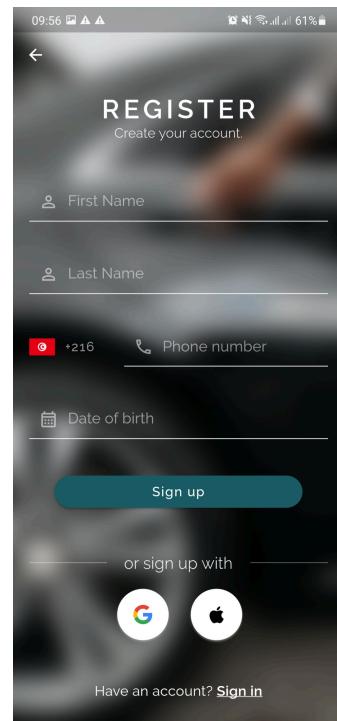


FIGURE 3.2 – Formulare de création de compte : 2ème étape.

3.1.2 Crédation de compte avec Google ou Apple

Cette méthode est la plus facile et la plus rapide pour créer un compte ou s'authentifier. Avec un simple appui sur le bouton adéquat, une requête envoyée aux services de Google ou Apple pour récupérer les données du compte choisi. Ses données sont :

- Un identifiant unique du compte Google ou Apple.
- L'adresse email du compte.
- Le nom et prénom utilisés avec le compte.
- La photo de profil utilisée avec ce compte.

Ses informations seront nécessaires pour passer la première étape de création de compte et avec eux l'utilisateur gagnera un peu de temps lors de la création de son compte.

3.2 Authentification

L'authentification est la première étape dans le cycle de vie de l'application, lors du premier démarrage de l'application il est nécessaire de vérifier si l'utilisateur est déjà connecté à l'application. Grâce à cette étape on peut identifier l'utilisateur, et limiter les requêtes envoyées au serveur back-end.

Pour s'authentifier l'utilisateur peut choisir trois méthodes différentes : Email et mot de passe, avec son compte Google, ou avec son compte Apple.

La validation des champs est une étape nécessaire pour s'assurer que l'utilisateur n'envoie que des valeurs correctes vers les API de connexion, ce qui permet d'éviter les erreurs inattendues.

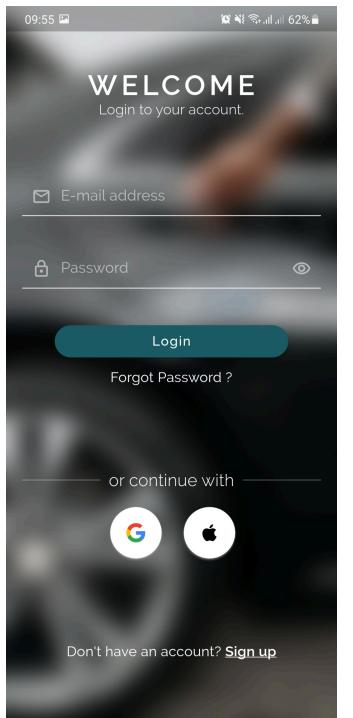


FIGURE 3.3 – Page de Login.

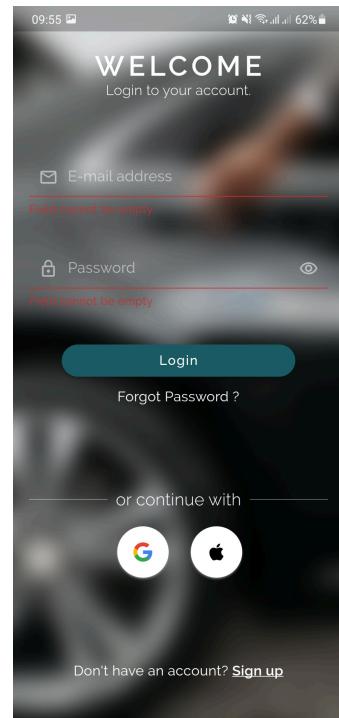


FIGURE 3.4 – Validation des champs de Login.

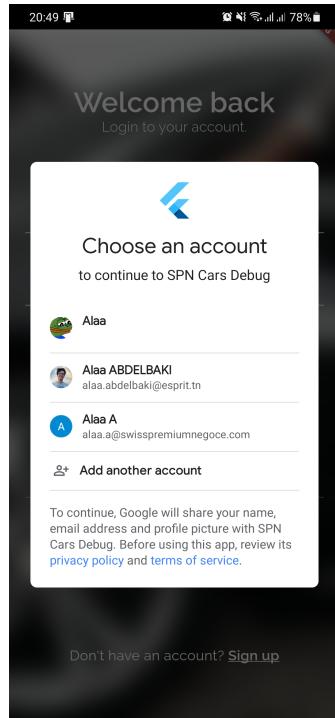


FIGURE 3.5 – Login avec Google.

3.3 Page d'accueil

Une fois l'utilisateur est réussi à s'authentifier, la page d'accueil s'affiche. Cette page diffère d'un utilisateur à un autre selon les services demandés par l'utilisateur : Si l'utilisateur a un service actif le moment de sa connexion, la liste de voitures louées avec les détails de chaque service actif demandé, et s'il n'a pas de services actif le moment de sa connexion, deux boutons seront affichés : «Rent a car» pour louer une voiture sans chauffeur et «Request a transfer» pour demander un chauffeur avec une voiture.

3.4 Gestion de profil

Chaque utilisateur possède un profil personnel, ce profil affichera les informations nécessaires pour faciliter la communication entre utilisateurs dans le futur (Ex : Contact entre chauffeur et client).

La page de profil, dans l'application SPN-Cars, est une page très simple qui contient les informations de base de l'utilisateur :

- La photo de profil.
- Le nom.
- Le prénom
- L'adresse E-mail.
- Le numéro de téléphone.
- La date de naissance.

De toutes ses données seuls le nom, prénom, photo de profil et numéro de téléphone seront accessible aux chauffeurs pour assurer une communication avec les clients.

L'utilisateur peut modifier tous ses informations personnelles tout simplement en modifiant les champs contenant l'information à changer, et pour la photo de profile il suffit d'appuyer sur la photo pour la remplacer par une autre, soit prendre une nouvelle photo en utilisant l'appareil photo du smartphone, ou en choisissant une photo existante depuis la galerie. Une fois l'image est choisie, l'utilisateur sera redirigé vers une page pour recadrer l'image et choisir la zone qui sera affichée dans la page de profile, une fois l'image est recadrée, elle sera compressée pour réduire sa taille et faciliter son transfert vers le serveur distant.

3.5 Demander un service

Pour louer une voiture, l'utilisateur a besoin de spécifier tout d'abord les paramètres suivants :

- Le type de service demandé (Location / Transfert / Excursion / Long Ride).
- L'adresse de départ.
- L'heure de départ.
- L'adresse d'arrivée (Pas toujours disponible selon le type de service).
- L'heure d'arrivée (Pas toujours disponible selon le type de service).
- La durée du service demandé (Pas toujours disponible selon le type de service).

3.5. DEMANDER UN SERVICE

45

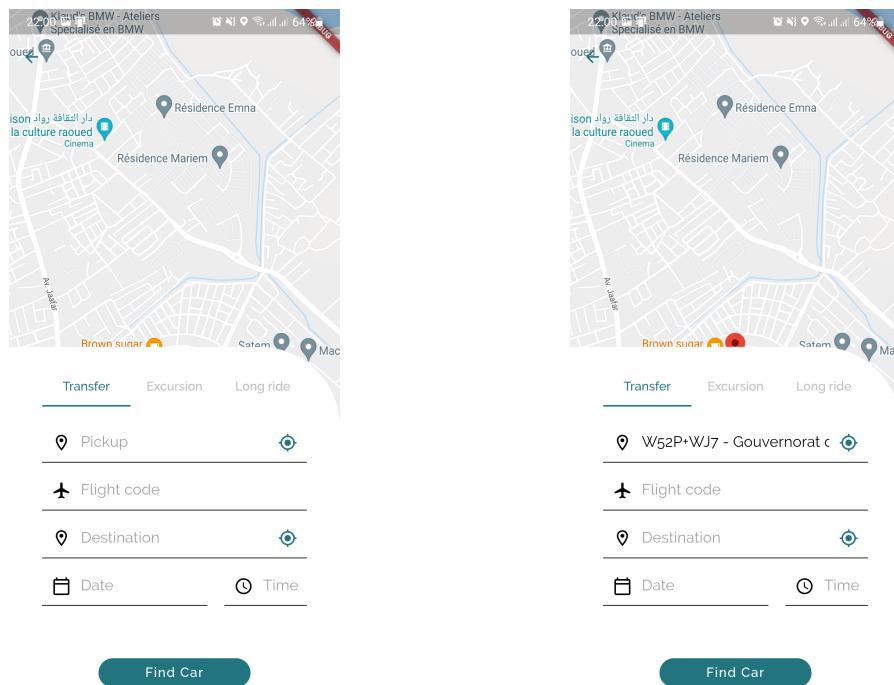


FIGURE 3.6 – Formulaire de demande de transfert.

FIGURE 3.7 – Utiliser le bouton «Localiser» pour choisir la position actuelle.

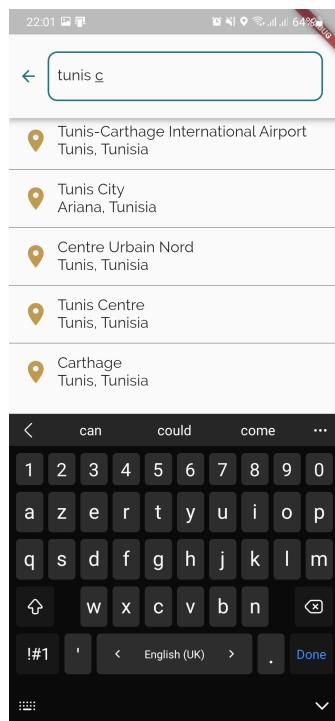


FIGURE 3.8 – Rechercher un emplacement à l'aide de Google Places.

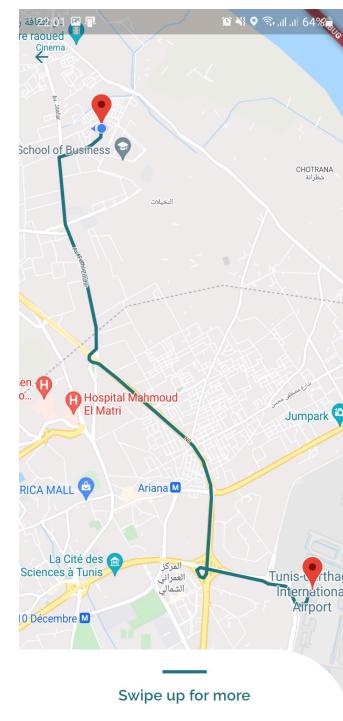


FIGURE 3.9 – Afficher la meilleure route entre le point de départ et le point d'arrivée.

3.6 Affichage des voitures disponibles

Après sélection des informations nécessaires par l'utilisateur, une recherche des voitures qui répondent aux critères de recherche choisis. Une fois une liste de voitures est prête, les voitures seront affichés. L'utilisateur peut appuyer sur une voiture pour découvrir ses caractéristiques et choisir ensuite de la louer ou continuer sa recherche.

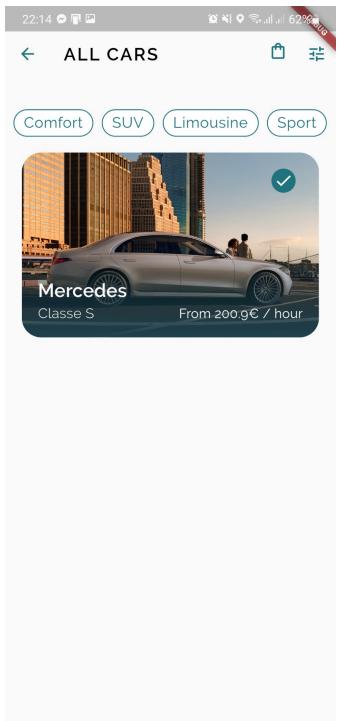


FIGURE 3.10 – Liste des voitures disponibles selon le point de départ sélectionné.

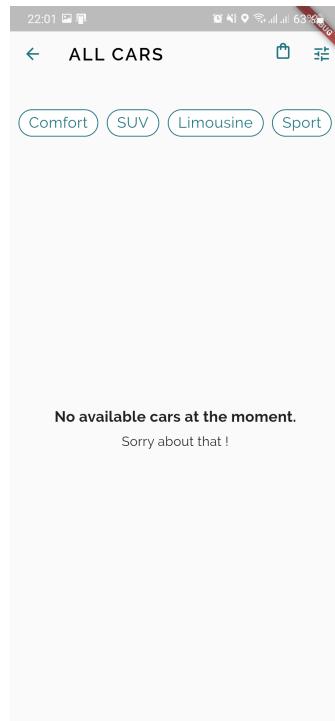


FIGURE 3.11 – Affichage si aucune voiture n'est disponible.

3.7 Signature numérique de contrat de location

Pour le cas de location de voitures, un contrat doit être signé entre le client et la société. Pour s'assurer de la signature du contrat, on a opté pour la solution de signature numérique à l'aide du service offert par DocuSign, qui se spécialise dans le domaine de la signature électronique et la gestion des transactions numériques pour faciliter les échanges et les validations électroniques des contrats et de documents.

Pour créer une réservation de location de voitures, il suffit d'appuyer sur le bouton de «Book now» dans la page de détails de la voiture choisie, et une réservation sera créée avec tous les détails : La position GPS, la date de récupération et retour de voiture et les informations de contact du client. Une fois ses informations sont enregistrées, un email contenant le contrat sera envoyé au client pour le signer, et l'identifiant de ce contrat sera enregistré avec les informations de la

réservation.

Dans l'application le client est redirigé vers la page de vérification de signature où il doit appuyer sur le bouton de vérification de l'état de contrat. Pour signer ce contrat, il suffit d'ouvrir le document envoyé à l'adresse email du client, le signer, et retourner vers l'application. Lors de l'appui sur le bouton «Verify Signature», une requête sera envoyée au back-end qui contactera les APIs de «DocuSign» pour vérifier l'état de signature du contrat à l'aide de l'identifiant du contrat fournis lors de la création de la réservation. Si l'état de contrat est «sent», le contrat n'est pas encore signé, et un message d'erreur sera affiché à l'utilisateur. Sinon si l'état du contrat est «completed», les contrat est, donc, signé et l'utilisateur est redirigé vers la page de paiement.

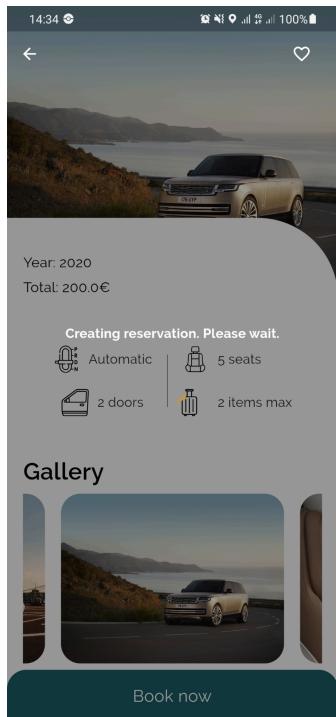


FIGURE 3.12 – Crédit de réservation.

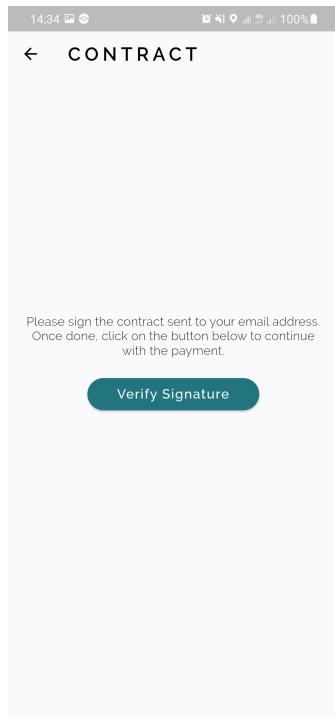


FIGURE 3.13 – Page de vérification de signature

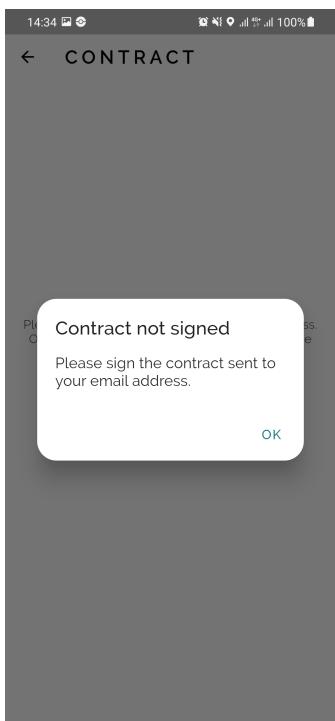


FIGURE 3.14 – Vérification de signature avec un document non signé.

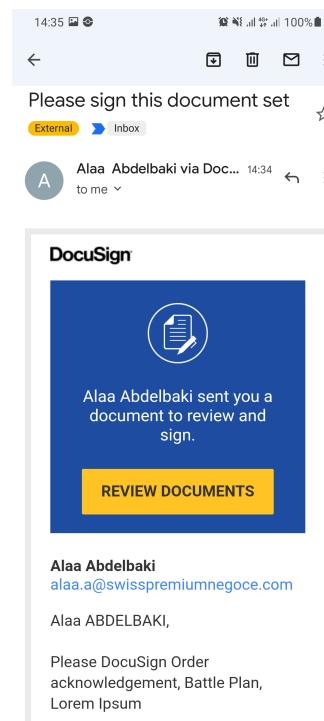


FIGURE 3.15 – Email contenant les documents à signer.



FIGURE 3.16 – Choix de signature.

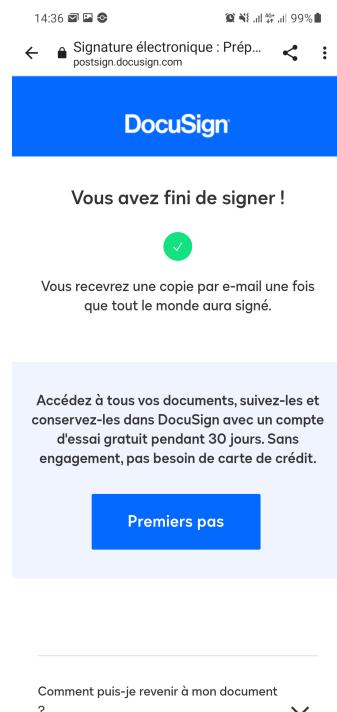


FIGURE 3.17 – confirmation de signature.

3.8 Paiement

Pour le paiement des services offert par SPN Cars, l'utilisateur utilisera «Apple Pay» s'il utilise un iPhone ou «Google Pay» s'il utilise un smartphone Android, qui sont, grâce à leurs haute disponibilités, le choix optimal pour effectuer des paiements en ligne avec son propre smartphone.

Pour effectuer un paiement avec l'application SPN Cars, il suffit de signer le contrat de location dans le cas de location de voitures, ou sélectionner une voiture dans le cas de demande de transfert, après la page de paiement avec le bouton de paiement selon le type de smartphone sera affiché. Lors de l'appui, L'interface des applications de paiement adéquates seront affichées pour continuer la transaction. Une fois terminé, l'utilisateur sera redirigé vers la page d'accueil si le paiement est effectué, ou un message d'erreur sera affiché si non.

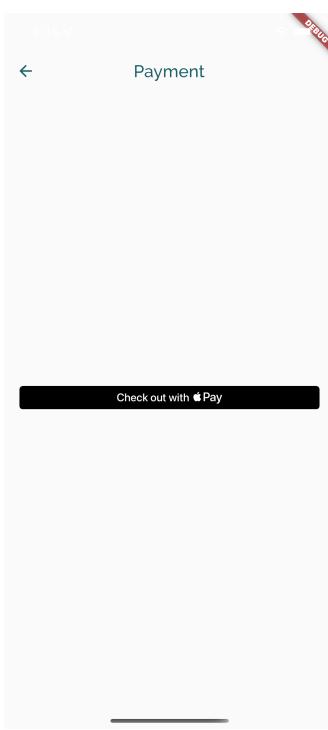


FIGURE 3.18 – Interface de paiement.

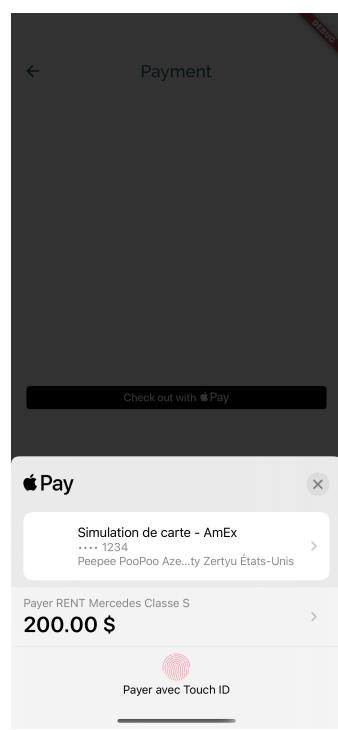


FIGURE 3.19 – Détails de paiement.

3.9 Localiser une voiture

Lorsqu'une réservation atteint la date de début et sera affichée dans la page d'accueil, le client peut appuyer sur la réservation et voir plus de détails sur sa réservation et sa voiture. Une fois appuyé sur le bouton «See more», le client sera redirigé vers une page qui affiche la position de sa voiture sur un plan. Cette position sera mise à jour en temps réel si le chauffeur

est en train de conduire la voiture.

La page permet aussi d'afficher les informations basiques du chauffeur avec une possibilité de le contacter soit par appel téléphonique soit par message.



FIGURE 3.20 – Affichage détails de la réservation.

3.10 Messagerie instantanée

Une fois l'utilisateur a appuyé sur le bouton pour contacter le chauffeur par messagerie instantannée, l'utilisateur est redirigé vers une page de messagerie avec le chauffeur où ils peuvent échanger des messages.

L'utilisateur peut aussi retourner vers sa liste de conversations et afficher ses anciens messages depuis la pages d'accueil.

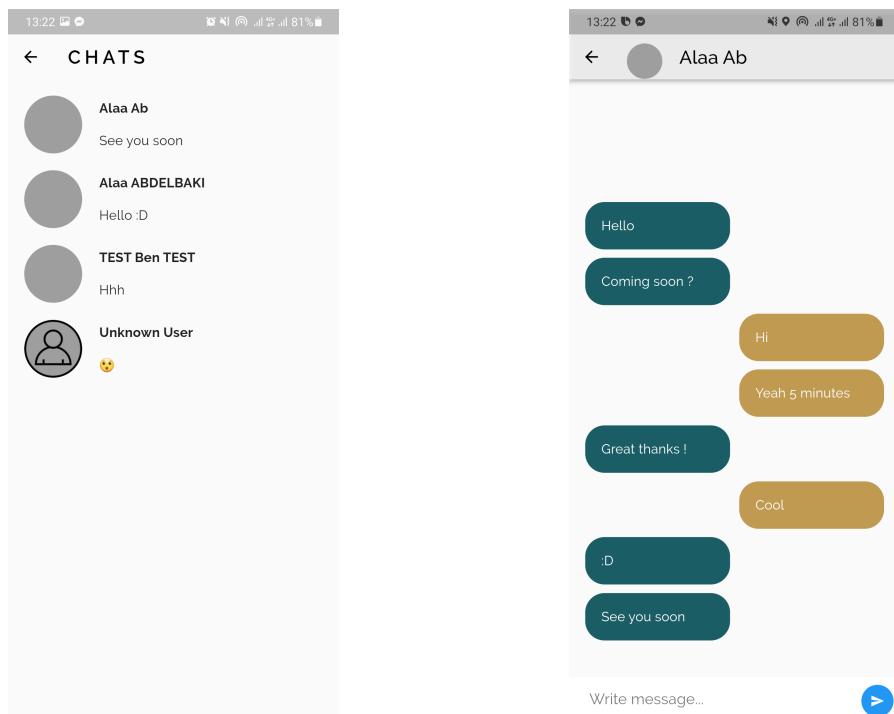


FIGURE 3.21 – Liste de messages de l’utilisateur.

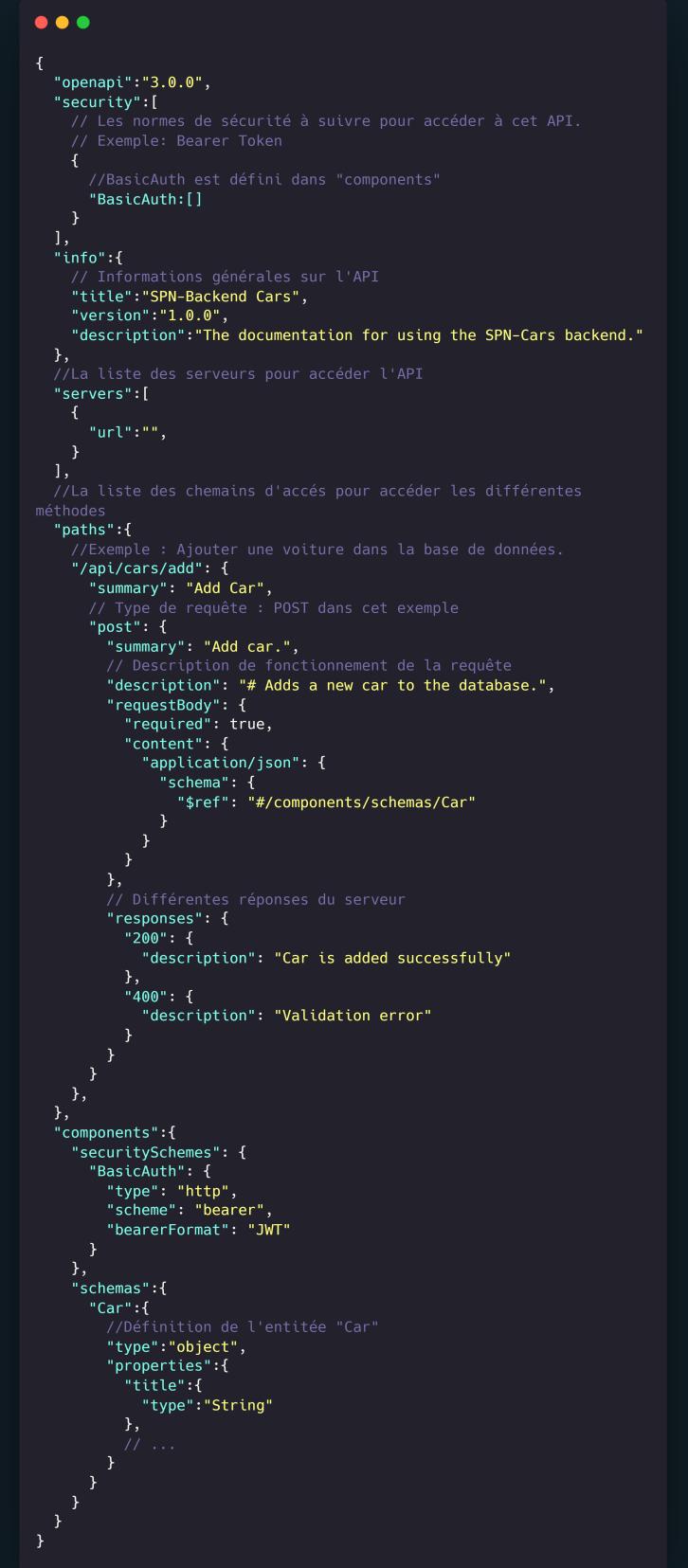
FIGURE 3.22 – Conversation avec utilisateur.

3.11 Documentation des API

La documentation des APIs est une étape nécessaire dans le développement de n’importe quelle application. Une documentation accessible et facilement exploitable est une condition préalable au développement des API. Il est important d’avoir une documentation toujours à jour quand le code/les fonctionnalités de l’API évoluent.

Et puisque le modèle de développement chez SPN est centré sur le travail collaboratif, il est très important d’avoir une documentation claire et facile à exploiter des APIs.

On va prendre un exemple de documentation de l’API de l’entité «Voiture» où on a suivi les spécification de Open Api pour créer la documentation de ses APIs.



```
{
  "openapi": "3.0.0",
  "security": [
    // Les normes de sécurité à suivre pour accéder à cet API.
    // Exemple: Bearer Token
    {
      //BasicAuth est défini dans "components"
      "BasicAuth": []
    }
  ],
  "info": {
    // Informations générales sur l'API
    "title": "SPN-Backend Cars",
    "version": "1.0.0",
    "description": "The documentation for using the SPN-Cars backend."
  },
  //La liste des serveurs pour accéder l'API
  "servers": [
    {
      "url": ""
    }
  ],
  //La liste des chemins d'accès pour accéder les différentes
  méthodes
  "paths": {
    //Exemple : Ajouter une voiture dans la base de données.
    "/api/cars/add": {
      "summary": "Add Car",
      // Type de requête : POST dans cet exemple
      "post": {
        "summary": "Add car.",
        // Description de fonctionnement de la requête
        "description": "# Adds a new car to the database.",
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Car"
              }
            }
          }
        },
        // Différentes réponses du serveur
        "responses": {
          "200": {
            "description": "Car is added successfully"
          },
          "400": {
            "description": "Validation error"
          }
        }
      },
      "components": {
        "securitySchemes": {
          "BasicAuth": {
            "type": "http",
            "scheme": "bearer",
            "bearerFormat": "JWT"
          }
        },
        "schemas": {
          "Car": {
            //Définition de l'entité "Car"
            "type": "object",
            "properties": {
              "title": {
                "type": "String"
              },
              // ...
            }
          }
        }
      }
    }
  }
}
```

FIGURE 3.23 – Schéma de documentation des API.

SPN-Backend Cars 1.0.0 OAS3

The documentation for using the SPN-Cars backend

Servers Authorize

default

POST /api/cars/add Add car. ✓

DELETE /api/cars/delete Delete car. ✓

POST /api/cars/find Find car. ✓

PUT /api/cars/update Updates car ✓

Schemas

Car >

ExtraServices >

This figure shows the API documentation for the SPN-Backend Cars service. At the top, there's a header with the service name, version (1.0.0), and an OAS3 badge. Below the header, a 'Servers' dropdown is set to 'https://localhost:3000' and an 'Authorize' button with a lock icon is available. The main content area is divided into sections: 'default' and 'Schemas'. The 'default' section lists four endpoints: 'POST /api/cars/add' (Add car), 'DELETE /api/cars/delete' (Delete car), 'POST /api/cars/find' (Find car), and 'PUT /api/cars/update' (Updates car). Each endpoint entry includes a status indicator (green for successful, red for failing) and a lock icon. The 'Schemas' section contains two items: 'Car' and 'ExtraServices', each with a corresponding navigation arrow.

FIGURE 3.24 – Documentation des API.

POST /api/cars/add Add car.

Adds a new car to the database.

Parameters

No parameters

Request body required

application/json

Example Value | **Schema**

```
{
  "title": "string",
  "description": "string",
  "commission": 0,
  "enabled": true,
  "registration": "string",
  "brand": "string",
  "model": "string",
  "year": 0,
  "nbSeats": 0,
  "nbLuggage": 0,
  "price": 0,
  "status": "PARKED",
  "nbDoors": 0,
  "manual": true,
  "energy": "DIESEL",
  "image": "string",
  "extraServices": [
    {
      "title": "string",
      "price": 0
    }
  ]
}
```

Responses

Code	Description	Links
200	Car is added successfully	No links
400	Validation error	No links

FIGURE 3.25 – Détails API ajout voiture.

Conclusion

Dans le présent rapport, on a dressé le bilan complet de ce travail qui se situe dans le cadre de mon projet de fin d'études. L'objectif principal de ce projet étant de concevoir et de développer une application mobile de location de voitures de luxes pour les clients de Swiss Premium Negoce, l'entreprise accueillante.

On a entamé ce rapport par une présentation du cadre général du projet où on a présenté l'entreprise accueillante et ses activités, puis on a présenté des différentes applications qui offrent des services de location de voitures et service taxi, ses applications on présenté des inconvénients. On a dégagé d'après ses inconvénients la problématique ayant engendré le besoin d'une telle application. Pour réaliser cette application, on a dû ensuite séparer ses besoins en besoins fonctionnels et non fonctionnels qui ont permis de commencer l'étape de conception, où on a décrit les différentes fonctionnalités nécessaires. Cette étape a permis d'identifier les technologies et outils à utiliser pour arriver à réaliser ce projet dans les délais choisis.

Une fois les outils et technologies choisies, on a passé au développement de l'application, où on a commencé par le développement des interfaces graphiques avec Flutter et leurs fonctionnalités avec Javascript et ExpressJS en coté serveur, au fin de développement de chaque interface et ses fonctionnalités, elle est testée pour dégager les différentes erreurs qui peuvent se déclencher et les corriger, jusqu'au développement de la dernière interface de l'application.

Ce projet fut une expérience enrichissante et fructueuse qui m'a permis d'acquérir de nouvelles compétences de grande valeur dans plusieurs domaines à la fois : Le développement web, d'avenir mobile et le design des interfaces graphiques et l'expérience utilisateur (UI/UX Design), en particulier le développement des applications coté serveur avec ExpressJS, développement des applications mobiles avec Flutter et la création des maquettes des interfaces utilisateur avec Adobe XD. Ainsi, ce stage qui représente une nouvelle expérience professionnelle,

m'a été bénéfique, il a été une nouvelle chance pour consolider mes compétences techniques et mettre en pratique mes compétences théoriques.

J'ai appris à être autonome, ouvert aux autres, et plus conscient de la vie professionnelle qui nécessite la ponctualité, le sérieux et le travail acharné. J'ai pris la responsabilité et j'ai développé l'application qui répondait aux besoins définis dans le cahier de charges, les exigences, et les délais.

Comme tout projet informatique, des contraintes ont été rencontrées et elles ont été surmontées tout au long de la réalisation du projet. Elles ont constitué un défi pour acquérir de nouvelles connaissances techniques et de développer les capacités opérationnelles. Ces défis concernent notamment la familiarisation avec les nouveaux frameworks et la maîtrise de certains outils techniques.

J'espère que le présent rapport soit suffisamment clair et structuré pour que le lecteur ait une idée précise et complète sur les différentes tâches que j'ai effectuées.

Bibliographie

- [1] *Swiss Premium Negoce*. URL : <https://www.swisspremiumnegoce.com>.
- [2] *Sixt Car hire*. URL : <https://www.sixt.com/>.
- [3] *Blacklane*. URL : <https://www.blacklane.com/>.
- [4] *Uber*. URL : <https://www.uber.com/>.
- [5] *Méthode en cascade*. URL : <https://blog-gestion-de-projet.com/modele-en-cascade/>.
- [6] *Flutter*. URL : <https://flutter.dev/>.
- [7] *ExpressJS*. URL : <https://expressjs.com/>.
- [8] *Mongodb*. URL : <https://www.mongodb.com/>.
- [9] *Firebase*. URL : <https://firebase.google.com/>.
- [10] *Visual Studio Code*. URL : <https://code.visualstudio.com/>.
- [11] *Microsoft*. URL : <https://www.microsoft.com>.
- [12] *Postman*. URL : <https://www.postman.com/>.
- [13] *Postman atteint 20 million d'utilisateurs*. URL : <https://venturebeat.com/2022/04/19/postman-api-platform-hits-20m-users-helps-drive-the-api-economy/>.
- [14] *Adobe XD*. URL : <https://www.adobe.com/products/xd.html>.
- [15] *Git*. URL : <https://git-scm.com/>.
- [16] *Swagger*. URL : <https://swagger.io/>.



ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siège Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : 1-2 rue André Ampère - 2083 - Pôle Technologique - El Ghazala - Tél +216 70 250 000 - Fax +216 70 685454

