



Haute école d'ingénierie et d'architecture Fribourg  
Hochschule für Technik und Architektur Freiburg

# Projet Semestre

5

---

APPLICATION DE  
COVOITURAGE



Rédigé par

Nestor Peña López  
Patrick Audriaz

Superviseur

Sandy Ingram  
Philippe Joye

Version

1.0

Copyright © 2018 Nestor Peña & Patrick Audriaz

HAUTE ÉCOLE D'INGÉNIEURS DE FRIBOURG

Latex template - Mathias Legrand "The Legrand Orange Book" (licence : CC BY-NC-SA 3.0)

Images et banners sous licence Creative Commons



# 1. Documents

## 1.1 Table de versions

Version	Date	Remarques
0.1	01.10.18	Création et publication du document
0.2	30.10.18	Analyse technologique
0.3	28.11.18	Analyse UX et Marketing
0.4	12.12.18	Conception
0.5	09.01.19	Réalisation
1.0	31.01.19	Version Finale

## 1.2 Organisation des documents

Tous les documents concernant le projet sont déposés sur la forge à l'adresse :

<https://forge.hefr.ch/projects/application-de-co-voiturage-villageois>

Tous les fichiers du projet sont déposés sur le dépôt Git disponible à l'adresse suivante :

<https://gitlab.forge.hefr.ch/patrick.audriaz/projetsem5-peña-audriaz>

Le code source du projet est déposé sur le dépôt Git disponible à l'adresse suivante :

<https://gitlab.forge.hefr.ch/patrick.audriaz/projetsem5-peña-audriaz/tree/master/sonnaz-go-final>

L'application est accessible via l'URL publique :

<https://sonnazgo.firebaseio.com/>





# Table des matières

<b>1</b>	<b>Documents</b>	<b>3</b>
1.1	Table de versions	3
1.2	Organisation des documents	3
<b>I</b>	<b>Introduction</b>	
<b>2</b>	<b>Introduction</b>	<b>10</b>
2.1	Introduction	10
2.2	Contexte	10
2.3	Plan	11
<b>3</b>	<b>Cahier des charges</b>	<b>12</b>
3.1	Objectifs principaux	12
3.2	Objectifs secondaires	13
<b>II</b>	<b>Analyse technologique</b>	
<b>4</b>	<b>Synthèse des souhaits et besoins des mandants</b>	<b>15</b>
<b>5</b>	<b>Analyse des technologies web et mobiles</b>	<b>16</b>
5.1	JavaScript	16
5.2	Backend	17
5.2.1	Node JS	18
5.3	Frontend	19
5.3.1	HTML5	19
5.3.2	CSS3	19
5.3.3	Écosystème Javascript	20
<b>6</b>	<b>Comparaison des technologies</b>	<b>22</b>
6.1	Site web responsive	22
6.2	Application mobile native	22
6.3	Progressive Web App (PWA)	23
6.4	Fonctionnement des PWA	24
6.5	Analyse multicritère	25
<b>7</b>	<b>Comparaison des Frameworks JavaScript</b>	<b>27</b>
7.1	Qu'est-ce que c'est un framework?	27
7.2	Differences entre un framework et une librairie?	27
7.3	Comparaison des deux frameworks et une librairie	28
7.3.1	Angular.js	28
7.3.2	React.js	28
7.3.3	Vue.js	29

7.4	<b>Analyse multicritère</b>	29
<b>III</b>	<b>Analyse besoins UX et marketing</b>	
<b>8</b>	<b>Besoins des utilisateurs</b>	32
8.1	<b>Analyse des besoins</b>	32
8.1.1	WAAD . . . . .	32
8.1.2	Rôles . . . . .	33
8.2	<b>Problèmes potentiels qui peuvent être rencontrés</b>	34
<b>9</b>	<b>Analyse de la concurrence</b>	35
9.0.1	Utilité . . . . .	37
9.0.2	Utilisabilité . . . . .	37
9.0.3	Impact émotionnel . . . . .	39
<b>10</b>	<b>Identité visuelle et promotion</b>	40
<b>10.1</b>	<b>Material design vs Flat design</b>	40
10.1.1	Principes du flat design . . . . .	40
<b>10.2</b>	<b>Principes du material design</b>	41
<b>10.3</b>	<b>Framework CSS</b>	42
<b>10.4</b>	<b>Identité visuelle</b>	43
10.4.1	Nom et valeurs . . . . .	44
10.4.2	Logo et formes . . . . .	44
10.4.3	Identité visuelle finale . . . . .	48
<b>10.5</b>	<b>Motivation à l'utilisation</b>	48
<b>10.6</b>	<b>Communiqué de presse</b>	49
10.7	<b>Idées de promotion locale</b>	49
<b>IV</b>	<b>Conception</b>	
<b>11</b>	<b>Modélisation des processus</b>	52
11.1	<b>Business process model and notation (BPMN)</b>	52
<b>12</b>	<b>Modélisation UML</b>	54
12.1	<b>Cas d'utilisation</b>	54
<b>13</b>	<b>Maquettes</b>	55
13.1	<b>Introduction</b>	55
13.2	<b>Enregistrement et connexion</b>	55
13.3	<b>Guide d'utilisation</b>	56
13.4	<b>Page d'accueil conducteur</b>	57
13.5	<b>Annonce d'un trajet (conducteur)</b>	58
13.6	<b>Page d'accueil passager</b>	58
13.7	<b>Réservation d'un trajet (passager)</b>	59
13.8	<b>Modifications à tenir compte dans la réalisation</b>	59
<b>V</b>	<b>Réalisation</b>	
<b>14</b>	<b>Notions essentielles de React</b>	61
14.1	<b>DOM virtuel</b>	61
14.2	<b>JSX</b>	62
14.3	<b>Composants</b>	62
14.3.1	Composants stateless (ou fonctionnel) . . . . .	62
14.3.2	Composants stateful (ou à état) . . . . .	63

<b>14.4</b>	<b>Tendance déclarative</b>	<b>63</b>
<b>15</b>	<b>Mise en place de l'environnement React</b>	<b>64</b>
<b>15.1</b>	<b>Installation et utilisation de React</b>	<b>64</b>
15.1.1	Prérequis . . . . .	64
15.1.2	create-react-app . . . . .	64
<b>15.2</b>	<b>Routing</b>	<b>65</b>
15.2.1	Mise en place . . . . .	65
15.2.2	BrowserRouter . . . . .	66
15.2.3	Route . . . . .	66
15.2.4	Link . . . . .	67
<b>15.3</b>	<b>Service Worker</b>	<b>67</b>
15.3.1	Fonctionnement hors ligne . . . . .	68
15.3.2	Customisation . . . . .	68
<b>15.4</b>	<b>Hosting Firebase</b>	<b>69</b>
<b>16</b>	<b>Extraits de code</b>	<b>70</b>
<b>16.1</b>	<b>Dossiers actions, selectors, reducers et store</b>	<b>70</b>
<b>16.2</b>	<b>index.js</b>	<b>70</b>
<b>16.3</b>	<b>AddRidePage5.js</b>	<b>70</b>
<b>16.4</b>	<b>Code réutilisé dans plusieurs composants</b>	<b>71</b>
<b>16.5</b>	<b>AddRideItem.js</b>	<b>71</b>
<b>16.6</b>	<b>DirectionChoice.js</b>	<b>72</b>
<b>16.7</b>	<b>RidesList.js</b>	<b>72</b>
<b>17</b>	<b>GUI et CSS</b>	<b>73</b>
<b>17.1</b>	<b>Style</b>	<b>73</b>
17.1.1	Framework CSS . . . . .	73
17.1.2	SASS . . . . .	74
17.1.3	Personnaliser les framework CSS . . . . .	74
<b>17.2</b>	<b>Responsive</b>	<b>75</b>
17.2.1	Grid Materialize . . . . .	75
17.2.2	Media Queries . . . . .	76
<b>18</b>	<b>Redux</b>	<b>77</b>
<b>18.1</b>	<b>Pourquoi avons-nous besoin de Redux ?</b>	<b>77</b>
<b>18.2</b>	<b>Comment mettre en place Redux ?</b>	<b>78</b>
18.2.1	Étape 1 : Créer un "Store" . . . . .	80
18.2.2	Étape 2 : Créer un "Reducer" . . . . .	80
18.2.3	Étape 3 : Utiliser la méthode "Subscribe" . . . . .	81
18.2.4	Étape 4 : Utiliser la méthode "Dispatch" . . . . .	81
<b>19</b>	<b>Carte interactive</b>	<b>82</b>
<b>19.1</b>	<b>Introduction</b>	<b>82</b>
<b>19.2</b>	<b>Customisation de Mapbox</b>	<b>83</b>
<b>19.3</b>	<b>Implémentation de Mapbox</b>	<b>83</b>
<b>19.4</b>	<b>Dépendance</b>	<b>84</b>
<b>19.5</b>	<b>License</b>	<b>84</b>

<b>VI</b>	<b>Test et validation</b>	
<b>20</b>	<b>Test empirique</b>	86
20.1	Test de 5 secondes	86
20.2	Test d'utilisabilité	88
<b>VII</b>	<b>Conclusions</b>	
<b>21</b>	<b>Conclusions du projet</b>	93
21.1	Validation des objectifs	93
21.2	Problèmes rencontrés et solutions apportées	93
21.2.1	Redux	93
21.2.2	Restructuration du projet	94
21.2.3	Cohérence des labels	95
21.2.4	Organisation	95
21.2.5	Petits problèmes UI persistants	95
21.3	Perspectives Futures	95
21.4	Remerciements	95
21.5	Conclusion du projet	96
21.6	Conclusion personnelle	96
<b>22</b>	<b>Conclusions du document</b>	97
22.1	Licences	97
22.2	Déclaration d'honneur	98
	Glossaire	103
	Acronymes	104
	Table de figures	107
22.3	Planning	108
22.4	Tâches heures par heures	109



# Introduction

<b>2</b>	<b>Introduction</b>	<b>10</b>
2.1	Introduction	
2.2	Contexte	
2.3	Plan	
<b>3</b>	<b>Cahier des charges</b>	<b>12</b>
3.1	Objectifs principaux	
3.2	Objectifs secondaires	



## 2. Introduction

### 2.1 Introduction

Nous, Nestor Peña Lopez et Patrick Audriaz, nous sommes vu attribuer un projet pour notre cinquième semestre en filière télécommunication, orientation internet et communication, à l'école d'ingénierie et d'architecture de Fribourg. Il vise à développer plusieurs compétences, dont la gestion de projet, les présentations orales et la rédaction de rapport. Un projet concret sera donc réalisé au cours de ce semestre par nos soins avec l'assistance et la supervision de deux professeurs responsables, Madame Sandy Ingram et Monsieur Philippe Joye.

### 2.2 Contexte

Trois habitants de la commune de la Sonnaz et accessoirement également parents de plus ou moins jeunes enfants (école primaire à cycle d'orientation) souhaitent développer, en collaboration avec l'école d'ingénierie et d'architecture de Fribourg, une application de covoiturage local.

Leurs motivations personnelles sont de créer un réseau de transport privé pour pallier le manque de transport en commun dans la région et faire des économies d'argent et du temps. L'objectif est principalement de faciliter l'accès aux différentes gares et arrêts de bus situés en périphérie de la commune de la Sonnaz. Les mandants étant tous parents, ils sont constamment exposés au problème qu'est d'amener et d'aller chercher leur enfant à ces différents lieux et au besoin d'avoir un système fiable sur lequel ils peuvent se reposer afin de faciliter le covoiturage, une pratique qu'ils effectuent déjà, mais en s'organisant oralement.

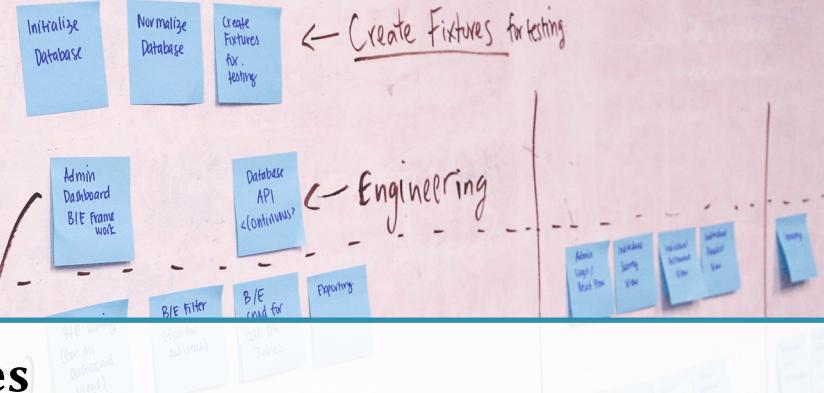
L'application est, selon les mandants, nécessaire et bénéfique à l'ensemble de la commune. La communauté pourra bénéficier d'un accès facilité et donc de potentiellement encourager une nouvelle population de venir s'y installer en améliorant indirectement la qualité de vie. Le souhait de réduire l'empreinte CO<sub>2</sub> ainsi que de limiter les embouteillages dû au sur engorgement des routes est également très marqué.

L'utilisation de l'application sera ouverte à tous les membres de la commune, toute personne intéressée pourra soit profiter du service en tant qui passager ou en tant que conducteur s'il répond à quelques critères préalables. Ils souhaitent créer une application qui puisse évoluer avec son temps, s'adapter aux différents besoins de transport de manière simple et flexible. Le service doit rester totalement gratuit, basé sur l'entraide et la fidélisation de ses membres.

### **2.3 Plan**

Une application doit être créée. Utilisable sur plateforme mobile et fixe. Nous allons ci-après rédiger un cahier des charges afin de mettre en évidence les aspects sur lesquels notre projet de semestre va se focaliser, nous réaliserons également un planning qui va nous permettre d'atteindre les objectifs décrits à continuation. Une fois que nos objectifs et notre planning seront validés, nous commencerons une analyse en profondeur des besoins, de la concurrence et des technologies. Cette analyse pourra nous permettre d'enchaîner sur une partie conception en ayant toutes les clés en main pour réaliser un travail cohérent, réaliste, conforme au cahier des charges et servant de base pour la suite du projet.

# WEEK 4 : Design



# WEEK 5 : Design

## 3. Cahier des charges

### 3.1 Objectifs principaux

#### 1. Analyse technologique

- Synthèse de besoins des mandants pour appuyer nos critères d'analyse
- Quelles sont les technologies modernes permettant de créer une application web et mobile (isomorphe) efficient et robuste ?
- Comparaison entre les progressives web app et les applications mobiles natives
- Comparaison des différentes bibliothèques JavaScript

**Livrable :** Analyse comparative multicritère détaillée

#### 2. Analyse besoins UX et marketing

- Analyse des besoins des utilisateurs pour la conception et l'interaction de l'objectif 3
- Étude de la concurrence, des potentielles alternatives
- Création d'une identité visuelle comprenant une charte graphique, un nom et un logo en accord avec le projet
- Création d'un communiqué de presse pour présenter le projet à diverses parties externes
- Réflexion sur des moyens de promotion à l'échelle de la commune

**Livrables :** Maquette interactive, communiqué de presse et identité visuelle

#### 3. Prototype front-end de l'application de covoiturage couvrant des fonctions prioritaires suffisantes à la présentation de l'application à des acteurs externes.

- Prise en main de la technologie choisie à l'objectif 1 et sur les PWA
- Conception de l'interaction au moyen de maquettes et de diagrammes Unified Modeling Language (UML) des fonctionnalités
- Réalisation d'une version intermédiaire (prototype 1) livrable comprenant les fonctionnalités suivantes :
  - Réserver un trajet (passager)
  - Annoncer un trajet (conducteur)

**Livrables :** Prototype 1

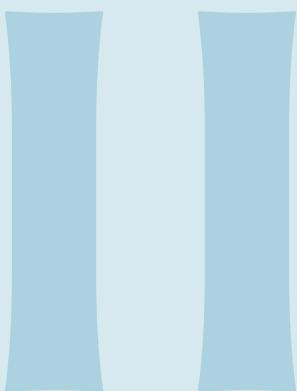
### **3.2 Objectifs secondaires**

#### **1. Étoffer le front-end pour y intégrer d'autres fonctionnalités.**

→ Réalisation d'un prototype 2 comprenant les fonctionnalités suivantes :

- Login/logout/inscription
- Système de "badges" (conducteurs)
- Aide/tutoriel
- Profil de l'utilisateur et paramètre
- Historique des trajets
- Partage de la course
- Notifications
- Signalisation d'un imprévu
- Signalement d'un accident
- Signalisation d'un conducteur/passager
- Oubli d'un objet dans le véhicule
- Notation de la course

**Livrable :** Prototype 2



# Analyse technologique

<b>4</b>	<b>Synthèse des souhaits et besoins des mandants . . . . .</b>	<b>15</b>
<b>5</b>	<b>Analyse des technologies web et mobiles</b>	<b>16</b>
5.1	JavaScript	
5.2	Backend	
5.3	Frontend	
<b>6</b>	<b>Comparaison des technologies . . . . .</b>	<b>22</b>
6.1	Site web responsive	
6.2	Application mobile native	
6.3	Progressive Web App (PWA)	
6.4	Fonctionnement des PWA	
6.5	Analyse multicritère	
<b>7</b>	<b>Comparaison des Frameworks JavaScript</b>	<b>27</b>
7.1	Qu'est-ce que c'est un framework?	
7.2	Différences entre un framework et une librairie?	
7.3	Comparaison des deux frameworks et une librairie	
7.4	Analyse multicritère	



## 4. Synthèse des souhaits et besoins des mandants

Suite à la réunion initiale avec les mandants, nous avons pu établir une liste de besoin et de souhaits initiaux. Nous nous en servirons pour l'analyse technologique afin de choisir les outils adaptés à leurs attentes.

Le but est de choisir des technologies utilisées dans le développement web (frontend) moderne et de choisir des outils adaptés afin de faciliter la partie réalisation de ce projet de semestre. Certains des besoins ci-dessous n'ont aucune influence sur le développement tandis que d'autres sont très manifestement dépendants de certaines technologies afin de les mettre en œuvre de manière optimale. Avant toute chose, il était du souhait de Madame Ingram de se diriger vers des applications de type Progressive Web App (PWA) pour ce projet de semestre. Il est aussi recommandé d'utiliser la librairies/frameworks ReactJS pour la partie programmation frontend. Nous allons pouvoir voir grâce à l'analyse suivante si ce genre d'approche est adapté à un projet tel que le nôtre ou si d'autres technologies sont plus préférable.

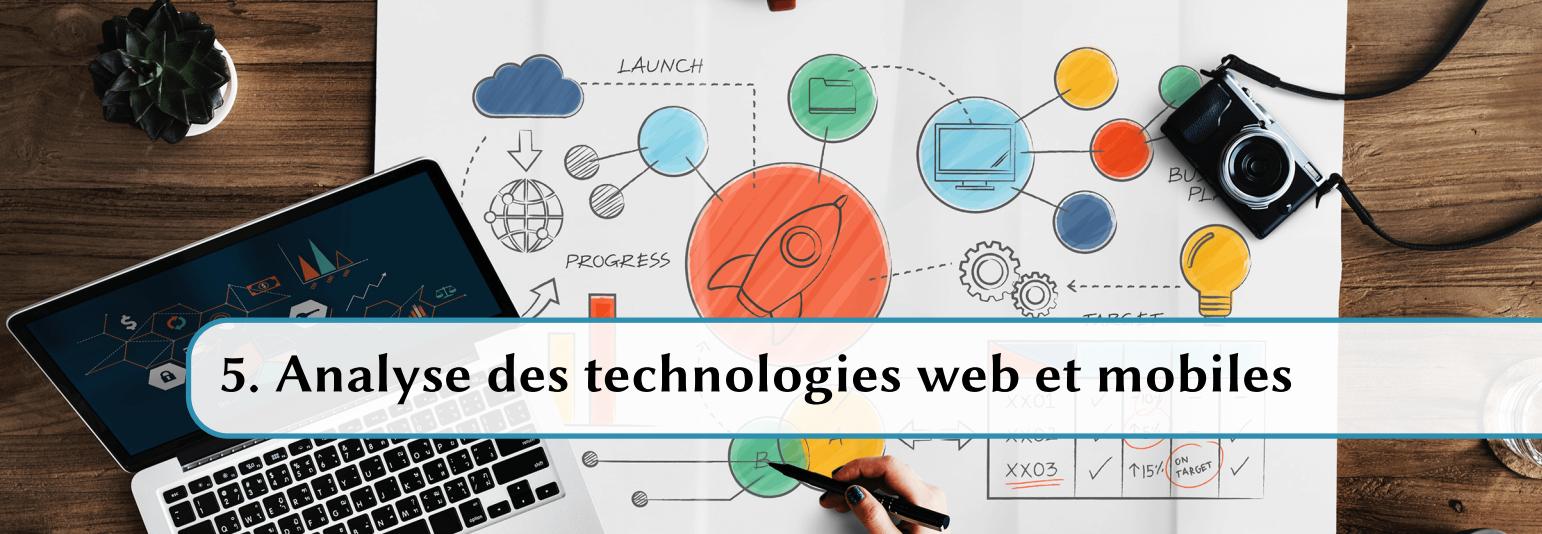
Bien sûr, ceci n'est qu'une étude initiale préalable utilisant les informations que nous avons à notre disposition pour faire un choix de technologie. Une étude plus poussée des besoins propres à une application de covoiturage sera effectuée dans la partie III intitulée "**Analyse besoins UX et marketing**".

### Besoins et souhaits influençant les choix des technologies de développement :

- Multiplateforme (site web accessible sur ordinateur et application ou site web sur smartphone)
- Minimaliste, intuitif, conviviale (interface élégante et moderne, propre, flexible, fluide et performant, responsive)
- Identité visuelle (possibilité de créer un côté esthétique poussé)
- Géolocalisation (accès à la position des utilisateurs en temps réel)

### Besoins et souhaits n'ayant aucun impact sur l'analyse technologique :

- Gratuité (application totalement gratuite, pas de frais de déplacement ni de moyen de paiement aux chauffeurs)
- Interne à la commune (contrôle des membres par l'administrateur)
- Arrêts prédéterminés (définis au préalable)
- Contrôle utilisateur (traçabilité des trajets, des évènements)
- Fidélisation (motiver les conducteurs et les passagers à employer le service via une gamification, un système de badges...)
- Pérennisation (créer une communauté autour du service pour en assurer la longe vie)
- Documentation de présentation (flyers, communiqué de presse pour des acteurs externes)
- Administration (ajout d'arrêts, gestion des conducteurs, contrôle des utilisateurs ...)



## 5. Analyse des technologies web et mobiles

Nous allons ici nous concentrer sur l'aspect frontends du développement web, mais ce n'est pas pour autant que nous n'allons pas commencer par expliquer des concepts essentiels au développement web moderne dont, le côté backends entre autres. Nous allons commencer par citer des notions de base avant de montrer une architecture type et de décrire chaque entité qui la compose.

Cette analyse sera valable que pour une durée limitée, tout évolue très vite, de nouvelles technologies sortent chaque année et modifient potentiellement tout le workflow établit.

### 5.1 JavaScript

La suivante définition<sup>1</sup> est donnée par Wikipedia, nous allons découvrir plus en détail pourquoi elle n'est pas tout à fait correcte actuellement.

“ JavaScript est un langage de programmation de scripts côté client orienté objet principalement employé dans les pages web interactives. ”

*Wikipedia*

Son intérêt est de pouvoir étendre les possibilités offertes par HTML, il permet de le rendre interactif. JavaScript est basé sur les évènements. Son utilisation dans une page web peut être par exemple : ouvrir des pop-ups, faire apparaître, disparaître, bouger des éléments de la page, mettre à jour des éléments sans recharger la page ou simplement gérer des actions effectuées par l'utilisateur (click, clavier, souris, etc.).

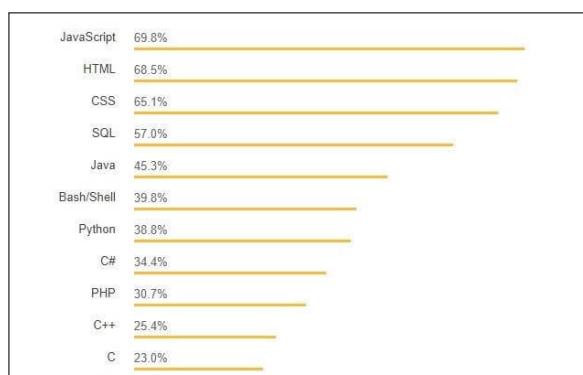


FIGURE 5.1 – Popularité des langages sur StackOverflow

1. <https://fr.wikipedia.org/wiki/JavaScript>

Comme vu dans la FIGURE 5.1<sup>2</sup> JavaScript est aussi un des langages de programmation le plus populaire au monde actuellement. Choisi par la plupart des programmeurs par sa simplicité et son évolution fulgurante et constante. Comment nous pouvons constater dans l'image ci-dessous, JavaScript est pour la sixième année consécutive, le langage préféré des programmeurs selon un sondage annuel du site StackOverflow.

Ça vaut peut-être la peine de noter que Java et JavaScript ne sont pas du tout la même chose et que, si on exclut le fait que ce sont des langages de programmation, il n'existe pas plus de similarité entre eux qu'entre " tram " et " trampoline ".

JavaScript, qui été considéré par les programmeurs d'il y a 10 ans comme un " Toy language " est aujourd'hui capable non seulement de fonctionner du coté client interprété par le " JavaScript Engine " de notre navigateur préféré, mais aussi du côté serveur avec l'introduction du Node JS ce qui permet d'exécuter des requêtes comme dans PHP. Aussi grâce au travail des grandes compagnies telles que Facebook et Google qui ont développé des frameworks comme React et Angular respectivement, JavaScript peut faire beaucoup plus : nous pouvons néanmoins créer des applications mobiles et de bureau, des jeux, des applications de Réalité virtuelle (VR) et même programmer les comportements des robots.

## 5.2 Backend

Le backend qualifie la "partie immergée" d'une application web, elle est invisible pour l'utilisateur, mais représente la clé de voûte de celle-ci. Une belle interface est une coquille vide sans un backend solide. Il est composé de trois piliers essentiels à un bon fonctionnement : un serveur d'hébergement (endroit où est stocké l'ensemble de l'application), une application web en elle-même et une base de données (ou l'on stocke les données de l'application).

Avant NodeJS (plus de détails ci-dessous), les développeurs utilisaient JavaScript uniquement du côté client, c'est le navigateur web qui se chargeait d'exécuter le code et de rendre par extension la page interactive. Il y avait, dans un sens, deux couches d'interface (UI layer), une dans le navigateur et une sur le serveur qui générait un payload pour le navigateur. Il y avait que très peu de contrôle sur la couche UI côté serveur et les choix étaient par conséquent grandement dépendants des ingénieurs backend qui ne prenaient pas forcément en considération les besoins du frontend. Schémas ci-dessous<sup>3 4</sup>.

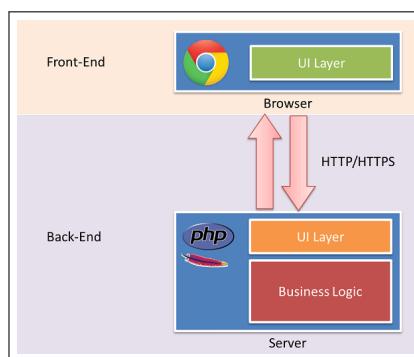


FIGURE 5.2 – Architecture sans Node.JS où tout le travail frontend se faisait dans le navigateur

2. <https://insights.stackoverflow.com/survey/2018>

3. <https://humanwhocodes.com/images/wp-content/uploads/2013/10/nodejs1.png>

4. [https://user.oc-static.com/files/421001\\_422000/421108.png](https://user.oc-static.com/files/421001_422000/421108.png)

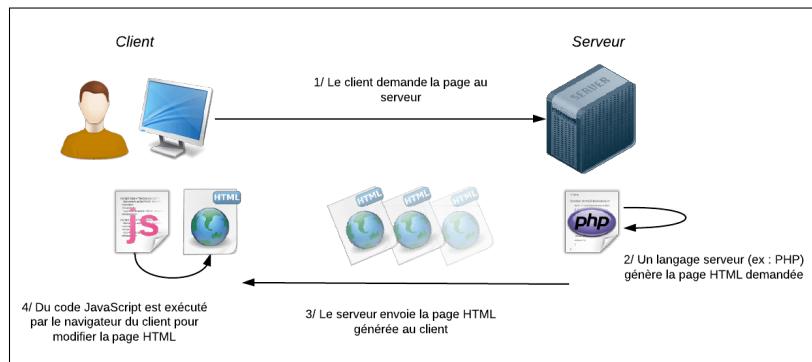


FIGURE 5.3 – Communication Client / serveur "traditionnelle" avec PHP sur le server et JavaScript chez le client

### 5.2.1 Node JS

NodeJS vient résoudre ce "problème" en proposant un environnement bas niveau JavaScript côté serveur. Il vient remplacer les langages dit "serveur" comme Hypertext Preprocessor (PHP). L'avantage face à ceux-ci est que, contrairement aux autres plateformes, il a une approche non-bloquante, il permet de gérer des actions et des événements de manière asynchrone (traite plusieurs requêtes concurrentes en même temps, pas de temps d'attente, on peut faire des choses en arrière-plan) et est, comme JavaScript, basé sur des événements. Il possède également un gestionnaire de paquets (NPM) permettant de télécharger des librairies afin d'en accroître les possibilités.

NodeJS permet de libérer la partie du backend consacrée à l'interface (UI) du reste du backend. Il redonne le pouvoir aux ingénieurs frontend. Il a débloqué l'utilisation de JavaScript en dehors du browser. Schémas ci-dessous<sup>5</sup> <sup>6</sup>

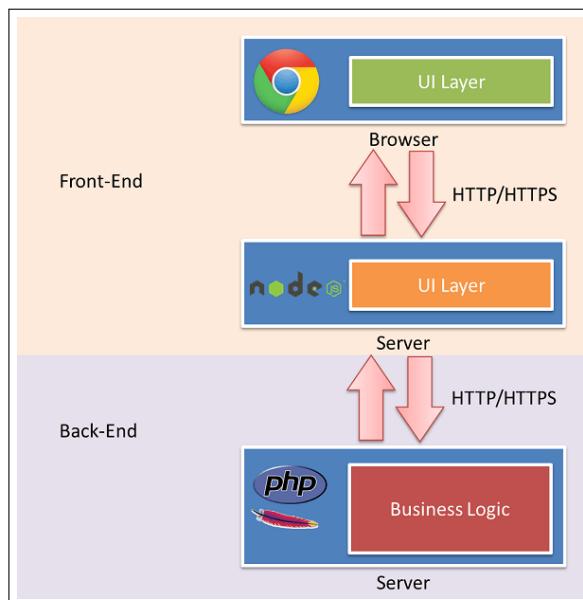


FIGURE 5.4 – Architecture type avec Node.JS

5. <https://humanwhocodes.com/images/wp-content/uploads/2013/10/nodejs2.png>

6. [https://user.oc-static.com/files/421001\\_422000/421109.png](https://user.oc-static.com/files/421001_422000/421109.png)

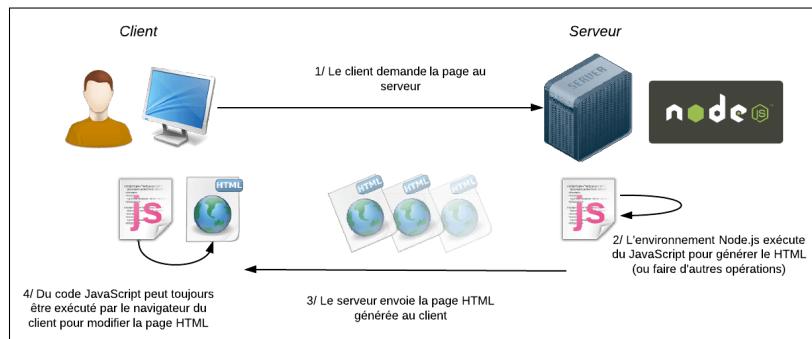


FIGURE 5.5 – Serveur utilisant Node.JS

### 5.3 Frontend

Le frontend est toute la partie visible de l'application web, tous les éléments avec lesquels l'utilisateur peut interagir, toute la partie destinée à être manipulée. Les principaux domaines touchés par le frontend et donc par extension, les compétences requises pour faire une application agréable sont la conception d'interfaces graphiques, le design et le développement. Pour ce dernier point, les différents langages suivants sont utilisés :

- **HTML** : langage de balises pour représenter des pages web, il permet de structurer et de mettre en forme le contenu des pages
- **CSS** : permet de gérer l'apparence du HTML, en décrit la présentation en permettant d'y ajouter des styles
- **JavaScript** : qui est venu s'ajouter à l'arsenal du développeur frontend comme décrit ci-dessus

De nos jours, il n'est plus suffisant de connaître des bases d'HTML, de CSS et de JavaScript pour construire une application répondant aux exigences du web moderne. HTML et CSS restent cependant une fondation dont il est nécessaire d'avoir une compréhension poussée pour venir y greffer des éléments au moyen de JavaScript.

#### 5.3.1 HTML5

Successeur de HTML 4.0.1, HTML5 est la dernière version en date de HTML, elle offre principalement davantage de flexibilité pour une navigation sur appareil mobile, une amélioration de la comptabilité et de l'expérience image / vidéo (support de nouveaux formats), l'exécution native de JavaScript, de nouveaux types de contrôle de formulaires, de nouvelles balises (header, article, nav, footer, menu, etc). Schéma ci-dessous<sup>7</sup>.

#### 5.3.2 CSS3

Successeur de CSS 2.1, CSS3 est la dernière version en date de CSS et offre de nombreuses fonctionnalités et additions. Par exemple l'ajout de côtés arrondis, d'ombres, de dégradés, de transitions, d'animations et de nouvelles mises en pages (multicollonne, grille, boîte flexible, etc.).

CSS possède de nombreux framework permettant de "mâcher" le travail au développeur. Ce sont des collections de composants standardisés qui sont prêt à l'emploi, cela permet de rendre le travail plus rapide, facile et standard. Les principaux utilisés de nos jours sont : Materialize, Bootstrap, Foundation, Bulma, Sematic UI, etc.

<sup>7</sup>. <https://www.hostinger.fr/tutoriels/wp-content/uploads/sites/9/2017/04/differences-between-html-and-html5.png>

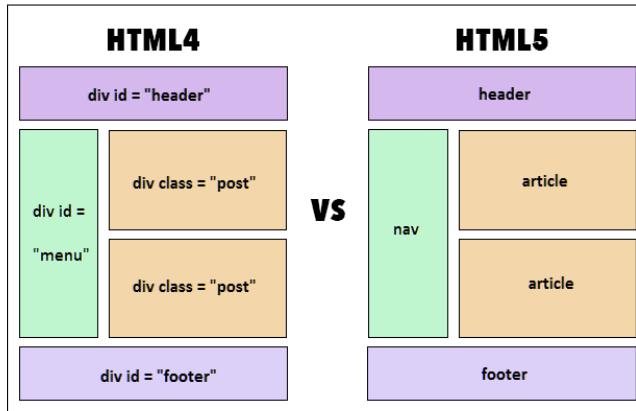


FIGURE 5.6 – Différences de structure et de balises entre l'ancien et le nouveau HTML

```
.square {
    background: black;
    width: 100px;
    height: 100px;
    -webkit-animation: pulsing 2s infinite;
    animation: pulsing 2s infinite;
}
```

FIGURE 5.7 – Exemple d'animation CSS faisant "pulser" un carrés

### 5.3.3 Écosystème Javascript

NodeJS a permis, dans un sens, un renouveau de JavaScript grâce à son gestionnaire de package NPM. Cela a débloqué aux développeurs frontend de nombreux outils alors pas disponibles comme des outils en ligne de commande pour JavaScript, des Integrated development environment (IDE), des outils de construction, d'empaquetage et bien d'autres. Bien sûr, il existe de nos jours de nombreux autres gestionnaires de paquets, mais NPM reste le plus fourni et soutenu par les développeurs.

Des outils de construction (build tools) comme Gulp sont utilisés comme "task runner" (c'est-à-dire qu'ils effectuent des tâches pour nous). Ils sont là pour effectuer automatiquement des actions nécessaires à la construction d'une application web comme la création d'un serveur web local, le rafraîchissement automatique du navigateur quand le fichier est modifié et l'utilisation de préprocesseurs (traducteur de nouvelle syntaxe en CSS) comme SASS (amélioration de syntaxe de feuille de style permettant de gérer du CSS complexe).

Il est également possible d'utiliser d'autres langages ou "Language Spec" (comme TypeScript) transcompilés en JavaScript pour améliorer la sécurité et la production de code pour celui-ci.

Finalement, il est fréquent de se reposer sur des frameworks JavaScript, qui sont des bibliothèques pré-écrites en JavaScript, pour accélérer et faciliter le développement d'applications web. Il existe différents frameworks frontend que nous allons décrire dans le chapitre "**Comparaison des frameworks JavaScript**" de ce rapport. Essentiellement, c'est un ensemble de classes, de fonctions et utilitaires pour faciliter la vie du développeur et nous éviter de réinventer la roue. Ils comprennent une gestion de l'interface utilisateur, des événements, du DOM (interface de programmation d'applications), des formulaires, entre autres fonctionnalités. Schéma ci-dessous<sup>8</sup>.

8. [https://lh3.googleusercontent.com/0jP6hKKh1Z0IBuxEbCWNJeu0BX1qFww9eUiGynrwTepqKhjxyu5bwbcMySaftEEu\\_OmuDInSZQvna6SBMmZMjUz9huJOY39Z7Bj5eHtRr1xLOIL20K2xGUvCcKgeJs1](https://lh3.googleusercontent.com/0jP6hKKh1Z0IBuxEbCWNJeu0BX1qFww9eUiGynrwTepqKhjxyu5bwbcMySaftEEu_OmuDInSZQvna6SBMmZMjUz9huJOY39Z7Bj5eHtRr1xLOIL20K2xGUvCcKgeJs1)

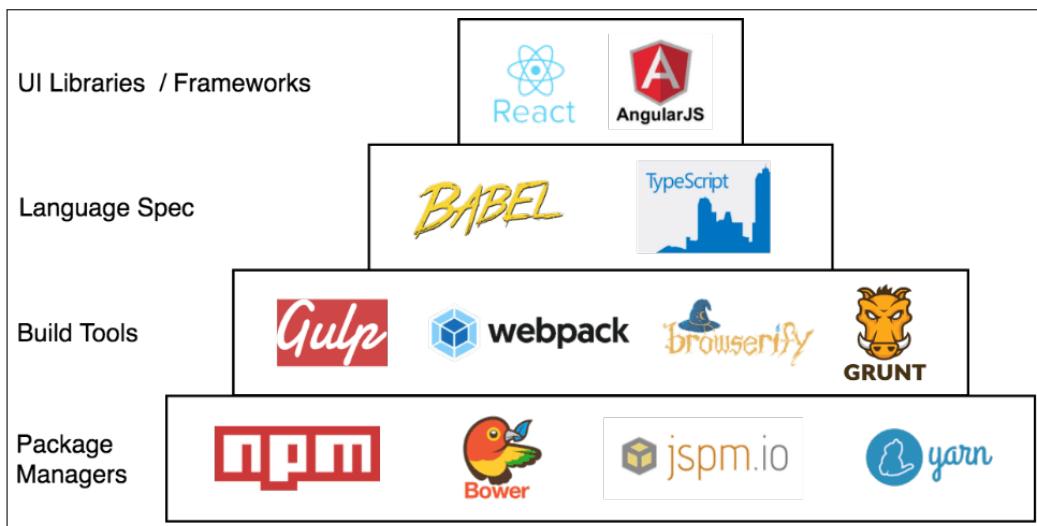


FIGURE 5.8 – État de l'écosystème JavaScript



## 6. Comparaison des technologies

Que sont réellement les Progressive Web App (PWA) ? Avant tout, il convient de définir les différences entre une application mobile native et un site responsive(ou responsive web design).

### 6.1 Site web responsive

Approche de conception web qui a pour but d'offrir une expérience optimale sur tout support en modifiant la présentation des éléments en fonction de l'appareil utilisé.

- Accessible facilement depuis un navigateur via une Uniform Resource Locator (URL)
- Développement web (HTML, CSS, JS, PHP, etc)
- Capacités multiplateformes
- Faible coût de développement, car on développe qu'une fois l'application
- Pas de système de notifications
- Difficulté d'accéder aux capteurs de l'appareil
- Mises à jour transparentes
- Connexion internet obligatoire

### 6.2 Application mobile native

Qualifie une application développée spécifiquement pour un système d'exploitation qui se télécharge au sein d'un "store" qui sert de bibliothèque d'applications.

- Doit être installé depuis un "store"
- Accessible grâce à une icône dans un lanceur d'application
- Développement spécifique à un OS (iOS, Android...)
- Langage natif à une plateforme (C#, Java, Objective C...)
- Deux ou plus de développements distincts pour chaque OS
- Accès facilité aux fonctionnalités et matériel de l'appareil (capteurs, notifications)
- Plus de performances et de fluidité
- Interaction avec d'autres applications
- Mises à jour requises
- Accessible hors-ligne

### 6.3 Progressive Web App (PWA)

Les PWA sont des pages web proposant les fonctionnalités, les interactions et le design d'une application native. C'est basiquement une application accessible depuis le web. Elles ont pour objectif d'améliorer l'expérience utilisateur en proposant des applications facilement partageables (accès via URL plutôt que store), en réduisant les temps de chargement et en les rendant facilement accessibles grâce à l'ajout d'une icône dans le lanceur d'application du smartphone.

Elles rassemblent, dans un sens, les avantages des deux types précédents. Une PWA est un site web ultra optimisé pour les plateformes mobiles tout en fournissant des fonctionnalités jusqu'alors réservées aux applications natives (fonctionnement hors ligne, icône dans le lanceur d'applications, accessible via le web, pas dépendant d'un store, rapide grâce à des données stockées localement).

Côté utilisateur, elles offrent un affichage en plein écran exactement comme une application native ainsi que des animations totalement fluides et des interactions poussées. Il est également possible d'avoir un système de notification push et une consultation hors ligne et des mises à jour transparentes grâce à des "service worker" (nous expliquerons cette notion plus tard).

Côté développeur, les PWA représentent un gain de temps en nous permettant de faire le développement qu'une seule fois pour toutes les plateformes grâce à ses propriétés héritées des sites web (accessible depuis un navigateur web et utilise les technologies de développement web comme Hypertext Markup Language (HTML)5, CSS3 et JS) au contraire des applications mobiles natives. Schéma ci-dessous<sup>1</sup>.



FIGURE 6.1 – Les 4 piliers d'une PWA

Le choix d'une PWA peut également se faire sur la base de l'engagement de l'utilisateur, en effet, l'utilisateur passe beaucoup plus de temps sur les applications mobiles natives dédiées, mais il y a beaucoup plus de personnes qui accèdent aux sites web qu'aux applications grâce à leur facilité d'accès. Les PWA permettent d'offrir une expérience digne d'une application native dans un navigateur web. L'objectif est de combiner le temps d'engagement des applications natives aux nombres d'utilisateurs des sites web dans le PWA. Schéma ci-dessous<sup>2</sup>.

1. <https://blueninja.io/wp-content/uploads/2018/04/Four-key-technology-of-a-PWA.jpg>

2. <https://igniteonline.com.au/app/uploads/2017/02/Social-ImageA.png>

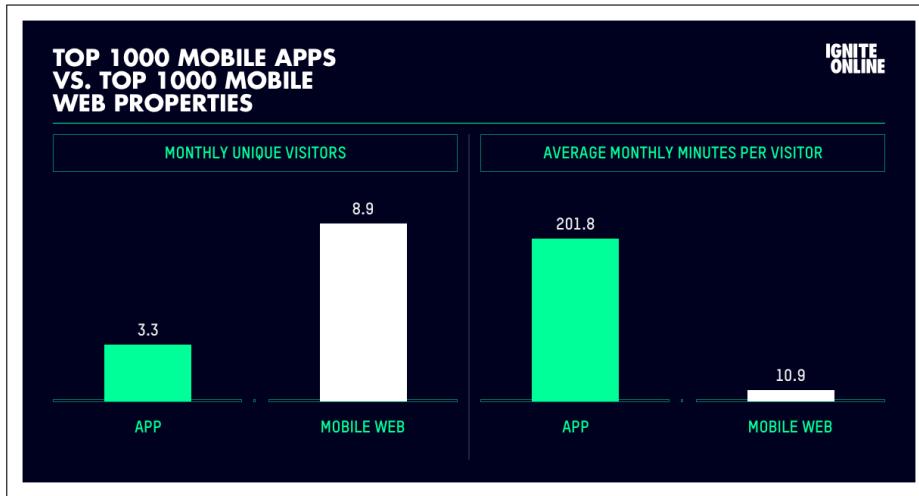


FIGURE 6.2 – Comparaison de l’engagement de l’utilisateur entre le web et les applications

## 6.4 Fonctionnement des PWA

### Fichier manifest.json

Permet au développeur de contrôler comment l’application va apparaître à l’utilisateur. Donne des META informations sur la page web et permet de l’afficher en plein écran sans la barre de navigation du browser. Il possède des informations comme l’icône de l’application (que l’utilisateur verra dans son lanceur d’application), la couleur d’arrière-plan de l’application, le nom de l’application, etc.

Pour linker un fichier manifest.json à notre application PWA :

```
1 <link rel="manifest" href="/manifest.json">
```

```
{
  "short_name": "PWA",
  "name": "My PWA",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "192x192",
      "type": "image.png",
    }
  ],
  "start_url" : "./index.html",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff",
}
```

FIGURE 6.3 – Exemple de fichier manifest.json

### Service Worker

Les Service Worker sont des scripts basés sur des évènements qui permettent d'accéder à l'application en mode hors-ligne, d'envoyer des notifications ou de mettre à jour le contenu en arrière-plan. Ils fonctionnent en arrière-plan et en parallèle de l'application web. Concrètement, ce sont des scripts JavaScript qui écoutent pour des évènements comme "fetch" ou "install" et exécutent des tâches. Schéma ci-dessous<sup>3</sup>.

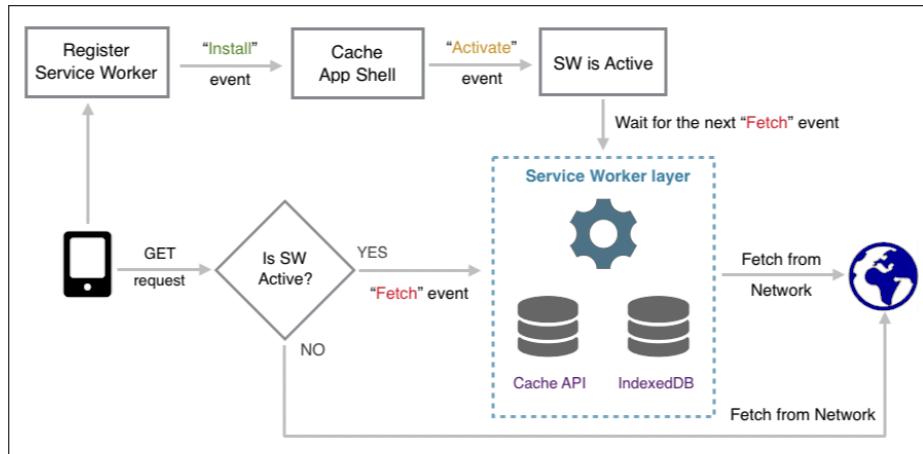


FIGURE 6.4 – Schéma de fonctionnement d'un service worker

### Notifications Push

Une notification push est un message qui apparaît sur le périphérique de l'utilisateur. Elles peuvent être envoyées localement depuis une application ouverte ou "poussées" depuis un serveur vers l'utilisateur. Elles permettent de tenir au courant l'utilisateur de nouvelles mises à jour du contenu ou de le réengager. Pour les PWAs, elles sont mises en œuvre grâce à deux Application programming interface (API) : Notification API (afficher des notifications système à l'utilisateur) et Push API (permet à un service worker de gérer un message push d'un serveur même si l'application n'est pas active). Ces deux API sont construites au-dessus de Service Worker API (répond aux évènements "push message" en arrière-plan et les relaie à l'application).

## 6.5 Analyse multicritère

Nous utilisons cette méthode de décision pour mettre en relations des critères que nous pensons indispensables à la réussite de notre projet avec les différents types d'applications que nous avons à disposition. Un poids est ajouté à chaque critère en fonction de son importance pour le projet et les solutions sont notées de 1 à 3 en fonction de leur remplissage de ceux-ci (1 ne remplissant pas du tout le critère, 2 le remplissant plus ou moins et 3 le remplissant complètement). À sa suite, une solution optimale va ressortir et sera par extension le type d'application le plus adapté à notre projet d'application de covoiturage.

3. [http://webagility.com/posts/how-progressive-web-apps-make-the-web-great-again/\\_image/f2bdbfdcc-e271-4977-94ba-e9320f21eef3:a5510468926c3cb9c06e5f07a329bd577344f5cb/full/pwa-7.png](http://webagility.com/posts/how-progressive-web-apps-make-the-web-great-again/_image/f2bdbfdcc-e271-4977-94ba-e9320f21eef3:a5510468926c3cb9c06e5f07a329bd577344f5cb/full/pwa-7.png)

Les critères que nous avons choisis sont les suivants (ils ont été mis en place et analysés grâce au site <https://whatwebcando.today>) :

- **Multiplateforme** : L’application doit être accessible autant bien sur un ordinateur que sur un mobile ou une tablette et a un caractère responsive. Si possible en ne requérant pas de développement supplémentaire et en pouvant utiliser le même code sur toutes les plateformes (code isomorphiques).
- **Notifications push** : l’application doit être capable d’envoyer des notifications facilement à l’utilisateur afin de le prévenir d’évènements ou d’informations diverses.
- **Facilement accessible** : L’application doit pouvoir être facilement partageable et accessible (sur le web mieux que sur un store par exemple)
- **Utilisation offline** : On doit pouvoir consulter un certain nombre d’informations même si l’absence d’un réseau.
- **Mises à jour** : Le système de mises à jour est transparent pour l’utilisateur et n’encombre pas l’emploi de l’application.
- **UX / UI** : il est possible de créer une interface graphique et des interactions poussées, fluides et intuitives en accord avec les principes du design d’application moderne.
- **Géolocalisation** : L’accès aux données de géolocalisation en temps réel est primordial pour notre application de covoiturage.
- **Accès au matériel** : Facilité d’utiliser les fonctionnalités et le matériel du périphérique (capteurs...)
- **Vitesse/chargement** : L’application est rapide à démarrer, à charger, elle n’est légère et économique en ressource.

CRITERES	POIDS	WEB RESPONSIVE	APP NATIVE	PWA
<b>Multiplateforme</b>	3	3	2	3
<b>Notifications push</b>	2	2	3	2
<b>Facilement accessible</b>	1	3	2	3
<b>Utilisation offline</b>	2	1	3	3
<b>Mises à jour</b>	1	3	2	3
<b>UX / UI</b>	2	2	3	3
<b>Géolocalisation</b>	2	3	3	3
<b>Accès au matériel</b>	1	2	3	2
<b>Vitesse / chargement</b>	2	2	3	3
<b>SCORE TOTAL</b>		<b>37</b>	<b>43</b>	<b>45</b>

Ce n’est pas une surprise les PWA arrivent en premier ! Notre analyse préalable du sujet nous avait déjà montré qu’elles rassemblent beaucoup d’avantages et sont un nouvel outil puissant et nos critères ont été confirmés par celle-ci. Le second choix se serait porté sur une application native qui permet une plus grande facilité d’accès aux capteurs de l’appareil et une meilleure gestion des notifications, mais requérant malheureusement trop d’investissement en termes de développement et leur dépendance d’un store les rend moins accessibles.

La solution la plus adaptée est donc les PWA. En rassemblant les avantages des applications natives et des sites web responsives, elles se hissent à la tête de notre analyse multicritère et nous allons donc utiliser cette approche pour réaliser notre projet de semestre.



## 7. Comparaison des Frameworks JavaScript

Nous allons comparer différents frameworks JavaScript dans le but de choisir le plus approprié et celui que nous allons utiliser tout au long du notre projet. Pour cela nous avons décidé d'en choisir le 3 le plus utilisés parmi tous les frameworks existants.

Tout d'abord nous avons besoin de répondre aux questions suivantes avant de passer à la partie analyse et comparaison. Qu'est-ce que c'est un framework JavaScript ? Quelles sont les différences entre un framework JavaScript et une librairie JavaScript ?

### 7.1 Qu'est-ce que c'est un framework ?

“ Un framework est un ensemble de fonctions et conventions que l'on peut réutiliser et standardiser pour faciliter la création de tout ou une partie d'un système logiciel. ”

*Nestor Peña López*

Cette définition est un résumé des au moins une dizaine des autres définitions que nous trouvons sur internet, mais si nous voulons rendre plus simple et concret ce terme nous vous proposons de voir les frameworks de la manière décrite à continuation.

Un framework, qui est formé par les mots anglais " frame " qui veut dire " cadre " et " work " qui veut dire " travail ", n'est pas autre chose que ce que son nom indique : du travail que nous allons réaliser dans et avec l'aide d'un cadre prédéfini par quelqu'un d'autre, dans le but d'économiser du temps et de nous simplifier notre travail.

### 7.2 Différences entre un framework et une librairie ?

Tout d'abord nous commencerons par définir une librairie JavaScript comme une collection des fonctions JavaScript que nous allons pouvoir réutiliser et intégrer dans notre application.

Cette différence entre librairie et framework en particulier, divise la communauté des développeurs en deux avis principaux : ceux qui assurent qu'un framework est un ensemble des librairies, donc un outil plus complexe et massif, et ceux qui assurent que la principale différence s'avère être la façon dont nous appelons ses fonctions. Avec une librairie on appelle les fonctions dans notre code tant dit qu'avec un framework nous appelons notre code depuis cette structure prédéfinie.

Un framework est beaucoup plus restrictif qu'une librairie, mais les deux ont des règles que nous devons respecter pour leur bon fonctionnement.

Comme conclusion après notre travail de recherche sur cette différence nous considérons important retenir une chose : framework ou librairie, peu importe, les deux sont des outils indispensables pour le développement web actuel et nous allons les utiliser main à main l'un et l'autre.

### 7.3 Comparaison des deux frameworks et une librairie

De nos jours ils existent une centaine de frameworks et librairies, la FIGURE 7.1<sup>1</sup> Nous montre une synthèse des frameworks et librairies le plus relevant et populaires dans la communauté.

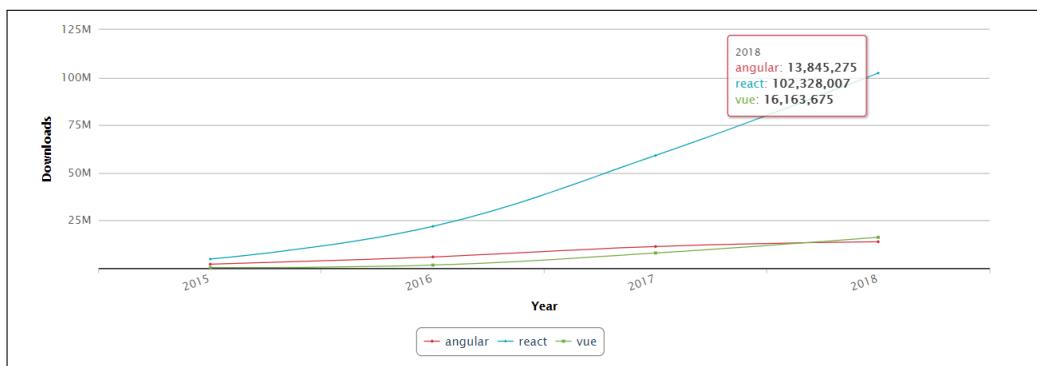


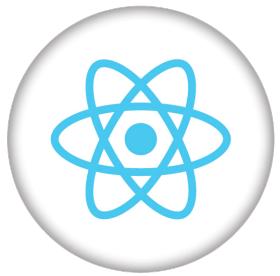
FIGURE 7.1 – Nombre de téléchargements effectués entre 2015 et 2018 dans NPM

#### 7.3.1 Angular.js



- Framework développé par Google, année de sortie en 2009.
- Très populaire parmi les développeurs.
- Très bon pour des solutions rapides et pas très travaillées.
- Beaucoup d'outils pour le travail en équipe.
- Meilleur que React pour le développement des applications niveau entreprise.
- Packages maintenus par Google.
- **Utilisé par** : Google, Ryanair, Bose ou Rockstar Games

#### 7.3.2 React.js



- Librairie développée par Facebook, année de sortie en 2013.
- Plus facile à mettre à l'échelle.
- Bon pour les gros projets front-end.
- API relativement petite.
- Le rendu constant des composants permet une organisation efficace en cas de complexité croissante.
- **Utilisé par** : Facebook, AirBnB, Netflix, Paypal, Twitter

<sup>1</sup>. <https://npm-stat.com/charts.html?package=react&package=vue&package=angular&from=2015-01-01&to=2018-10-29>

### 7.3.3 Vue.js



- Framework développé par Evan You, année de sortie 2014.
- API très simple.
- Permet de poursuivre des modules sélectifs.
- Toujours adoptable.
- Facile à intégrer avec d'autres bibliothèques ou projets existants.
- Bon pour les applications à grande échelle.
- **Utilisé par :** Alibaba, Nintendo, Gitlab, Expedia

## 7.4 Analyse multicritère

Maintenant que nous avons une vue globale de ces 3 frameworks, nous allons définir les aspects importants et déterminants (les critères) pour réaliser notre choix. Tout cela en tenant bien en compte la technologie choisie avec notre analyse antérieure : les PWA. Nous avons choisi pour cette analyse d'utiliser un système de "classement" à 3 places (la position 1 étant au plus bas et la position 3 la meilleure). Il est possible d'avoir des égalités dans le classement. À sa suite, une solution optimale va ressortir et sera par extension le framework à privilégier pour notre application de covoiturage.

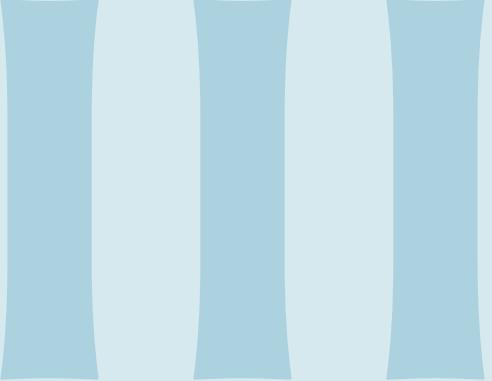
Les critères que nous avons choisis sont les suivants :

- **Courbe d'apprentissage :** La courbe d'apprentissage de Vue.js est clairement plus facile à suivre que celle des autres frameworks proposés ; Vue a tendance à utiliser simplement JavaScript en ajoutant quelques idées, tandis que React ou Angular imposent d'adapter des pratiques éculées de programmation JavaScript à une nouvelle façon de faire particulière au framework : JSX pour React et TypeScript pour Angular.
- **Richesse de son écosystème :** Les librairies techniques sont pléthoriques pour React comme pour Vue, ce qui peut sembler rebutant au premier abord ; il faut cependant relativiser et bien comprendre qu'une fois ajoutées les deux ou trois librairies incontournables (bien documentées, bien maintenues et facilement intégrables), vous couvrirez généralement 90% de vos besoins applicatifs. Les 10% restants étant couverts par des librairies additionnelles que pour l'essentiel vous ajouterez également à vos frameworks "tout en un". Pour la création des applications mobiles, React Native donne la possibilité de créer des composants pour mobile en se basant sur la même syntaxe que React, un composant utilisable dans le navigateur pouvant également être utilisé dans une application mobile.
- **Intégration aux autres technologies :** Angular est clairement réservé à la construction de (grosses) applications "simple-page" et même s'il est théoriquement possible de l'intégrer à des applications web utilisant un autre framework, ça peut très vite devenir compliqué. À l'inverse Vue.js et React.js offrent cette possibilité de très facilement s'intégrer à n'importe quelle application web puisqu'il suffit d'importer la librairie (très légère) correspondante.
- **Popularité et réactivité communautaire :** Comme nous avons vu précédemment React est en tête du classement, suivi par Vue et Angular. Cette popularité s'avère un facteur important vu que la communauté joue un rôle fondamental dans le développement moderne.(FIGURE 7.1 )

- **Qualité de sa documentation** : La qualité de la documentation est indéniable pour l'ensemble des solutions considérées ; on peut toutefois donner un léger avantage aux solutions orientées “framework tout-en-un” comme Angular, puisqu'elles garantissent de fait une documentation centralisée et de qualité homogène.
- **Performance** : Il y a un revers à toutes les fonctionnalités : le framework Angular est assez bouffi. La taille du fichier compressé est de 143kb, contre 23kb pour Vue et 43kb pour React. Tous ces frameworks sont vraiment très proches les uns des autres par rapport aux frameworks particulièrement lents ou rapides. Encore une fois : les critères de performance ne doivent être considérés que comme une note d'accompagnement, pas comme un verdict.

CRITERES	POIDS	ANGULAR	REACT	VUE
Courbe d'apprentissage	3	1	2	3
Richesse d'ecosystème	3	1	3	2
Integration aux autres technos	2	2	3	1
Popularité et réactivité	2	1	3	2
Qualité documentation	1	3	2	2
Performance	2	1	2	3
<b>SCORE TOTAL</b>		<b>17</b>	<b>33</b>	<b>29</b>

Comme nous pouvons remarquer les résultats globaux favorisent React suivi de près par Vue. Ce résultat vient confirmer la suggestion de notre responsable Madame Ingram. Nous voulons de toute façon mettre en évidence que cette décision reste plutôt subjective et que l'évolution constante du monde des frameworks peut sans doute faire ressortir des résultats très différents de ceux qu'on constate aujourd'hui.



# Analyse besoins UX et marketing

<b>8</b>	<b>Besoins des utilisateurs .....</b>	<b>32</b>
8.1	Analyse des besoins	
8.2	Problèmes potentiels qui peuvent être rencontrés	
<b>9</b>	<b>Analyse de la concurrence .....</b>	<b>35</b>
<b>10</b>	<b>Identité visuelle et promotion .....</b>	<b>40</b>
10.1	Material design vs Flat design	
10.2	Principes du material design	
10.3	Framework CSS	
10.4	Identité visuelle	
10.5	Motivation à l'utilisation	
10.6	Communiqué de presse	
10.7	Idées de promotion locale	



## 8. Besoins des utilisateurs

“ Le covoiturage est l'utilisation conjointe et organisée (à la différence de l'auto-stop) d'une voiture automobile, par un conducteur non professionnel et un ou plusieurs tiers passagers, dans le but d'effectuer un trajet commun.<sup>1</sup> ”

*Wikipedia*

### 8.1 Analyse des besoins

L'analyse des besoins a été réalisée par la classe I-3 dans le cadre du cours IHM-2 donné par Madame Ingram (respectivement : Ayrton Dumas, Yann Doudin, Vallat Matthieu, David Alvarez, Gilles Waeber et Alain Schaller). Nous nous reposons sur leurs travaux pour définir les besoins utilisateurs.

#### 8.1.1 WAAD

Le WAAD regroupe toutes les différentes activités dans lesquels notre application aura un impact. Plusieurs rôles ont été définis et différents souhaits ont été formulés autour de l'idée de l'application de covoiturage local.

- **Externe** : personne souhaitant découvrir la plateforme ou devenir utilisatrice
- **Admin** : personne administrant la plateforme
- **Passagers** : personne utilisant la plateforme en tant que passagère
- **Conducteur** : personne conduisant des passagers

1. <https://fr.wikipedia.org/wiki/Covoiturage>

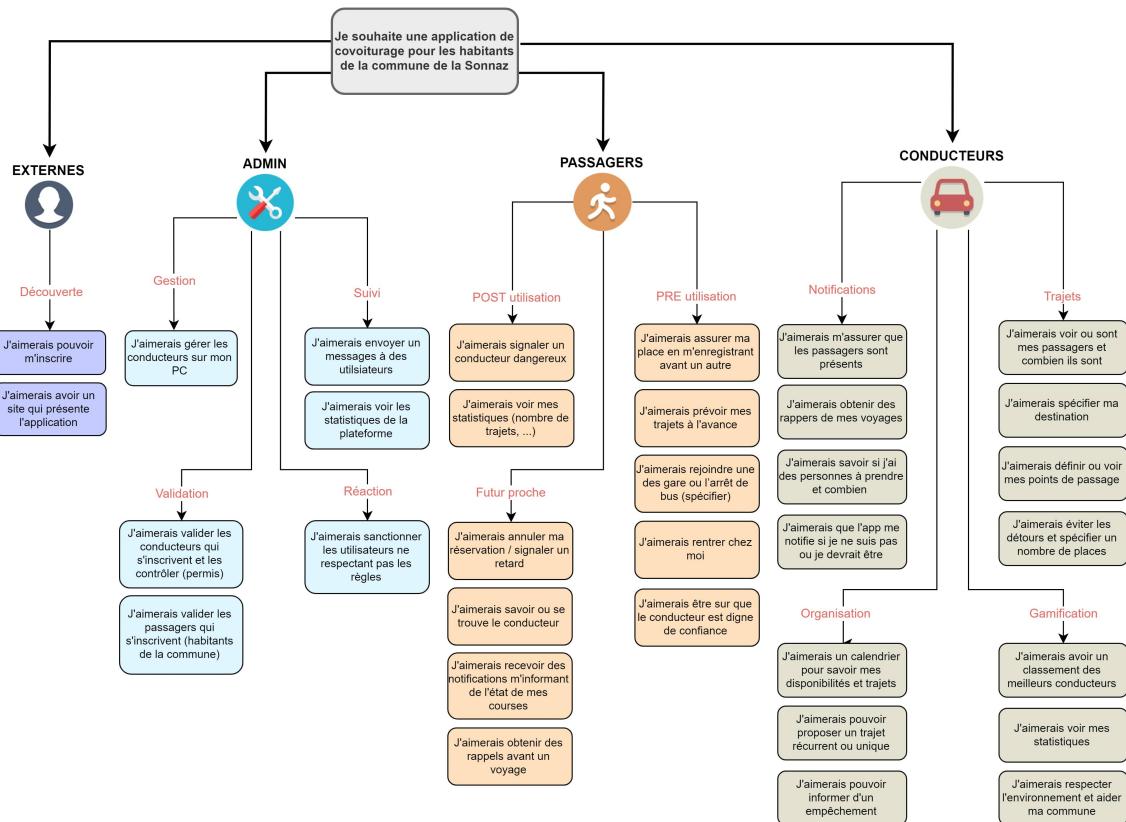


FIGURE 8.1 – WAAD dans sa globalité

### 8.1.2 Rôles

Les rôles définissent les différents types d'utilisateurs qui interagissent avec l'application. Chaque rôle a des objectifs et des besoins spécifiques et doit effectuer différentes tâches.

- **Rôle externe** : ce ne sont pas eux que l'on va interviewer pour l'utilisation de l'application
- **Rôle direct** : agit directement avec l'application. L'application doit lui être adaptée.

#### Externe

Objectifs :

- S'inscrire en tant que conducteur ou passager
- Visiter le site pour avoir un aperçu de l'app

Rôle : Externe

#### Admin

Objectifs :

- Authentifier les utilisateurs

Tâches :

- S'assurer que les utilisateurs inscrits proviennent de la commune
- Vérifier la validité des permis de conduire des conducteurs
- Gérer les plaintes des utilisateurs

Rôle : Interne

**Passager**

Objectifs :

- Être amené par un conducteur à une des gares de Belfaux

Tâches :

- Sélectionner un trajet proposé par un conducteur
- Demander à ce qu'on le prenne à un certain endroit

Rôle : Interne

**Conducteur**

Objectifs :

- Prendre des passagers dans son véhicule afin de les amener à une des gares de Belfaux

Tâches :

- Définir par quels endroits il passe dans le village (définir un trajet)
- Proposer de prendre un passager ayant fait une demande pour qu'on le prenne à un certain endroit

Rôle : Interne

## 8.2 Problèmes potentiels qui peuvent être rencontrés

Un projet réalisé en 2004 par la HEIA-FR et par Monsieur Schroeter<sup>2</sup> a eu une approche assez similaire à la notre en souhaitant offrir un système de covoiturage avec des arrêts définis. Le prototype à relever le potentiel problème suivant lié à la mise en œuvre de notre application de voiturage :

**“ Pour une réussite du projet de covoiturage, des incitations financières et morales importantes sont indispensables. ”**

*Nicolas Schroeter*

**Nécessité de mesures incitatives :**

- Si on se base sur la seule participation volontaire, un échec du système de covoiturage est inévitable.

Nous sommes confiants que ce problème ne nous concerne pas, car nous réalisons un covoiturage dans une zone géographique très limitée. La plateforme étant restreinte qu'aux habitants de la commune, un sens de la communauté est plus présent et les personnes sont beaucoup plus enclin à faire confiance et à l'effort purement altruiste.

Nous étudierons des moyens incitatifs plus tard dans cette analyse dans le **Chapitre 10.5**.

---

2. SwissMobil : Service offert à l'usager



## 9. Analyse de la concurrence

Selon un article publié le 03/10/2018 sur le **Site Officiel de l'Etat de Fribourg**<sup>1</sup>, le canton de Fribourg dispose de deux plateformes de covoiturage distinctes :

- **Frimobility** : site de covoiturage développé par l'Association des communes fribourgeoises (ACF) en partenariat avec le Groupe E et les Transports publics fribourgeois (TPF).
- **Fribourg-covoiturage** : plateforme cantonale de covoiturage gérée par l'association suisse **e-covoiturage.ch** qui bénéficie du soutien de la Confédération.

Ainsi qu'une place de covoiturage créé en 2012 par l'état de Fribourg à proximité de la jonction autoroutière N12 de Vaulruz. Elle comporte 20 places de parc.

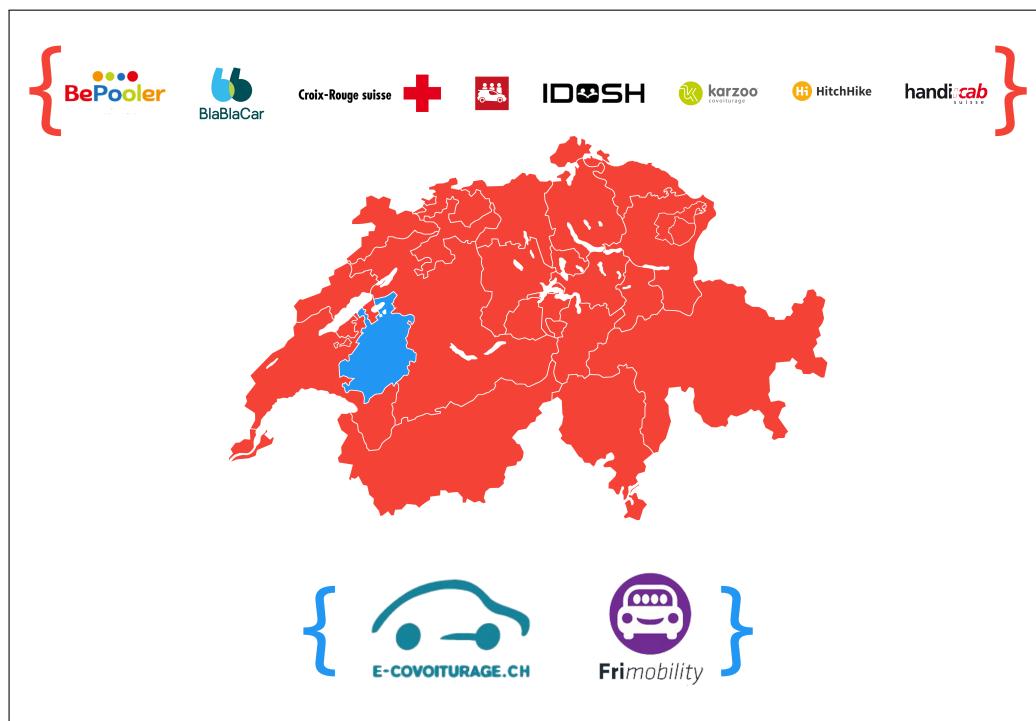


FIGURE 9.1 – Applications de covoiturage en Suisse

Comme nous pouvons observer dans la FIGURE 9.1 ils existent des autres applications de covoiturage que peuvent être utilisés dans le canton de Fribourg, mais son utilisation réelle actuelle est presque nulle, donc nous allons nous contenter de les citer :

1. <https://www.fr.ch/smo/mobilite-et-transport/en-voiture/covoiturage>

- **Mobility Carpool** - <https://www.mobility.ch/fr/mobility-carpool> : plateforme permettant d'effectuer du covoiturage en voiture Mobility ou privé. L'inscription et l'application sont gratuites, mais une taxe de 20% du prix de la course est perçue par Mobility. Le prix de la course variant entre 3 et 18 francs, selon la distance. Le paiement est effectué par carte de crédit.
- **BlaBlaCar** - [www.blablacar.fr](http://www.blablacar.fr) : plateforme permettant de faire du covoiturage dans une vingtaine de pays. L'inscription et l'application sont gratuites, mais les frais de réservation seront proportionnels au prix du trajet. Le prix de la course est fixé par le conducteur (un prix par kilomètre et par passager est recommandé par BlaBlaCar). Le paiement s'effectue par PayPal, carte de crédit ou de débit Visa et Mastercard.
- **BePooler** - <https://www.bepooler.com/ch/?lang=fr> : une application toute récente née en 2015. Une plateforme simple et une expérience utilisateur spécialement conçue pour les besoins des navetteurs. L'application vous permet de gérer le trajet partagé entre collègues de travail ou de district, ainsi que de réserver des places de stationnement réservées aux voitures partagées. Actuellement, présente seulement en Suisse italienne et à Milano.
- **Karzoo** - [www.karzoo.ch](http://www.karzoo.ch) : portail de covoiturage entre privés et au sein d'entreprise en Suisse et en Europe. L'inscription est gratuite. Le prix de la course négocié entre les deux parties (Karzoo propose des recommandations tarifaires), mais aucun paiement effectué via [www.karzoo.ch](http://www.karzoo.ch).
- **Idosh** - <https://idosh.me/> : application permettant de proposer ou rechercher un covoiturage en Suisse. L'application est gratuite. Les courses étaient gratuites durant la phase pilote puis à terme, un prix de 35 centimes a été imposé par kilomètre et par passager. Le paiement par carte de crédit via l'application. L'application est seulement disponible en allemand et anglais.
- **HitchHike** - <https://www.hitchhike.ch/> : plateforme de covoiturage personnalisée à l'intention d'entreprises, de régions, de hautes écoles ou de grandes manifestations. L'accès est gratuit pour les membres des communautés et le prix convenu entre les parties (HitchHike recommande une indemnisation financière ou en nature équitable). Le paiement est effectué directement entre les "HitchHikers".
- **Services de transports** - Croix Rouge Suisse - <https://www.redcross.ch> : services de transports régionaux prenant en charge le déplacement de personnes à mobilité réduite. L'inscription est faite par téléphone auprès des associations cantonales respectives. La participation est aux frais de la course. Il existe une diversité par rapport aux modes de paiement selon les cantons.
- **Handi-cab Suisse** - <http://www.handi-cab.ch/> : services de transport régionaux de personnes à mobilité réduite. Les informations sur l'inscription, le prix des courses et les modes de paiement acceptés sont sur les sites internet des différentes associations régionales.

Nous allons maintenant décrire plus en détail les fonctionnalités des applications principales et les comparer avec les fonctionnalités de notre future application, pour cela nous diviser notre analyse en trois parties principales :

1. Utilité : servir à la réalisation d'une activité humaine. Le système doit servir à quelque chose d'utile et faciliter la tâche de son utilisateur.<sup>2</sup>
2. Utilisabilité : caractérise la capacité d'un objet à être facilement utilisé par une personne donnée pour réaliser la tâche pour laquelle il a été conçu. C'est une notion fortement liée à celle d'ergonomie qui caractérise l'adaptation d'un système au travail et au bien-être des êtres humains (du grec ergon : travail et nomos : règle, loi naturelle).<sup>3</sup>

2. Concept tiré du cours d'IHM-1

3. Concept tiré du cours d'IHM-1

3. Impact émotionnel : Dans un contexte marketing, l'impact émotionnel désigne les conséquences à l'égard d'une marque ou entreprise des émotions ressenties par un ou plusieurs consommateurs.<sup>4</sup>

#### 9.0.1 Utilité

Pour cette partie nous tenons à préciser que le but global des trois applications est pratiquement le même, les trois vont servir à faire du covoiturage. Par contre si nous comparons plus en détail, nous pouvons remarquer que notre application est plus axée sur l'aspect communautaire. Son utilité ultime est bien celle de rendre la vie des habitants de la commune de la Sonnaz plus agréable et facile, en assurant un moyen de transport indispensable en ce moment pour tous ces habitants tout en la gardant son utilisation gratuite.

Nous pouvons observer un résumé des aspects que nous avons définis comme essentiels pour que notre application soit en concordance avec les besoins des mandats.

	Frimobility	Fribourg-covoiturage
<b>Disponible sur Mobile et Web</b>	non	non
<b>Trajets au sein de la commune</b>	non	non
<b>Gratuit</b>	non	non
<b>Confiance/sérénité/contrôle</b>	non	non

Nous constatons assez aisément qu'aucune de ces plateformes ne remplit le cahier des charges de notre application de covoiturage local. Les attentes sont très précises et particulières, il est donc nécessaire de pousser en avant le projet afin de fournir à la commune de la Sonnaz un service qui leur convient et de ne pas à avoir à détourner un système existant.

#### 9.0.2 Utilisabilité

	Frimobility	Fribourg-covoiturage
<b>Année de création</b>	2005	2014

Comment nous pouvons observer sur ce petit tableau comparatif ces applications ne sont pas très récentes, en spécial le cas de fribourg-covoiturage qui a déjà plus de 13 ans sur le marché. Cet aspect est relevant vu l'avance foudroyante de la technologie qui peut faire qu'une application sans une maintenance régulière devienne très vite incompatible avec le "modus operandus" moderne, ce qui va rendre son utilisabilité de plus en plus difficile.

Nous voulons aussi vous démontrer qu'actuellement est impossible de réserver un trajet depuis la Sonnaz avec ces deux applications. Les trajets qui nous sont proposés par les applications ne sont pas du tout en correspondance à nos critères de recherche.

4. <https://www.definitions-marketing.com/definition/impact-emotionnel/>

The screenshot shows the Frimobility search interface. At the top, there are input fields for 'A' (La Sonnaz, Suisse) and 'B' (Fribourg, Suisse), a 'Trajet aller-retour?' switch (set to 'NON'), and 'Vous êtes?' switches for 'Conducteur' (unchecked), 'Passager' (unchecked), and 'Les deux' (checked). A large purple button labeled 'Chercher Covoiturage' is at the bottom right. Below the search bar, the results are displayed under the heading 'Les trajets pour La Sonnaz - Fribourg'. Three entries are shown:

- Fabienne**: Châtel-Saint-Denis → Fribourg. L M M J V S D. 1 car. **Détails/Contact**
- Nicolas**: Villars-sur-Glâne ↔ Berne. L M M J V S D. 1 car. **Détails/Contact**
- Pascal**: Farvagny ↔ Fribourg. L M M J V S D. 1 car. **Détails/Contact**

FIGURE 9.2 – Réservation d'un trajet depuis la Sonnaz avec Frimobility

The screenshot shows the Fribourg-Covoiturage search interface. On the left, there are dropdown menus for 'Départ prédefini' (set to 'Votre choix') and 'Destination prédefini' (set to 'Votre choix'), and a 'Type de Trajet' section with a 'Tous' radio button selected. Below these are 'Mon statut' options and a 'RECHERCHER' button. The main area features a map of the Fribourg region with a blue route line and orange shaded areas indicating search results. A legend on the right includes icons for a question mark, a person, a gear, a magnifying glass, and a refresh symbol. At the bottom, it says 'Liste de tous les trajets : 1 résultat(s)' and shows a table with one result: 'Départ Belfaux' and 'Arrivée Fribourg'. The table includes icons for a person, a car, km, CO<sub>2</sub>, and a battery. A note at the bottom right states 'Aucun trajet ne correspond à vos critères ?' with links to 'Proposez votre propre trajet' and 'Trouvez d'autres moyens de transport'.

FIGURE 9.3 – Réservation d'un trajet depuis la Sonnaz avec Fribourg-Covoiturage

### **9.0.3 Impact émotionnel**

L'aspect émotionnel s'avère aussi un facteur clé dans la réalisation de notre application. Ça sera évidemment une application réalisée pour les habitants de cette commune, donc nous voulons qu'ils se sentent identifiés. Ils pourront jamais avoir ce genre de compromis avec les deux applications qu'ils existent sur le marché. Cette application sera réalisée sur mesure et en respectant les souhaits de nos mandants.

# 10. Identité visuelle et promotion

## 10.1 Material design vs Flat design

### 10.1.1 Principes du flat design

“ Less is More. ”

Ludwig Mies Van Der Rohe

Style d'interface graphique caractérisé par son minimalisme. Il se base sur un emploi de formes épurées et simples et d'aplats de couleurs vives (peu de profondeur, de dégradés, de textures, de reflets et d'ombres). Il a donc des significances moins flagrantes.

Il est le total opposé du skeuomorphismes (imitation de l'apparence d'objets réels pour faire comprendre à l'utilisateur son fonctionnement).

Les détracteurs du flat design disent qu'il manquerait d'affordance<sup>1</sup>, ce qui est la capacité d'un design à évoquer sa fonction, à appeler à l'action. À cause de son côté (trop) minimalist, il peut être perçu ou interprété de plusieurs manières différentes.

Les personnes le prêchant sont d'avis que le public a désormais les clés et les codes du monde digital. Le flat design serait donc un moyen de renforcer l'utilisabilité en simplifiant et épurant les interfaces.



FIGURE 10.1 – Exemple de skeuomorphisme dans iOS6 comparé à du flat design

1. <https://www.usabilis.com/definition-affordance>

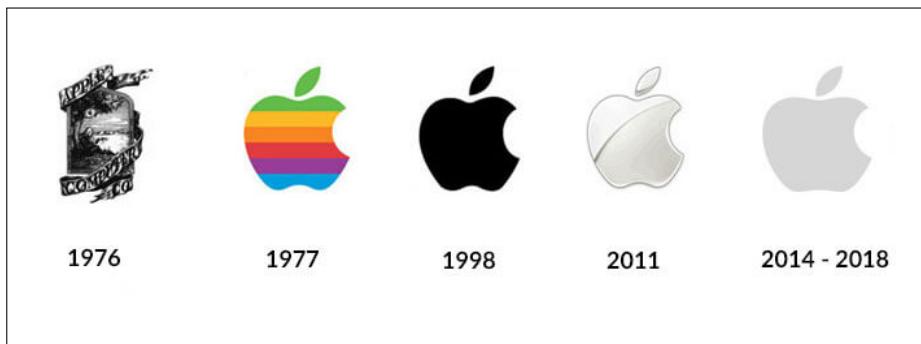


FIGURE 10.2 – Évolution du logo de Apple vers le flat design

## 10.2 Principes du material design

Le material design a été présenté par Google en 2014. Il s'inspire du flat design, mais lui donne une tout autre dimension : la profondeur. Il utilise des animations, des transitions et des effets de parallaxe pour le rendre le design plus "vivant" et met un point d'honneur à structurer les interfaces sous forme de "feuille de papier" ou de cartes empilés. Les éléments glissent les uns sous les autres au lieu d'apparaître et de disparaître. Cela donne de la profondeur à l'application, les superpositions sont mises en évidence par des jeux d'ombres et de lumières qui nous informent également des éléments avec lesquels il est permis d'interagir.

Le material design vise à éliminer les problèmes d'affordance du flat design. Ils le font grâce à l'introduction des principes cités ci-dessus (la profondeur, les ombres, le système de cartes, les animations). Il reprend, en un sens, des principes "skeuomorphe" pour les adapter au flat design. Tout en gardant une interface minimalist et épurée. Il parvient à en améliorer grandement l'utilisabilité grâce à l'ajout de petites astuces.

Google propose des directives précises sur les principes du material design sur le site suivant : <https://material.io/>

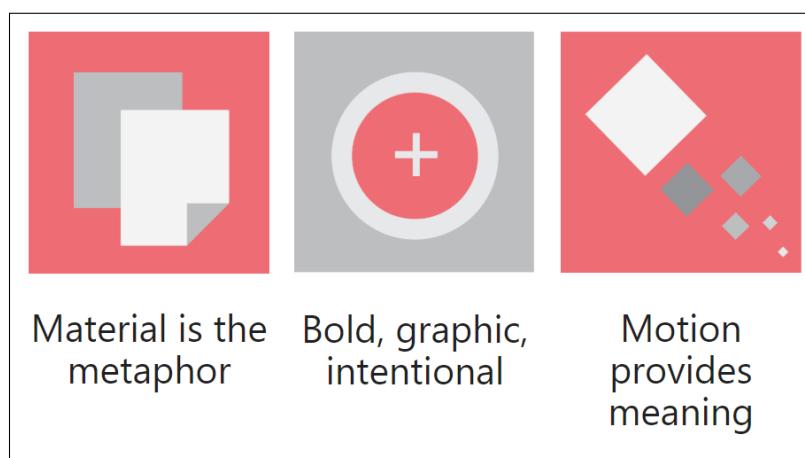


FIGURE 10.3 – Principes du material design par Google

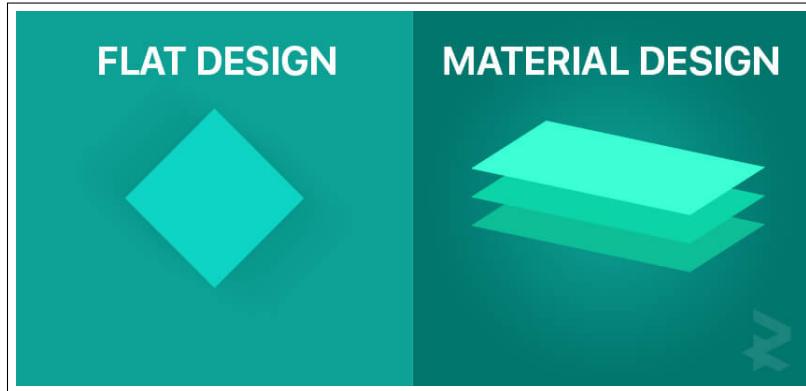


FIGURE 10.4 – Comparaison de flat design et material design

### 10.3 Framework CSS

La notion de framework a déjà été analysée plus haut. Un framework CSS est très similaire à un framework JavaScript, à la différence qu'il propose une collection d'outils utiles à la création graphique frontend sous forme d'ensemble de code CSS et HTML. Ils comprennent en général des formulaires, des boutons, des outils de navigation, des outils de mise en page (grilles), de gestion de responsive design, des éléments d'interaction, etc. Ils permettent un web design plus facile et plus standard en utilisant le langage CSS et, comme dit plus haut, il "mâche" le travail du développeur qui n'est pas forcément graphiste en lui proposant diverses fonctionnalités pour alléger son workflow.

Madame Ingram nous a dirigés pour ce projet de semestre vers le Framework : **Materialize CSS**. C'est un framework qui est disponible en version CSS ou Sass et qui est inspiré de l'univers graphique de Google : Le material design. Il offre autant de fonctionnalités que les autres frameworks CSS, mais à la différence qu'il suit justement les principes du material design. Il fournit de nombreux composants prêts à l'emploi afin de reproduire en un temps réduit des applications web respectant le style material design.



FIGURE 10.5 – Logo et résumé de Materialize

## 10.4 Identité visuelle

Avant de s'atteler à l'imagination et la création d'une identité visuelle, il convient de définir précisément de quoi il en retourne :

**“ Identité visuelle : ensemble d’éléments graphiques constituants l’identité d’une entité.”**

*Cherix Gilles*

Une identité visuelle permet de se démarquer, de se faire remarquer, connaître et reconnaître. Elle est couramment composée d'un logo, de typographies, de formes (éléments graphiques distincts, icônes) et de couleurs. Sans identité visuelle, une entité n'a pas de personnalité et sombrera dans l'inconscience collective. Le but est également d'avoir une uniformité dans les interfaces et dans les publications.

Sa création se fait avec une succession de choix reposant sur des analyses préalables :

- **Les valeurs que nous souhaitons véhiculer** : permet de nous différencier
  - Questions à se poser : quels sont nos valeurs, notre secteur, nos engagements, notre proposition ?
- **Les couleurs** : représentent nos valeurs et sont primordiales, reposent sur une lourde signification psychologique
  - Questions à se poser : sont-elles en rapport avec nos valeurs ? Quelles sont leurs significations psychologiques ? Sont-elles attrayantes dans mon secteur d'activité ?
- **La typographie** : qui peut connoter le sérieux ou non
  - Questions à se poser : est en rapport avec les valeurs ? Se marient bien avec nos choix de couleur ? Police plutôt légère ou grasse ?
- **Le logo** : contiens nos couleurs, notre typographie principale et nos valeurs. Le logo est le signe principal de reconnaissance d'une entité.
  - Questions à se poser : quelles formes souhaitons-nous utiliser (douces / dures) ? Quel est le sentiment à faire passer ? Souhaitons-nous suivre des directives de design (flat/responsive)
- **La charte graphique** : définit les règles fondamentales d'utilisation de notre identité visuelle, les "do's and don'ts" afin de garder une cohérence au sein de notre communication.

Les étapes de sa création sont :

- La recherche créative (analyse et réflexion)
- Définition des éléments graphiques
- Approbations de la charte graphique
- Diffusion de l'identité visuelle

Nous ne réaliserons pas toutes ces analyses préalables dans ce projet de semestre. Le temps est limité et il est nécessaire de fournir un prototype testable plus que tout, nous avons donc décidé de ne pas "perdre" trop de temps sur l'analyse graphique et avons donc foncés directement à l'étape de définition et réalisation des éléments graphiques en les justifiant brièvement. La recherche créative et les analyses étant mis au second plan.

#### 10.4.1 Nom et valeurs

Nous avons choisi : **Notre nom** : Sonnaz Go **Nos valeurs** : le partage, le transport et l'écologie.

Il faut bien sûr avant tout choisir un nom sur lequel toute notre identité visuelle se reposera. Il doit être court, percutant, mémorable et distinctif. Il n'est pas obligatoire qu'il reprenne nos valeurs, mais cela peut aider le consommateur final à mieux identifier le but du produit. Nous avons choisi la simplicité et la généricité. Pas de nom extravagant ou ultra original, juste un nom facile à retenir et que tout utilisateur de l'application peut aisément glisser dans une conversation et se faire comprendre. Les utilisateurs de l'application sont restreints à une même commune rurale, pas d'ambitions internationales. Quelque chose de simple, direct, compréhensible et percutant à la première lecture par tout un chacun est donc primordial dans un contexte tel que le nôtre.

"Sonnaz Go" rassemble toutes ces qualités, le "Go" exprime le mouvement, le voyage, le transport tandis que l'ajout du nom de la commune "Sonnaz" nous permet de facilement indiquer que le champ d'utilisation de l'application est restreint à cette zone seule, renforçant l'aspect communautaire.

#### 10.4.2 Logo et formes

**Élaboration et croquis initiaux**



FIGURE 10.6 – Recherche de logo en ayant initialement comme nom "Via Sonnaz" et étant ensuite passé à "Sonnaz Go", croquis et vectorisation



FIGURE 10.7 – Premier jet de logo : rejeté

Le processus de création de l'identité visuelle est passé par plusieurs étapes, la première étant l'élaboration de croquis du logo. Ces essais nous permettent de facilement faire un inventaire des possibilités, de ce qui plaît ou pas, de ce qui est cohérent ou non. Le logo doit respecter, comme tous les autres aspects de l'identité visuelle, nos valeurs.

Nous avons effectué plusieurs tests de logos que nous avons invalidés soit à cause de leur côté "hors sujet", soit à cause de formes trop simplistes et ne représentant pas assez nos valeurs et l'idée d'une application de covoiturage. Nous nous sommes finalement mis d'accord sur un des croquis et avons itéré sur celui-ci afin de créer notre logo final.

### Logo final

Le logo final a été réalisé dans le but de respecter nos valeurs ainsi que la direction material design que nous avons choisie, tant au niveau des formes que des couleurs.

Le logo est séparé en deux : un label ("Sonnaz") et un label graphique ("go") que nous pourrons utiliser indépendamment du label, mais n'est pas le cas pour l'inverse, tout ceci sera illustré dans un résumé ci-après. Nous avons repris les formes étudiées dans les esquisses afin de le mettre au propre. Nous avons fait une seconde version que nous avons soumise à nos responsables afin de pouvoir le critiquer et l'améliorer. Il en est ressorti qu'il fallait enlever la redondance du "Go" ainsi qu'ajouter un détail permettant de plus faire ressortir l'aspect "transport sur route". Pas de critiques à l'encontre des couleurs ou de la police, l'aspect material design ainsi que le concept de base ont plu et le résultat est esthétique bien que cette critique soit purement subjective.



FIGURE 10.8 – Second test de logo au propre



FIGURE 10.9 – Logo final de Sonnaz Go avec les modifications

### Couleurs

Les couleurs utilisées font partie de la palette de couleurs material design incluses dans le framework CSS Materialize.<sup>2</sup>

De plus, nous avons fait le choix d'un vert/bleu (teal ou bleu sarcelle) afin de retrouver nos valeurs écologiques dans le logo. Nous avons choisi de nous diriger vers un vert bleuté suite à des études.<sup>3</sup> Des Universités de l'Oregon et de Cincinnati sur la perception des couleurs qui ont prouvé que le bleu, plus que le vert, donne une impression de respect de l'environnement.

Le logo est représenté en deux variantes, l'une sur fond clair et l'autre sur fond foncé.

### Typographie

La police d'écriture choisie, notamment pour le "Sonnaz" du logo, est : **Brandon Grotesque (Black)**.

Elle a été désignée en 2010 par Hannes von Döhren et a largement été influencée par les styles géométriques et sans-sérifs qui étaient populaires dans les années 20 et 30. La typo est basée sur des formes géométriques qui ont été corrigées visuellement pour une meilleure lisibilité. La police a un look fonctionnel avec une touche chaleureuse et élégante de par ses courbes et ses volumes généreux.

Elle est parfaitement adaptée à notre application, parfaitement lisible, assez grasse pour être mise en évidence et un léger côté cartoon renforçant l'identité communautaire de notre plateforme. Elle a également l'avantage d'être librement utilisable sur une plateforme web<sup>4</sup>.



FIGURE 10.10 – Brandon Grotesque Font

2. <https://materializecss.com/color.html>

3. <https://around.uoregon.edu/content/new-study-suggests-color-affects-ethical-judgments-brands>

4. <https://www.myfonts.com/fonts/hvdfonts/brandongrotesque/licensing.html>

### 10.4.3 Identité visuelle finale

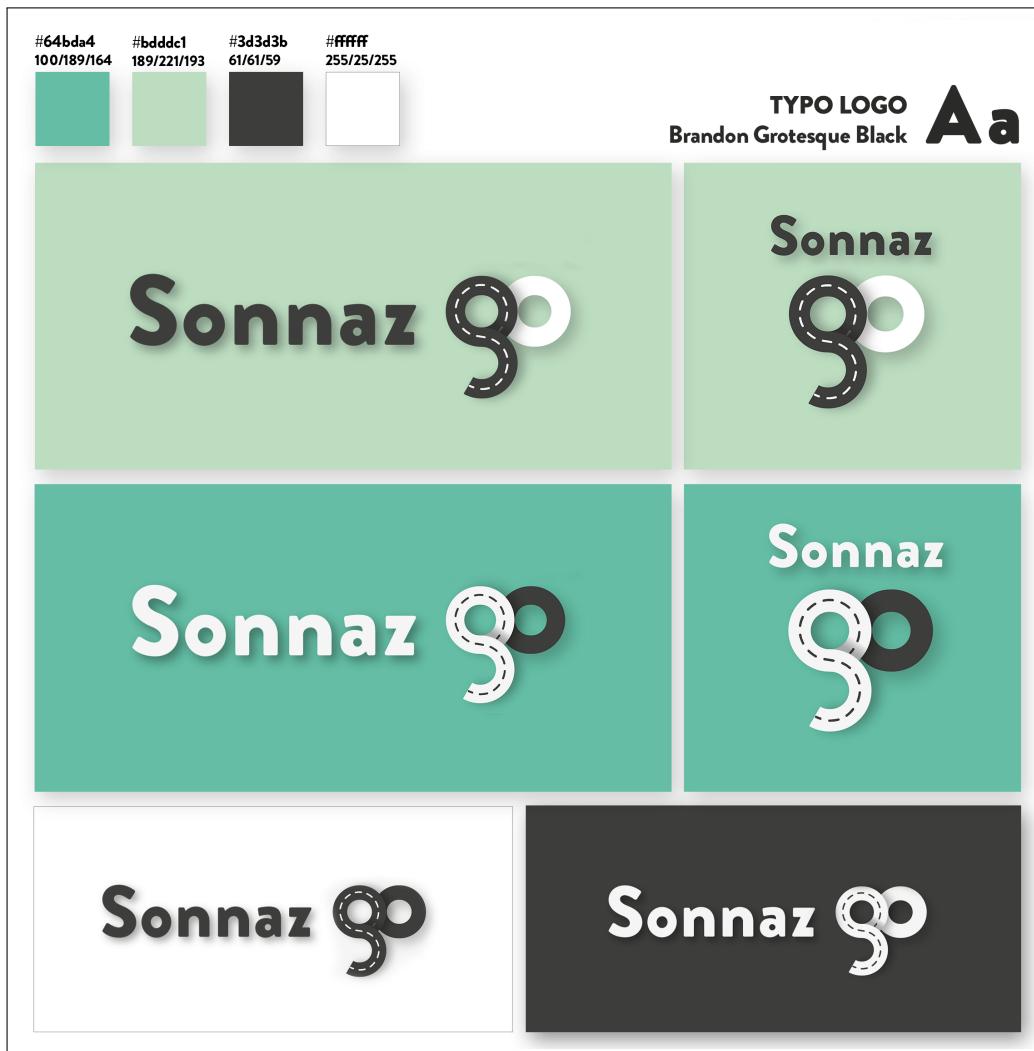


FIGURE 10.11 – Identité visuelle de Sonnaz Go

### 10.5 Motivation à l'utilisation

Nous avons réfléchi à comment stimuler l'utilisation de l'application tout en respectant le concept de gratuité souhaité par les mandants. Voici une petite liste des idées encore à développer :

- **Système de classement** : nous avons proposé cette idée depuis les débuts et nous avons imaginé un système de classement qui accentue l'aspect de la gamification de l'application en récompensant avec des badges aux utilisateurs les plus actifs et ceux qui ont réussi à relever certains "challenges".

- **Promotion de producteurs locaux** : cette idée nous est venue avec l'analyse de la concurrence, vu que les deux applications analyse en détail comptent avec l'aide des sponsors. Nous voulons surtout viser le parrainage des producteurs locaux pour accentuer l'esprit "quid pro quo" entre les habitants de la commune, ce qui est à la fin le but ultime de notre application.
- **Avantages communaux** : avec la préparation du communiqué de presse, nous attendons que nos trois mandants puissent "vendre" cette idée à la commune de la Sonnaz pour compter avec leur soutien. La commune de la Sonnaz pourrait dans la mesure de possible offrir certains avantages pour les utilisateurs de l'application et d'une manière très spéciale pour le futur conducteur.(des bonnes essences, des places de parking gratuites, etc.)

## 10.6 Communiqué de presse

Un communiqué de presse contenant les éléments suivants, qui ont été décrits dans les chapitres précédents, sera livré aux mandants :

- Contexte, valeurs et problèmes résolus.
- Analyse concurrentielle.
- Identité visuelle finale.
- Motivation à l'utilisation.

Notre but étant de réussir à condenser ses informations dans un document court (maximum de une page) pour faciliter sa compréhension et lecture. C'est un document que nous pouvons partager à divers organes afin de les informer de manière rapide et concise du projet.

## 10.7 Idées de promotion locale

Nous avons réfléchi de manière concise à différents moyens de faire parler de notre plateforme afin de rassembler le plus d'utilisateurs possible sur celle-ci.

- Annonces par les enseignants de la commune à leurs élèves.
- Grande affiche à proximité des arrêts
- "Bumper Stickers"

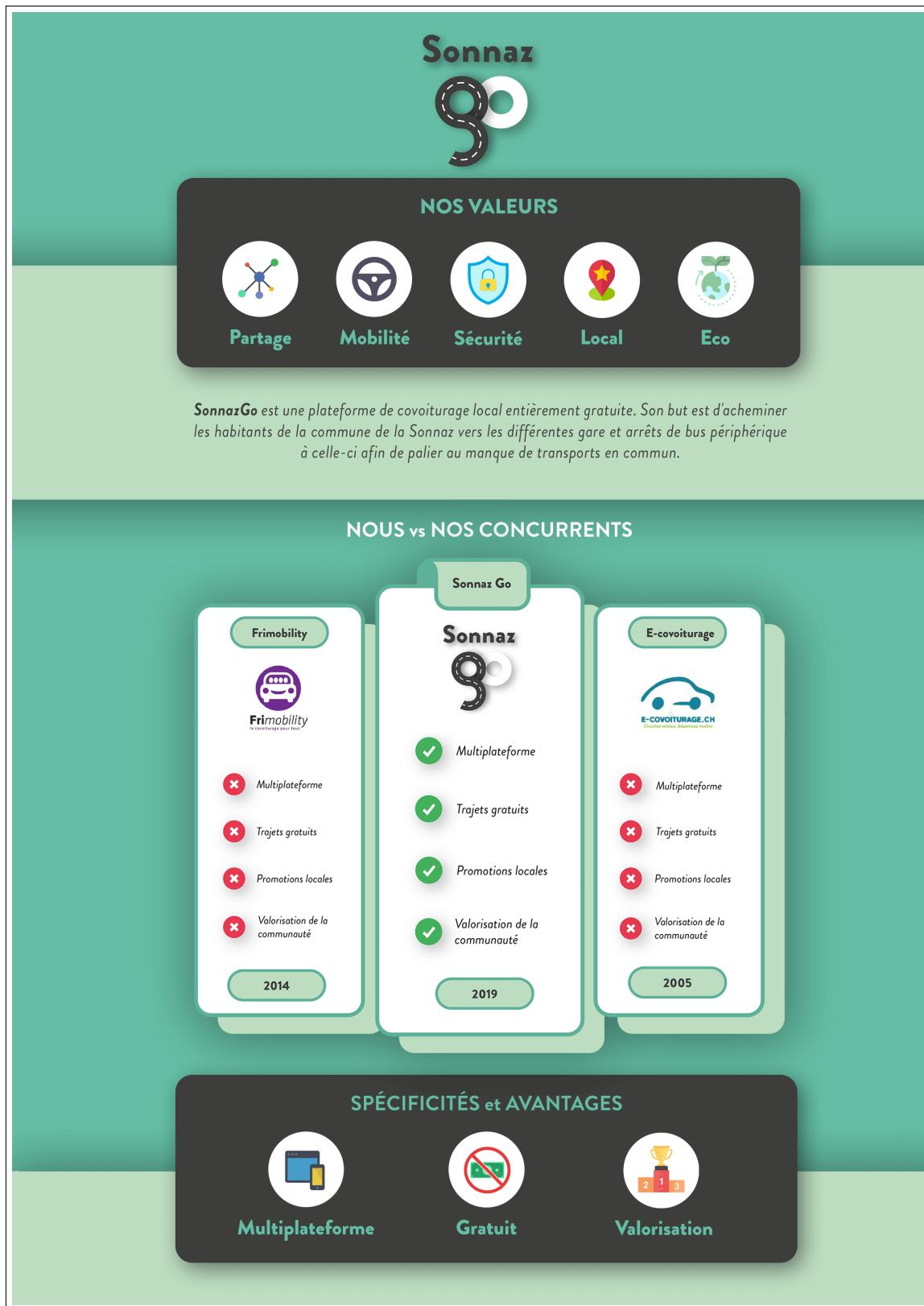
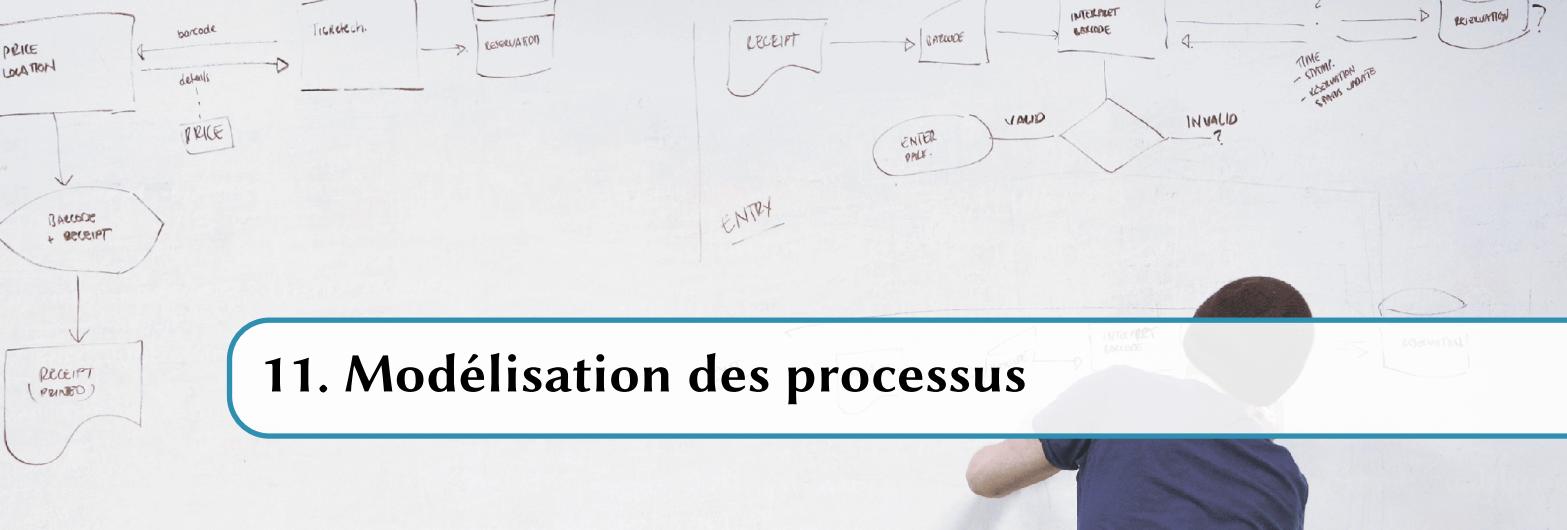


FIGURE 10.12 – Communiqué de presse Sonnaz Go

# Conception



<b>11</b>	<b>Modélisation des processus . . . . .</b>	<b>52</b>
11.1	Business process model and notation (BPMN)	
<b>12</b>	<b>Modélisation UML . . . . .</b>	<b>54</b>
12.1	Cas d'utilisation	
<b>13</b>	<b>Maquettes . . . . .</b>	<b>55</b>
13.1	Introduction	
13.2	Enregistrement et connexion	
13.3	Guide d'utilisation	
13.4	Page d'accueil conducteur	
13.5	Annonce d'un trajet (conducteur)	
13.6	Page d'accueil passager	
13.7	Réservation d'un trajet (passager)	
13.8	Modifications à tenir compte dans la réalisation	



## 11. Modélisation des processus

### 11.1 Business process model and notation (BPMN)

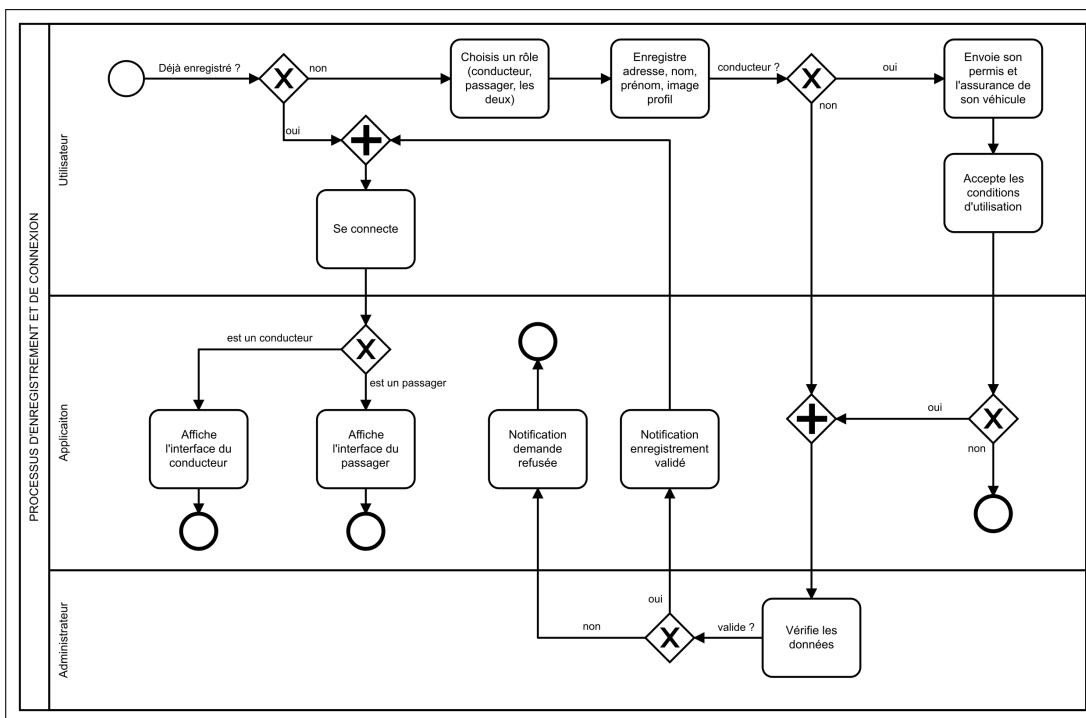


FIGURE 11.1 – BPMN du processus de connexion et d'enregistrement

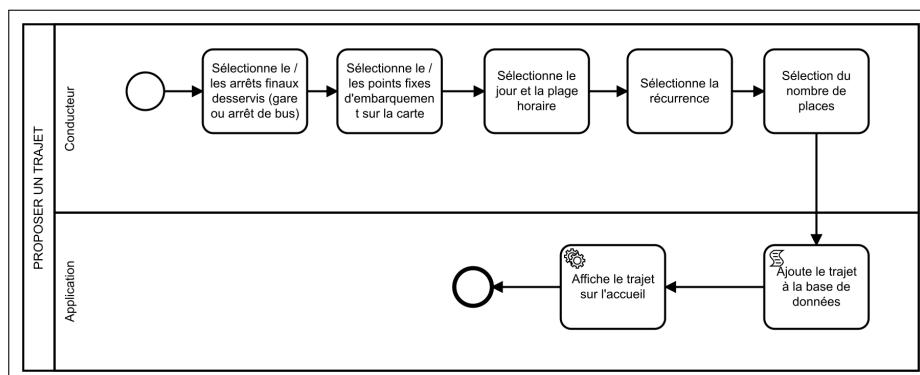


FIGURE 11.2 – BPMN du processus de proposition de trajet

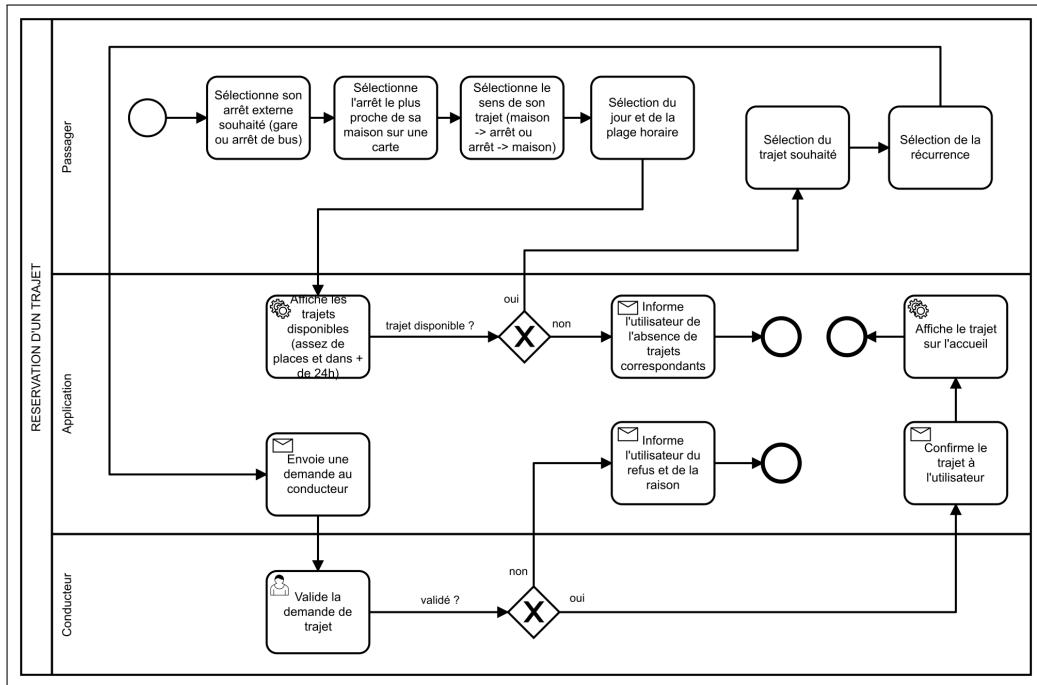


FIGURE 11.3 – BPMN du processus de réservation d'un trajet



## 12. Modélisation UML

### 12.1 Cas d'utilisation

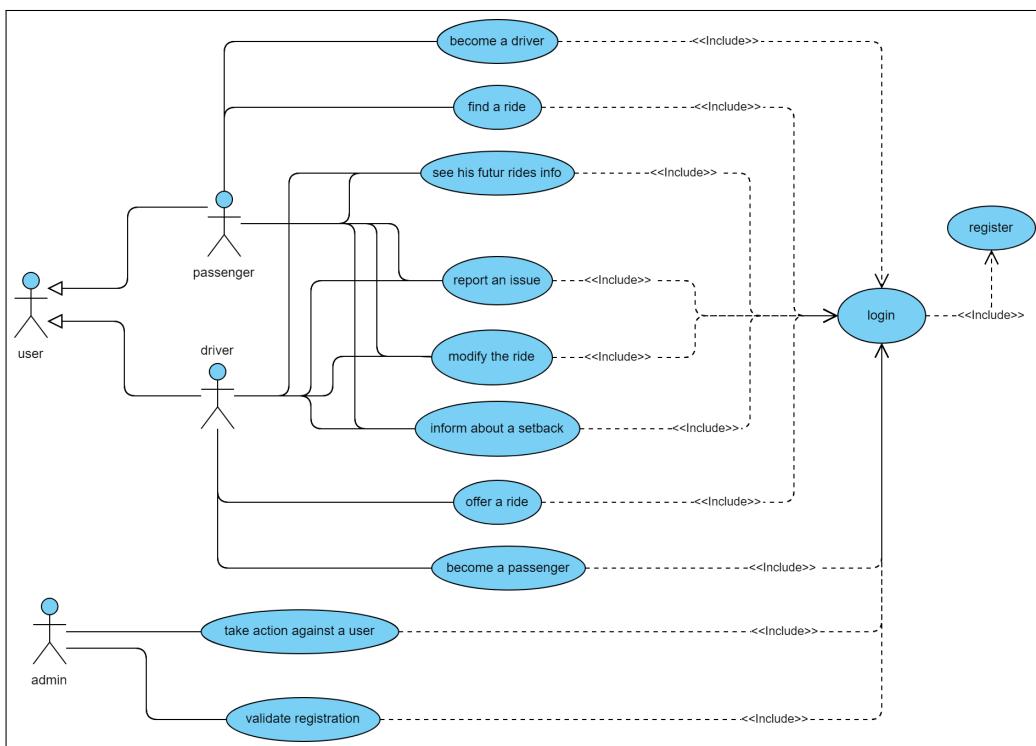


FIGURE 12.1 – Diagramme de cas d'utilisation

## 13. Maquettes

### 13.1 Introduction

Nous avons réalisé les maquettes avec l'outil Adobe XD. Ce choix s'est fait, car l'outil fait partie de l'environnement Adobe et fonctionne de pair avec les logiciels Adobe Photoshop et Adobe Illustrator que nous avons utilisés pour développer notre identité graphique.

Contrairement à d'autres logiciels de prototypage, Adobe XD offre la possibilité de faire des maquettes dynamiques et très "graphiques" représentant de manière réaliste et fidèle le produit final et permettant de proposer aux clients quelque chose d'attrayant.

Nous avons créé des maquettes "mobile first", car nos utilisateurs utiliseront l'application surtout de manière nomade.

### 13.2 Enregistrement et connexion

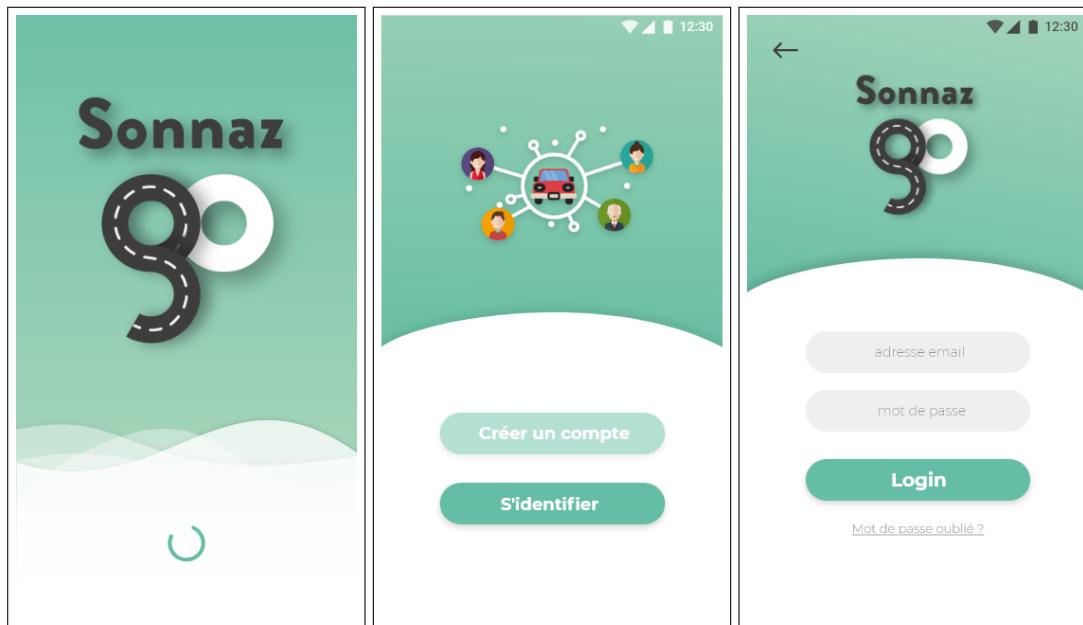


FIGURE 13.1 – Chargement, Enregistrement et Connexion

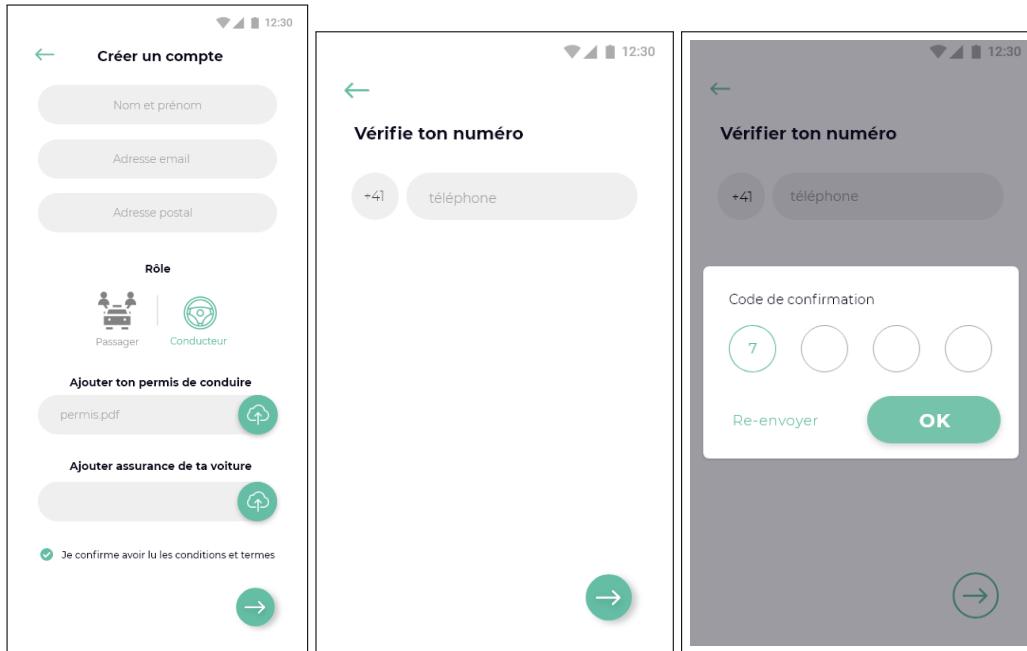


FIGURE 13.2 – Crée un compte (conducteur), Numéro de téléphone et Confirmation numéro de téléphone

### 13.3 Guide d'utilisation

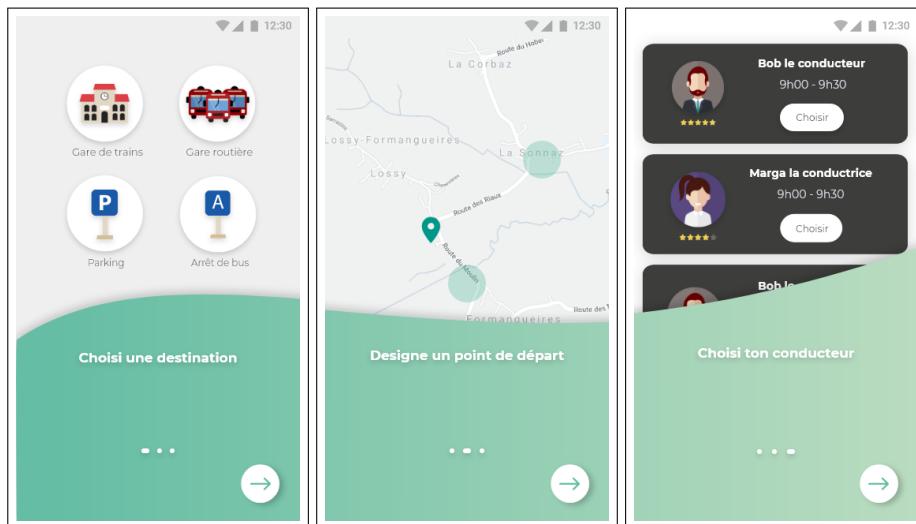


FIGURE 13.3 – Guide 1, 2 et 3

### 13.4 Page d'accueil conducteur

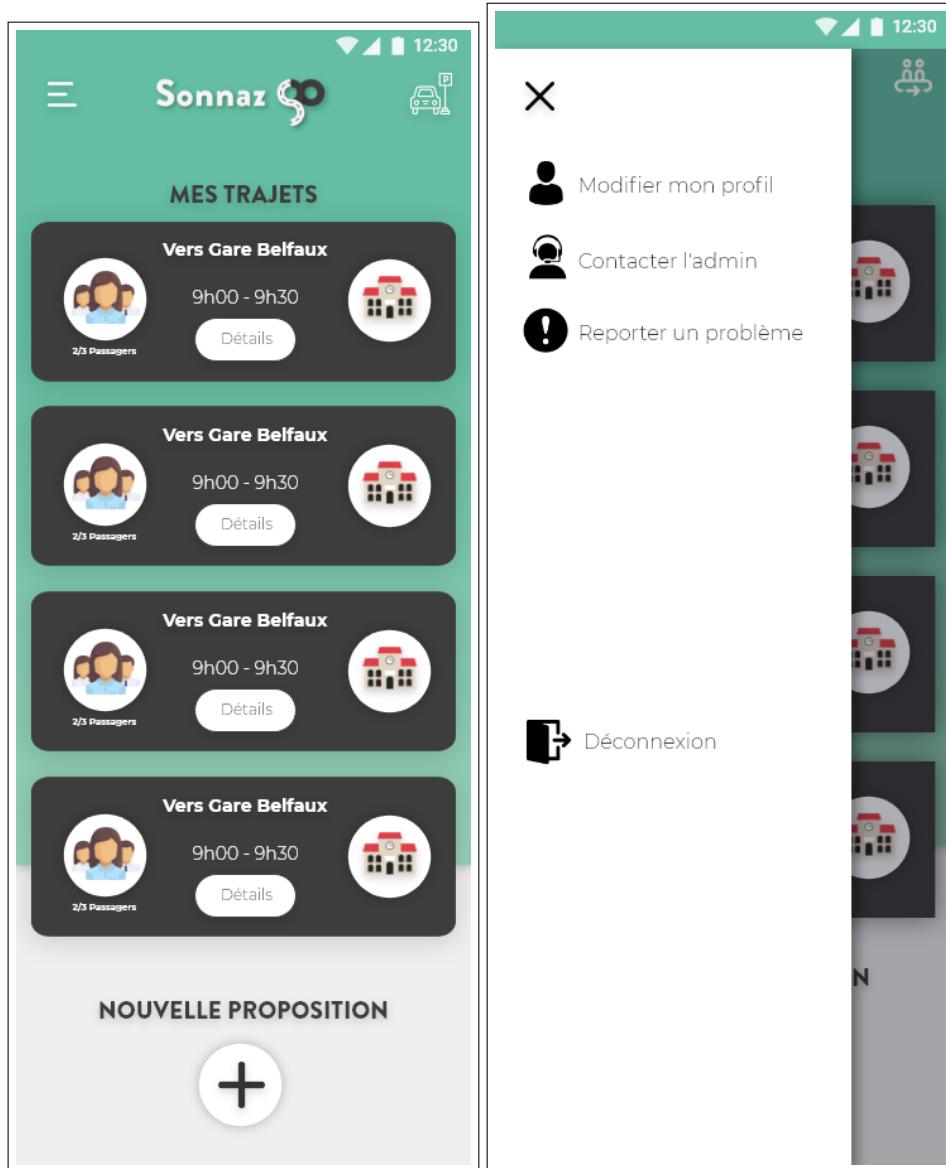


FIGURE 13.4 – Accueil conducteur et "Hamburguer" Menu

### 13.5 Annonce d'un trajet (conducteur)

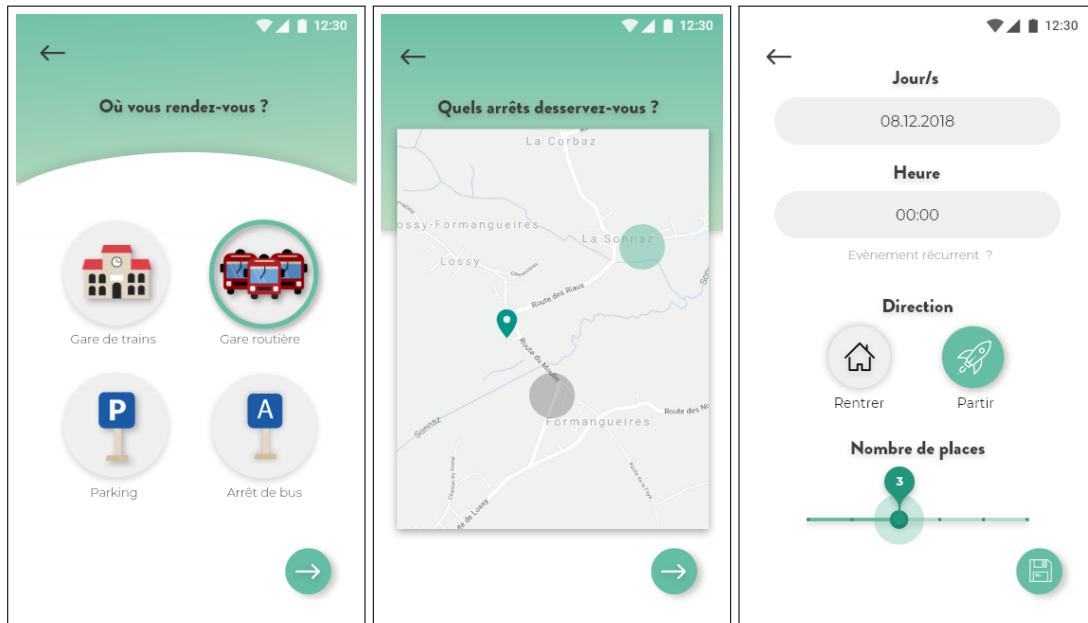


FIGURE 13.5 – Choix arrêt externe, interne et formulaire

### 13.6 Page d'accueil passager

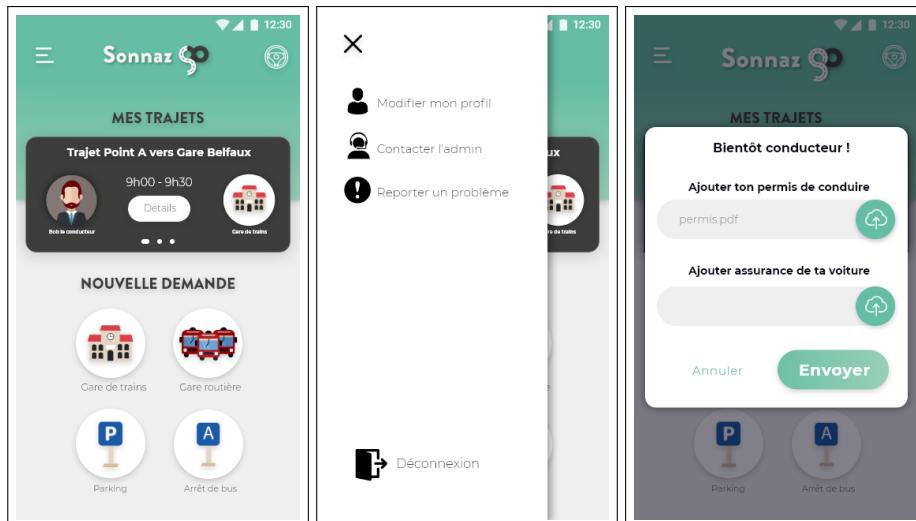


FIGURE 13.6 – Accueil passager, "Hamburguer" menu et Devenir conducteur

### 13.7 Réservation d'un trajet (passager)

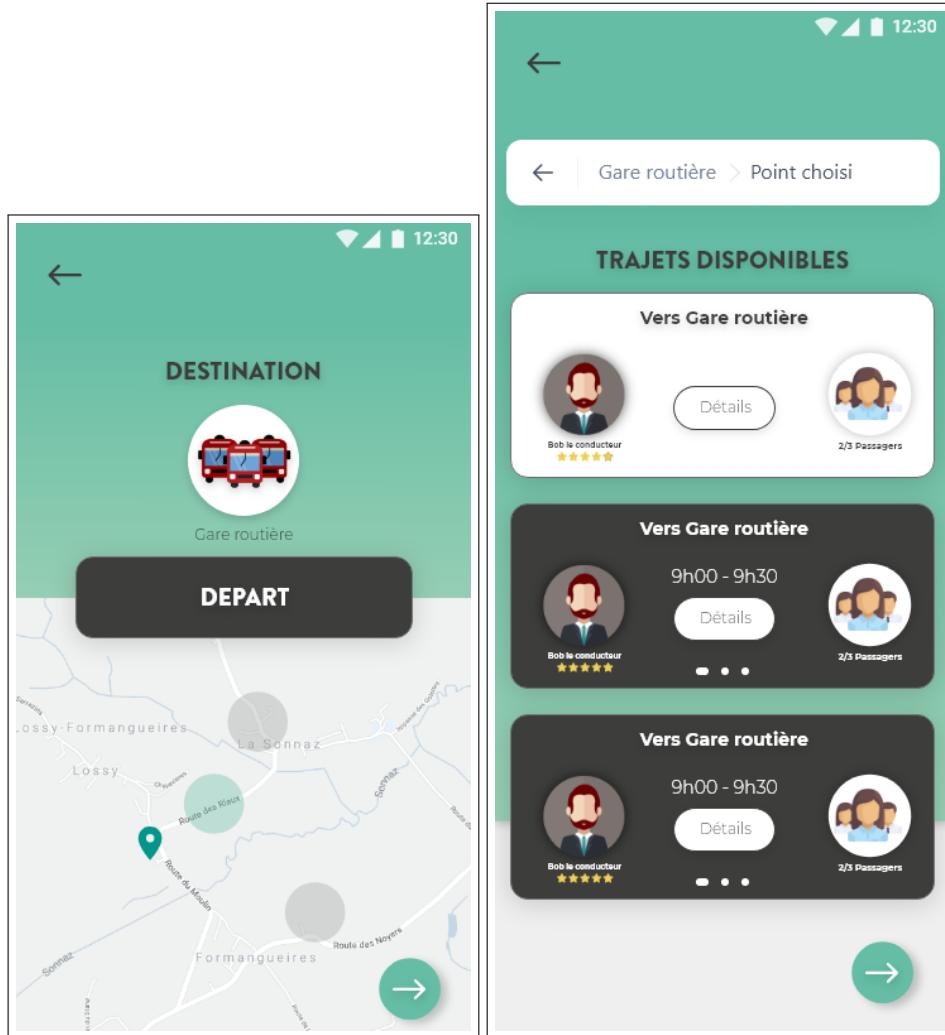


FIGURE 13.7 – Point d'embarquement et Choix du trajet

### 13.8 Modifications à tenir compte dans la réalisation

Suite à la présentation et au test de la maquette par un groupe de 5 personnes (responsables de projets ainsi que collègues), nous avons pu déduire que certaines choses seront à prendre en compte et à modifier dans la partie réalisation afin d'avoir une application ergonomique et qui correspond aux besoins des utilisateurs :

- Choix du sens du trajet avant tout le reste.
- Pas de guide d'utilisation.
- Attention à la cohérence, signification, dénominations entre les noms (embarquement, trajets, destination...)
- Les icônes pour passer du conducteur au passager et inversement ne sont pas assez explicites.



# Réalisation

<b>14</b>	<b>Notions essentielles de React .....</b>	<b>61</b>
14.1	DOM virtuel	
14.2	JSX	
14.3	Composants	
14.4	Tendance déclarative	
<b>15</b>	<b>Mise en place de l'environnement React .....</b>	<b>64</b>
15.1	Installation et utilisation de React	
15.2	Routing	
15.3	Service Worker	
15.4	Hosting Firebase	
<b>16</b>	<b>Extraits de code .....</b>	<b>70</b>
16.1	Dossiers actions, selectors, reducers et store	
16.2	index.js	
16.3	AddRidePage5.js	
16.4	Code réutilisé dans plusieurs composants	
16.5	AddRideItem.js	
16.6	DirectionChoice.js	
16.7	RidesList.js	
<b>17</b>	<b>GUI et CSS .....</b>	<b>73</b>
17.1	Style	
17.2	Responsive	
<b>18</b>	<b>Redux .....</b>	<b>77</b>
18.1	Pourquoi avons-nous besoin de Redux ?	
18.2	Comment mettre en place Redux ?	
<b>19</b>	<b>Carte interactive .....</b>	<b>82</b>
19.1	Introduction	
19.2	Customisation de Mapbox	
19.3	Implémentation de Mapbox	
19.4	Dépendance	
19.5	License	



## 14. Notions essentielles de React

### 14.1 DOM virtuel

React ne crée pas de HTML et n’interagit pas avec le Document Object Model (DOM) (interface de programmation normalisée par le W3C), qui permet à des scripts d’examiner et de modifier le contenu du navigateur web)<sup>1</sup>. Du navigateur, le DOM virtuel de React nous permet donc d’interagir avec, de manipuler et de modifier le DOM HTML du navigateur qui est représenté sous forme d’arbre (les éléments HTML sont des noeuds de l’arbre).

Modifier cet arbre peut être laborieux, car en cas de modifications, il faut entièrement recalculer le style et la disposition de tous les éléments. Pour pallier cet inconvénient, React implémente donc un concept de virtuel DOM afin d’alléger l’arbre. Ce sont des éléments React, légers, sans état et immuable qui constituent une représentation virtuelle d’un élément DOM, c’est un DOM virtuel<sup>2</sup>. cela se matérialise comme cela : Le virtuel DOM va comparer la page HTML finale au DOM virtuel et il va en déduire les opérations minimales à effectuer pour y arriver. Schéma ci-dessous<sup>3</sup>

Cette implémentation nous permet d’améliorer grandement les performances, de permettre un pré rendu côté serveur et de pouvoir la décliner sur d’autres supports. Toutes les méthodes liées à la manipulation du DOM du navigateur grâce à React sont documentées ici (par exemple méthode render() pour rendre un élément React dans le DOM HTML) : <https://reactjs.org/docs/react-dom.html>

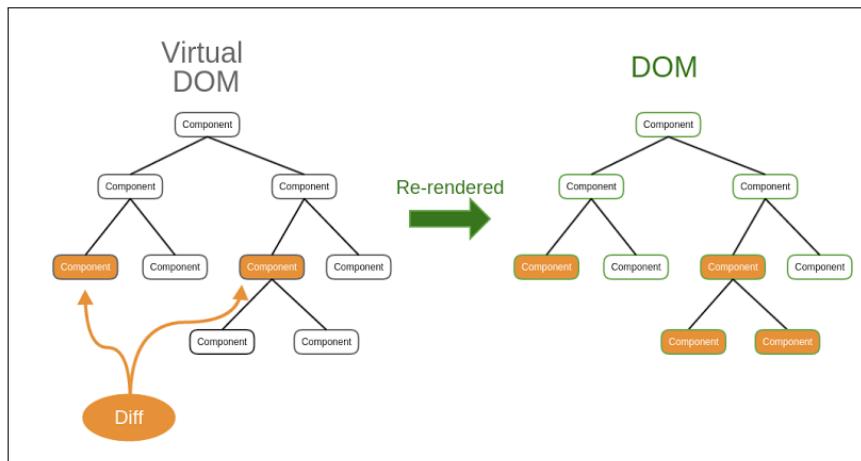


FIGURE 14.1 – Schématisation du DOM virtuel

1. [https://fr.wikipedia.org/wiki/Document\\_Object\\_Model](https://fr.wikipedia.org/wiki/Document_Object_Model)

2. <https://gist.github.com/sebmarkbage/fcb1b6ab493b0c77d589>

3. [https://miro.medium.com/max/928/1\\*CqdIWZy0NMPQhYx2rKzo9g.png](https://miro.medium.com/max/928/1*CqdIWZy0NMPQhYx2rKzo9g.png)

## 14.2 JSX

C'est une extension de syntaxe JavaScript créée spécialement pour React et utilisée avec le DOM virtuel. JSX est "simplement" un langage de balises comme le sont HTML et XML, mais à la différence qu'il est case sensitive et qu'il produit des éléments React<sup>4</sup>.

Il est recommandé de l'utiliser avec React afin de décrire à quoi l'Interface Utilisateur (UI) ressemblera. Il permet de résoudre le problème des appels "render()" qui peuvent vite être trop verbeux et difficiles à lire dans une syntaxe plus traditionnelle et de ressembler un maximum au HTML final qui va être rendu par React pour être mis dans le DOM HTML.

Il nécessite malheureusement une étape supplémentaire qui est la transpilation (prendre le code source d'un langage de programmation et le compiler dans un autre langage de programmation<sup>5</sup>). Nous utiliserons pour cela Babel : <https://babeljs.io/>

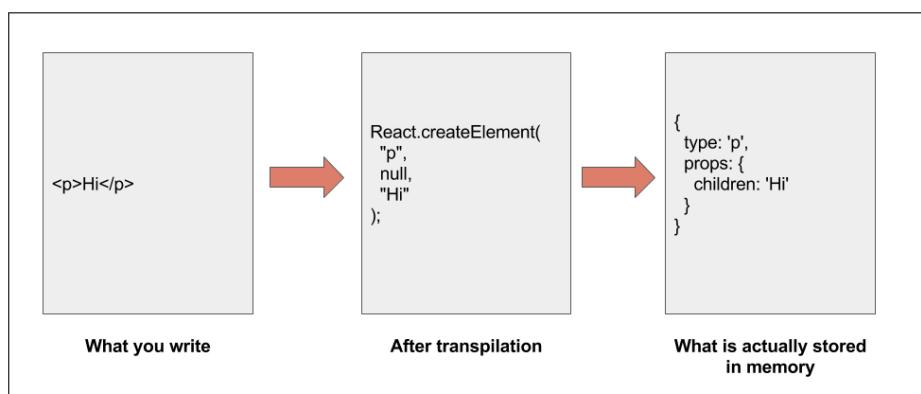


FIGURE 14.2 – Schématisation du fonctionnement de JSX

## 14.3 Composants

React se base sur l'utilisation de composants. Ce sont, conceptuellement, des fonctions JavaScript. Les composants sont des micro-entités indépendantes et autonomes., ils acceptent des paramètres (props) et retournent des éléments React qui décrivent ce qui doit apparaître à l'écran. Cela nous permet de diviser l'UI en différentes petites pièces indépendantes et réutilisables que nous pouvons concevoir en isolation.

React possède deux types de composants :

### 14.3.1 Composants stateless (ou fonctionnel)

Fonction qui permet uniquement de recevoir optionnellement des objets en paramètre (props) et de rendre des éléments JSX. On peut le modifier que par l'envoi de props, impossible de le modifier depuis "l'extérieur". Le composant stateless va utiliser les props pour générer l'UI, mais il lui est impossible de les modifier.

4. [https://cdn-images-1.medium.com/max/1600/1\\*ighKXxBnnSd1aOr5-ZOPg.png](https://cdn-images-1.medium.com/max/1600/1*ighKXxBnnSd1aOr5-ZOPg.png)

5. [https://fr.wikipedia.org/wiki/Compilateur\\_source\\_%C3%A0\\_source](https://fr.wikipedia.org/wiki/Compilateur_source_%C3%A0_source)

### 14.3.2 Composants stateful (ou à état)

Classe qui reçoit des props en paramètre, possède obligatoirement une méthode render(), un constructeur, retourne du JSX et possède un état interne du composant (state). L'état détermine comment le composant se comporte et est rendu, il permet de créer des composants dynamiques et interactifs. Nous pouvons modifier cet état avec un simple appel à la fonction setState().

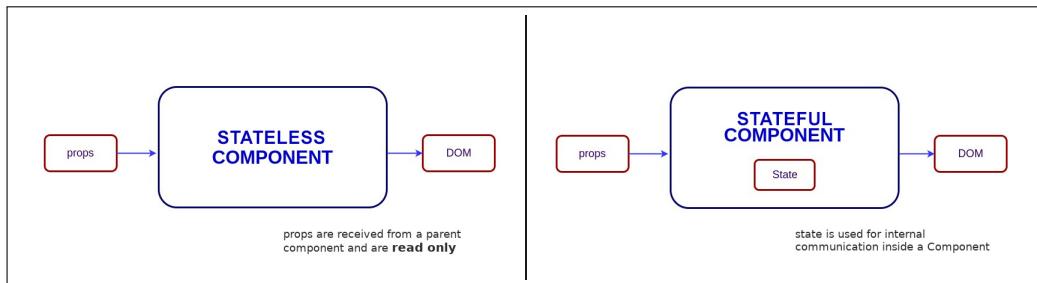


FIGURE 14.3 – Résumé des deux types de composants React

Les composants peuvent soit recevoir des données sous forme de props ou matérialiser leur propre état (state) et le gérer au fil du temps (life-cycle).

## 14.4 Tendance déclarative

“ La programmation déclarative est un paradigme de programmation. Elle consiste à créer des applications sur la base de composants logiciels indépendants du contexte et ne comportant aucun état interne. Autrement dit, l'appel d'un de ces composants avec les mêmes arguments produit exactement le même résultat, quels que soient le moment et le contexte de l'appel.<sup>6</sup> ”

*Wikipedia*

En d'autres mots, la programmation déclarative (HTML) décrit "quoi faire" et non "comment faire les choses" comme en programmation impérative (C, Java).

6. [https://fr.wikipedia.org/wiki/Programmation\\_déclarative](https://fr.wikipedia.org/wiki/Programmation_déclarative)

```
userLink = {
  ...
  secondaryLink,
  dren,
  userAvatar,
  ne,
  (
    in className={styles.container}>

  includeAvatar && [
    <UserDetailsCardOnHover
      user={user}
      delay={CARD_HOVER_DELAY}
      wrapperClassName={styles.avatarContainer}
    >
      <Avatar user={user} />
    </UserDetailsCardOnHover>
  ]
}

div
className={classNames(
  styles.LinkContainer,
  inline && styles.inlineContainer
)}
)

<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
  <Link
    to={pathnames.buildUserUrl(user)}>
    className={classNames(styles.name, {
      [styles.alt]: type === 'att',
      [styles.centerName]: secondaryLink,
      [styles.link]: !secondaryLink
    })}
  </Link>
  ...
  renderWhatsNewList()
  return [
    <div className={styles.list}>
      <ul className={styles.ul}>
        {this.renderListItems()}
      </ul>
    </div>
  ]
)

```

15. Mise en place de l'environnement React

## 15.1 Installation et utilisation de React

### 15.1.1 Prérequis

- **Node.js** : <https://nodejs.org/fr/download>
  - **NPM** (gestionnaire de packages de Node)

Mise à jour de NPM : `npm install -global npm`

### 15.1.2 create-react-app

<https://facebook.github.io/create-react-app/docs/getting-started>

Create React APP est la méthode officielle pour créer des applications React. L'outil génère un squelette de l'application (boilerplate) à partir duquel nous pourrons commencer le développement. Il offre une couche d'abstraction supérieure en pré configurant et en cachant des outils comme le service-worker, le JavaScript moderne (ES6 et JSX traduisent avec Babel), bundling de l'application (Webpack) et le serveur de développement.

## Démarrage rapide :

```
npm create-react-app my-app  
cd my-app  
npm start
```



FIGURE 15.1 – Structure du projet

On accède à notre application via l'URL : `http://localhost:3000/`

La commande "npm start" permet de démarrer le serveur de développement. La page va être automatiquement rafraîchie lorsque des changements sont faits sur le code et les erreurs de build s'afficheront dans la console.

```
Failed to compile.

./src/App.js
Syntax error: Unexpected token (8:7)

  6 |   render() {
  7 |     return (
>  8 |       <><div className="App">
    |         ^
  9 |         <header className="App-header">
 10 |           <img src={logo} className="App-logo" alt="logo" />
 11 |           <h1 className="App-title">Welcome to React</h1>
```

FIGURE 15.2 – Exemple d'erreur du serveur de développement dans la console

## 15.2 Routing

Le routing est très important dans React qui utilise le concept de "single page application". Cela permet à l'utilisateur de croire qu'il se situe sur un site "standard" en lui offrant une navigation basée sur différentes URL qu'il peut partager, mettre en favoris.

React Router permet d'afficher des composants en fonction de l'URL du navigateur.

### 15.2.1 Mise en place

<https://www.npmjs.com/package/react-router-dom> <https://reacttraining.com/react-router/web/guides/quick-start>

**Installation :** `npm install react-router-dom`

**Ajouter dans index.js :**

```
import BrowserRouter from "react-router-dom";
```

**Ajouter dans App.js pour définir les routes :**

```
import Route, Switch from "react-router-dom";
```

**Ajouter dans les fichiers des composants pour utiliser les routes :**

```
import NavLink from "react-router-dom";
```

### 15.2.2 BrowserRouter

**“** A `<Router>` that uses the HTML5 history API (`pushState`, `replaceState` and the `popstate` event) to keep your UI in sync with the URL.<sup>1</sup> **”**

*React Router*

Nous "encapsulons" notre application (App) dans un `<BrowserRouter>` afin de pouvoir y utiliser et configurer des routes par la suite.

```
ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById("root")
);
```

FIGURE 15.3 – index.js

### 15.2.3 Route

**“** The Route component is perhaps the most important component in React Router to understand and learn to use well. Its most basic responsibility is to render some UI when a location matches the route's path.<sup>2</sup> **”**

*React Router Route*

L'élément `<Route>` nous permet de définir un chemin (URL) associé à un composant.

- **Switch** : regroupe les routes, il itère ses éléments fils et render seulement ceux qui matchent le path courant.
- **Path** : chemin d'accès au composant de la Route, match toutes les routes commençantes par le même "path".
- **Exact** : Permet de matcher exactement le "path" au lieu de toutes les routes commençant par le même.
- **Component** : Render un composant React lors de l'accès au "path" correspondant.
- **Error** : Composant à afficher en cas de non-connaissance du chemin (équivalent de l'erreur HTTP 404). Se définit en décrivant une route sans "path".

1. <https://reacttraining.com/react-router/web/api/BrowserRouter>

2. <https://reacttraining.com/react-router/web/api/Route>

```
const App = () => {
  return (
    <div>
      <Switch className="App">
        <Route path="/" component={Home} exact />
        <Route path="/driver" component={Driver} exact />
        <Route path="/passenger" component={Passenger} exact />
        <Route path="/test" component={Test} exact />
        <Route component={Error} />
      </Switch>
    </div>
  );
};
```

FIGURE 15.4 – App.js

#### 15.2.4 Link

“ Provides declarative, accessible navigation around your application. `<NavLink>` is a special version of the `<Link>` that will add styling attributes to the rendered element when it matches the current URL.<sup>3</sup> ”

*React Router Link and NavLink*

```
<NavLink to="/driver" className="home-item">
  <Button
    floating
    large
    className="home-button"
    icon="directions_car"
  />
  <div className="home-label">Conducteur</div>
</NavLink>
```

FIGURE 15.5 – Exemple de lien sur un bouton et un label

#### 15.3 Service Worker

Les détails sur le fonctionnement des service worker ont été définis dans la section 6.4, `create-react-app` nous permet d'en enregistrer un très facilement grâce à cette ligne à modifier dans le fichier `Index.js` :

```
// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: http://bit.ly/CRA-PWA
serviceWorker.register();
```

FIGURE 15.6 – Enregistrement d'un service worker

3. <https://reacttraining.com/react-router/web/api/NavLink>

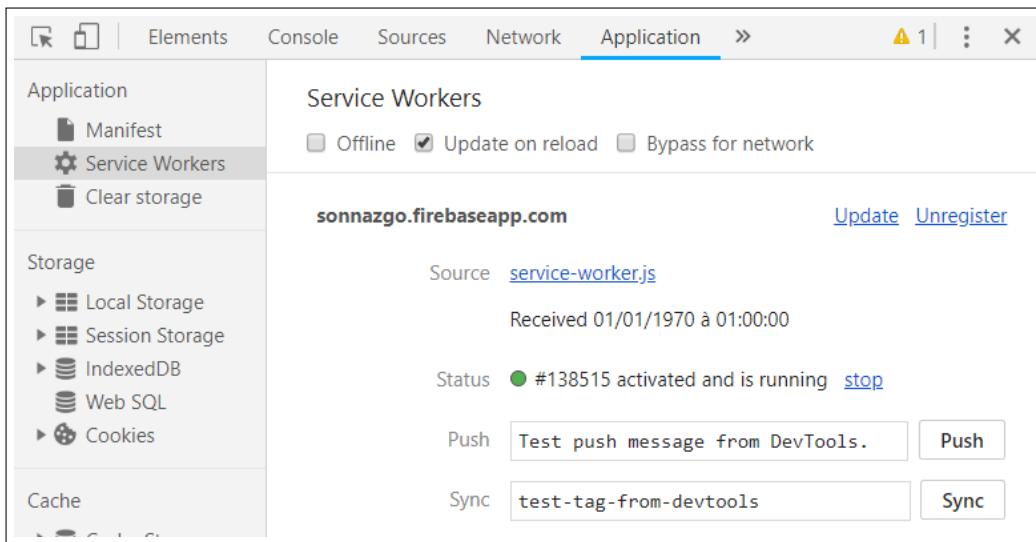


FIGURE 15.7 – Validation que le service worker est en place et fonctionne

### 15.3.1 Fonctionnement hors ligne

L'un des buts d'un service worker est de permettre d'accéder à une page même si nous ne possédons pas de connexion à internet.

Il faut faire attention à plusieurs choses pour que ce fonctionnement se produise comme attendu :

- Les polices d'écritures doivent être stockées en local (pas de href)
- Les packs d'icônes doivent être stockées en local (pas de href)
- Les images être stockées en local (pas de lien)

### 15.3.2 Customisation

Par défaut, le service worker défini par create-react-app fonctionne en mode "cache first"<sup>4</sup>. dans certains cas, il est préférable d'avoir un fonctionnement "network first" si nous avons des ressources qui se mettent souvent à jour.

Pour customiser le service worker, un utilitaire existe : sw-precache-cra<sup>5</sup>.

Il nous permet, au moment de "build" notre projet de définir des paramètres dans un fichier sw-config.js.

4. <https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/>

5. <https://github.com/liuderchi/sw-precache-cra/>

```

module.exports = {
  ...,
  runtimeCaching: [
    {
      urlPattern: /\.*/,
      handler: "networkFirst"
    },
    {
      urlPattern: "/",
      handler: "networkFirst"
    }
  ]
};

```

FIGURE 15.8 – Fichier sw-config définissant un fonctionnement "network first" pour le service worker

#### 15.4 Hosting Firebase

Nous avons besoin d'héberger notre application web sur une URL publique. En effet, cela est nécessaire afin de la faire tester son rendu et son fonctionnement sur divers support et tester le bon fonctionnement du service worker. Nous l'avons rendu possible en déployant le prototype sur un service d'hébergement.

Nous avons choisi pour cela la plateforme Firebase<sup>6</sup> de Google, et plus spécialement leur service de hosting<sup>7</sup>. Notre choix s'est porté sur ce service, car il propose un hébergement complètement gratuit, une mise en place extrêmement simple et rapide pour une application web React, un certificat SSL par défaut et de bonnes performances.

C'est ainsi que nous avons rendu notre application web accessible via l'url suivante : <https://sonnazgo.firebaseio.com/>.

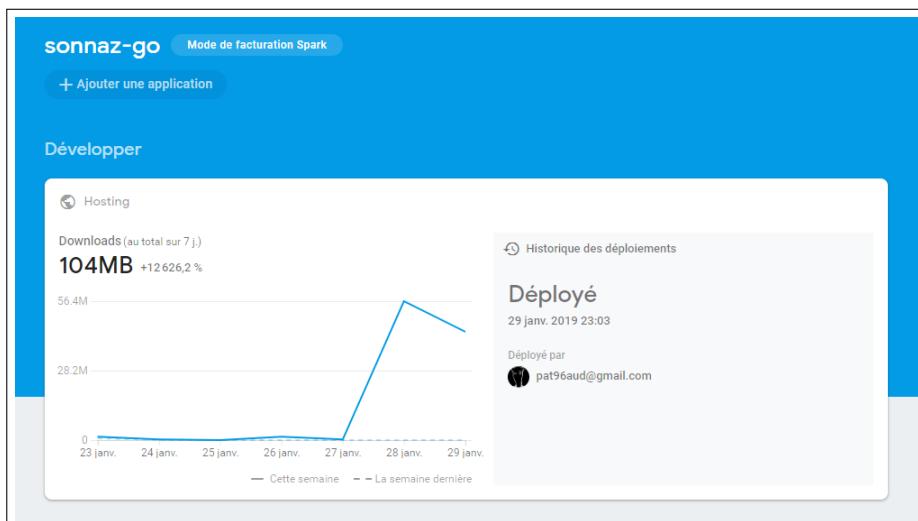


FIGURE 15.9 – Tableau de bord Firebase

6. <https://firebase.google.com/docs/hosting/quickstart>

7. <https://firebase.google.com/>

```

userLink = {
  ...
  secondaryLink,
  dren,
  dudeAvatar,
  ne,
  ...
  in className={styles.container}>
    ...
  </UserDetailsCard>
  ...
  includeAvatar && (
    <UserDetailsCardOnHover
      user={user}
      delay={CARD_HOVER_DELAY}
      wrapperClassName={styles.avatarContainer}>
      ...
      <Avatar user={user} />
    </UserDetailsCardOnHover>
  )
}

div
className={classNames(
  styles.LinkContainer,
  inline && styles.inlineContainer
)}
}

<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
  <Link
    to={{ pathname: buildUserUrl(user) }}
    className={classNames(styles.name, {
      [styles.alt]: type === 'alt',
      [styles.centerName]: !secondaryLink,
      [styles.secondaryName]: secondaryLink
    })}
  >
    ...
  </Link>
  ...
  renderWhatsNewLinks() {
    ...
    return (
      <div className={styles.footer}>
        ...
        <ul className={classNames(styles.list, {
          [styles.noMargin]: !trackTitle(url)
        })}>
          ...
          <li key={index}>
            {title}
            ...
          </li>
        </ul>
      </div>
    );
  }
  ...
  renderFooterSub() {
    ...
  }

```

## 16. Extraits de code

Dans ce chapitre nous voulons simplement faire référence à quelques extraits de notre code qui nous semblent intéressants, pour une compréhension plus approfondie de nos composants il faudra se référer aux commentaires dans le code.

### 16.1 Dossiers actions, selectors, reducers et store

Dans ces dossiers nous pouvons observer l'utilisation de redux pour la création et suppression d'un trajet, ainsi que pour la définition des filtres employés dans notre application.(voir commentaires dans le code pour plus d'information)

### 16.2 index.js

Le composant `<Provider />` fait que le store de Redux soit disponible dans tous les composants imbriqués qui ont été enveloppés dans la fonction `connect()`. Étant donné que tout composant React d'une application React Redux peut être connecté, la plupart des applications affichent un rendu `<Provider>` au niveau supérieur, avec l'intégralité de l'arborescence des composants de l'application, c'est pour ceci que nous avons utilisé ce composant à cette endroit en particulier.

```

const jsx = (
  <Provider store={store}>
    <App />
  </Provider>
);
ReactDOM.render.jsx, document.getElementById("root"));

```

FIGURE 16.1 – Utilisation du composant Provider

### 16.3 AddRidePage5.js

Nous passons les valeurs stockées dans le store de redux tout au long du processus de ajout d'un trajet dans le composant `RideForm` pour remplir le formulaire et créer un nouveau trajet.

```
<RideForm
  driver={this.props.filters.driverName}
  direction={this.props.filters.direction}
  outside={this.props.filters.outsideValue}
  inside={this.props.filters.insideValue}
  places={this.props.filters.places}
  createdAt={this.props.filters.createdAt}
  onSubmit={ride => {
    this.props.dispatch(addRide(ride));
    this.props.history.push("/driver");
  }}
/>
```

FIGURE 16.2 – Props pour la création d'un nouveau trajet

#### 16.4 Code réutilisé dans plusieurs composants

Nous allons utiliser le code suivant pour "reset" le scroll de la page à zéro.

```
componentDidMount() {
  window.scrollTo(0, 0);
};
```

FIGURE 16.3 – Utilisation de scrollTo

#### 16.5 AddRideItem.js

Nous utilisons ce code pour l'affichage de la variable createdAt qui contient un string avec l'heure et la date des trajets. Pour ceci nous utilisons le package momentjs.

```
<p className="white-text driver-item-label-t">
  <span className="white-text driver-item-label">
    {moment(this.props.createdAt).format("ddd, Do MMMM YYYY")}
  </span>
</p>
<span className="white-text driver-item-label">
  <Icon className="white-text driver-item-icon">access_time</Icon>
  {moment(this.props.createdAt).format("HH:mm")}
</span>
```

FIGURE 16.4 – Utilisation de momentjs

## 16.6 DirectionChoice.js

Affichage conditionnelle du button s'il a été clické ou non. Nous utilisons la même logique dans le composant OutsideChoice.js.

```
{this.state.direction === "home" ? (
  <Fab value="home" classes={{ root: classes.buttonfocused }}>
    <Icon>check</Icon>
  </Fab>
) : (
  <Fab
    value="home"
    onClick={this.handleClick}
    classes={{ root: classes.button }}
  >
    <Icon>home</Icon>
  </Fab>
)}
```

FIGURE 16.5 – Affichage conditionnelle du button

## 16.7 RidesList.js

Affichage conditionnelle des informations sur les trajets en fonction du props passé par le composant parent DriverPage.js ou PassengerPage.js au composant fils RidesLists.js.

```
<RidesList role={this.state.role} />
```

FIGURE 16.6 – Composant parent

```
{props.role === "driver" &&
  (props.rides.length ? (
    props.rides.map(ride => {
      return <RideListItemDriver key={ride.id} {...ride} />;
    })
  ) : (
    <div className="driver-label no-rides">Aucun trajet</div>
  )));
{props.role === "passenger" &&
  (props.rides.length ? (
    props.rides.map(ride => {
      return <RideListItemPassenger key={ride.id} {...ride} />;
    })
  ) : (
    <div className="driver-label no-rides">Aucun trajet</div>
  )));
...}
```

FIGURE 16.7 – Composant fils

```

userLink = {
  ...
  secondaryLink,
  dren,
  dudeAvatar,
  ne,
  ...
  in className={styles.container}>
    ...
  </UserDetailsCard>
  includeAvatar && [
    <UserDetailsCardOnHover
      user={user}
      delay={CARD_HOVER_DELAY}
      wrapperClassName={styles.avatarContainer}>
      ...
      <Avatar user={user} />
    </UserDetailsCardOnHover>
  ]
}

div
className={classNames(
  styles.LinkContainer,
  inline && styles.inlineContainer
)}

<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
  <Link
    to={{ pathname: buildUserUrl(user) }}
    className={classNames(styles.name, {
      [styles.alt]: type === 'alt',
      [styles.centerName]: !secondaryLink,
      [styles.secondaryName]: secondaryLink
    })}
  >
    ...
  </Link>
  renderWhatsNewLinks() {
    ...
    return (
      <div className={styles.footer}>
        <h4>What's new</h4>
        <ul>
          ...
        </ul>
      </div>
    );
  }
}

renderCenterSub() {
  ...
  return (
    <div className={styles.footer}>
      <h4>What's new</h4>
      <ul>
        ...
      </ul>
    </div>
  );
}

```

## 17. GUI et CSS

### 17.1 Style

Pour styliser notre applications, nous avons utilisé plusieurs moyens :

#### 17.1.1 Framework CSS

Nous avons expliqué l'utilité des Framework CSS dans le [Chapitre 10.3](#).

Nous avons initialement utilisé le portage React de Materialize<sup>1</sup> qui offre des composants React pour le framework Materialize<sup>2</sup> mais nous nous sommes vite rendu compte que il n'implémentait pas toutes les fonctionnalités de Materialize et que son utilisation était mal documenté pour une utilisation poussée et que ses capacités étaient limitées.

Face à une impasse pour afficher certains éléments graphiques selon notre convenance et l'incapacité de leur associer des props, nous nous sommes tournées vers le framework Material-UI<sup>3</sup> qui s'est avéré être bien plus adapté à nos besoins quoi que étant plus difficile à stylisé.

Nous avons malheureusement, à cause de ce virage effectué en cours de conception, un code qui utilise Materialize React pour certains éléments graphiques (dont la grille) et Material-UI pour d'autres (boutons, formulaire...)



FIGURE 17.1 – Logo Material-UI

1. <https://react-materialize.github.io/#/>
2. <https://materializecss.com/>
3. <https://material-ui.com/getting-started/installation/>

### 17.1.2 SASS

Nous avons écrit nos feuilles de style en utilisant SASS<sup>4</sup> (Syntactically Awesome Stylesheets) au lieu du CSS. C'est un préprocesseur CSS permettant plusieurs options intéressantes mais nous n'allons pas nous étaler sur le sujet. Nous l'avons surtout utilisé pour séparer notre CSS en différentes feuilles de styles correspondant à nos différents composants React.



```

        ▾ □ styles
          ▾ □ base
            _base.scss
          ▾ □ components
            _addbook.scss
            _driver.scss
            _loginPage.scss
            _map.scss
            _menu.scss
            _notfound.scss
            _passenger.scss
  •   91  /*
  92  -----
  93  COMPONENTS CSS
  94  -----
  95  */
  96
  97 .driver-wrapper {
  98   | box-shadow: 0px 5px 71px -11px □rgba(0, 0, 0, 0.48);
  99 }
  100

```

FIGURE 17.2 – Fichiers SCSS dans notre structure de fichier

### 17.1.3 Personnaliser les framework CSS

Les frameworks que nous utilisons possèdent de nombreux composants correspondants aux directives du material design. Nous avons par contre besoin de les modifier un peu afin d'avoir un rend fidèle à notre maquette.

#### Materialize

Il suffit de redéfinir les classes utilisées par Materialize ou de lui en définir de nouvelles et de les styliser dans une feuille de style.

4. <https://sass-lang.com/>

### Material-UI

Material-UI fonctionne différemment, il faut injecter du CSS dans le DOM directement via le composant React.<sup>5</sup>

```
// We can inject some CSS into the DOM.
const styles = {
  root: {
    background: 'linear-gradient(45deg, #FE6B8B 30%, #FF8E53 90%)',
    borderRadius: 3,
    border: 0,
    color: 'white',
    height: 48,
    padding: '0 30px',
    boxShadow: '0 3px 5px 2px rgba(255, 105, 135, .3)',
  },
};

function ClassNames(props) {
  const { classes, children, className, ...other } = props;

  return (
    <Button className={classNames(classes.root, className)} {...other}>
      {children || 'class names'}
    </Button>
  );
}
```

FIGURE 17.3 – Exemple de personnalisation d'un bouton Material-UI

## 17.2 Responsive

### 17.2.1 Grid Materialize

**“** The Material Design responsive layout grid adapts to screen size and orientation, ensuring consistency across layouts.<sup>6</sup> **”**

*Material-UI*

Pour simplifier la mise en page et avoir une application avec un rendu "responsive" (qui s'adapte à la taille de l'écran sur lequel l'application web est affichée), nous utilisons le système de grille de Materialize<sup>7</sup>. La chose étant très commune et répandue, nous ne nous étalerons pas plus sur le sujet.

Il est juste bon de préciser que la grille Materialize utilise un système à 12 colonnes et qu'elle est "mobile first". On crée un "container" dans lequel nous définissons des "row" et nous leur attribuons un certain nombre de colonnes en fonction de la taille du device utilisé. Pour plus d'informations, se référer à la documentation Materialize sur le sujet.

5. <https://material-ui.com/customization/overrides/>

6. <https://material-ui.com/layout/grid/>

7. <https://materializecss.com/grid.html>

```
const RidesList = props => (
  <div className="driver-rides">
    <div className="container">
      <div className="row">
        <div className="col s12 m10 l10 offset-l1 offset-m1">
          <div className="driver-label ">MES PROCHAINS TRAJETS</div>
        </div>
      </div>
    </div>
  </div>
)
```

FIGURE 17.4 – Exemple de mise en place de la grille sur la plage d'accueil du conducteur

### 17.2.2 Media Queries

Les media queries nous permettent elles de modifier le CSS de l'application en fonction de la taille de l'écran du device sur lequel l'application est affichée. Un exemple générique de sa mise en place est présent ci-dessus : FIGURE 17.5<sup>8</sup>.

```
/* Set the background color of body to tan */
body {
  background-color: tan;
}

/* On screens that are 992px or less, set the background color to blue */
@media screen and (max-width: 992px) {
  body {
    background-color: blue;
  }
}

/* On screens that are 600px or less, set the background color to olive */
@media screen and (max-width: 600px) {
  body {
    background-color: olive;
  }
}
```

FIGURE 17.5 – Exemple simple de media queries

8. [https://www.w3schools.com/css/css3\\_mediaqueries\\_ex.asp](https://www.w3schools.com/css/css3_mediaqueries_ex.asp)

```

userLink = {
  secondaryLink,
  dren,
  dudeAvatar,
  ne,
  {
    in className={styles.container}>
      includeAvatar && [
        <UserDetailsCardOnHover
          user={user}
          delay={CARD_HOVER_DELAY}
          wrapperClassName={styles.avatarContainer}>
        <Avatar user={user} />
      </UserDetailsCardOnHover>
    ]
  }
}

div
className={classNames(
  styles.LinkContainer,
  inline && styles.inlineContainer
)})

<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
<Link
  to={{ pathname: buildUserUrl(user) }}
  className={classNames(styles.name, {
    [styles.alt]: type === 'alt',
    [styles.centerName]: !secondaryLink,
    [secondaryLink]: !dren
  })}>

```

## 18. Redux

“ Redux est une bibliothèque open source JavaScript de gestion d'état pour applications web. Il est le plus couramment utilisé avec des bibliothèques comme React ou Angular pour la construction d'interfaces utilisateur. Semblable à (et inspiré de) l'architecture Flux, il a été créé par Dan Abramov et Andrew Clark.

”

*Wikipedia*



FIGURE 18.1 – Logo Redux

“ Redux est un conteneur d'état prévisible pour les applications JavaScript.

”

*Redux Documentation*

Redux vous aide à écrire des applications cohérentes, fonctionnant dans différents environnements (client, serveur et natif) et faciles à tester. En plus de cela, il fournit une grande expérience de développement, telle que l'édition de code en temps réel combiné à un débogueur temporel.

Nous pouvons utiliser Redux avec React ou avec toute autre bibliothèque JavaScript. En termes de taille il pèse seulement 2 Ko, mais il dispose d'un vaste écosystème de fonctionnalités disponibles.

### 18.1 Pourquoi avons-nous besoin de Redux ?

Dans notre application nous avons besoin d'utiliser Redux, car nous n'avons pas la possibilité de passer des états entre composants frères et il nous est impératif d'utiliser cette fonctionnalité pour l'implémentation de la création d'un trajet par exemple.

Nous avons également absolument besoin de la fonctionnalité "store"<sup>1</sup> de Redux nous permettant de retenir tout l'arbre d'état (state tree) de notre application. Cela nous permet d'avoir des trajets pré-enregistrées et d'en ajouter d'autres en n'ayant pas de back-end.(voir FIGURE 18.2 )

1. <https://redux.js.org/api/store>

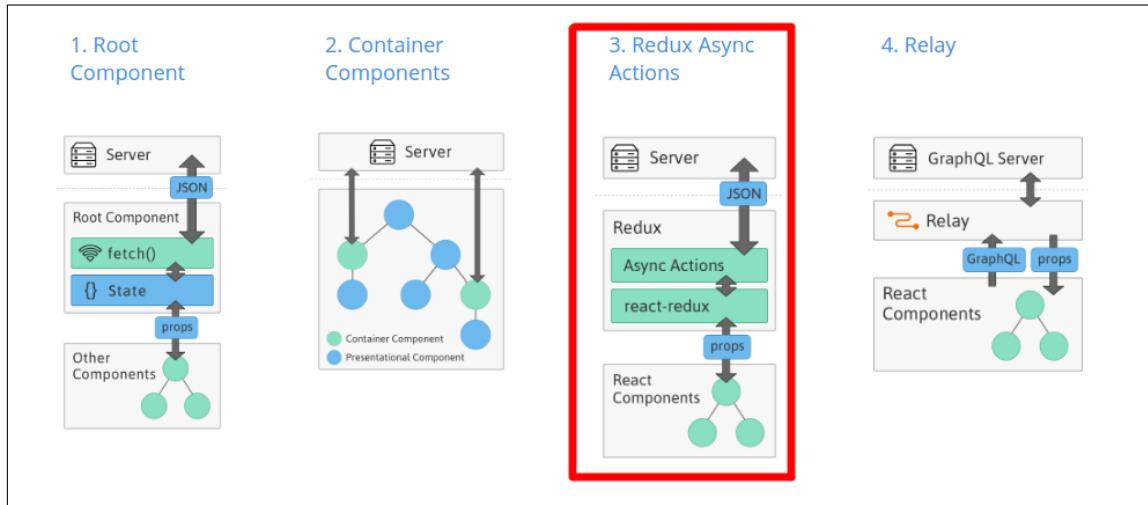


FIGURE 18.2 – Illustration du fonctionnement de React avec Redux

## 18.2 Comment mettre en place Redux ?

Redux dispose d'une documentation assez complète et très claire, qui nous aide à suivre le processus de mise en place avec l'exemple d'une application TODO. Vous trouver plus de détail sur le site <https://redux.js.org>.

Dans cette partie nous allons vous expliquer de manière très concise comment mettre en place le minimum nécessaire pour pouvoir utiliser Redux avec React.

Comme pré requis principal, il faudra avoir une structure d'un projet React déjà mise en place. Pour cela `create-react-app` fait très bien l'affaire.

Suite à ça nous devons tout simplement installer Redux avec la commande `npm install redux` qui va nous permettre d'installer la dernière version du Redux.

Une fois cela mis en place, nous pouvons procéder à l'intégration de Redux avec React. Dans le petit exemple qui suit, nous allons vous expliquer les quatre étapes pour réussir une telle intégration.

Nous allons suivre les 4 étapes suivantes :

1. Créer un "Store".
2. Créer un "Reducer".
3. Utiliser la méthode "Subscribe".
4. Utiliser la méthode "Dispatch".

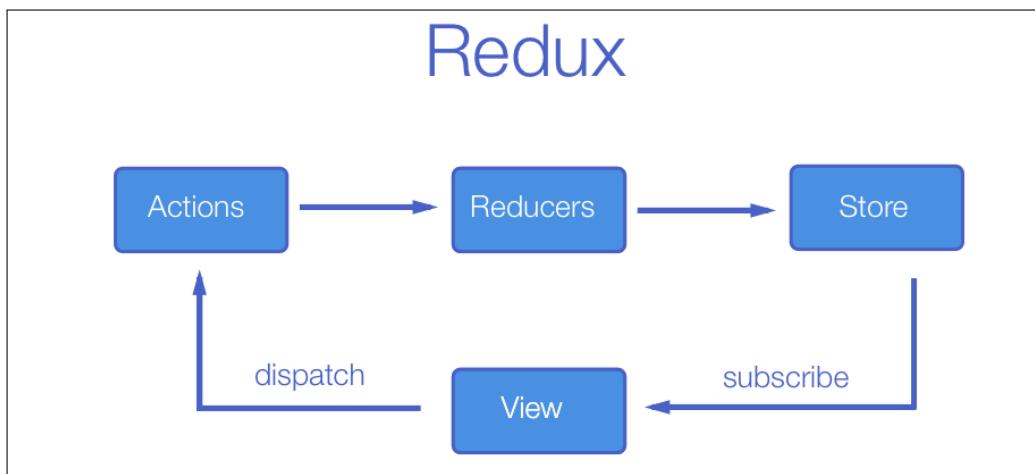


FIGURE 18.3 – Illustration du fonctionnement de Redux

Avant de nous embarquer dans l'intégration de Redux avec React, nous avons besoin de connaître quelques termes propres à Redux.

- **Action** : Les actions sont des données utiles (payload) qui envoient des données de votre application à votre Store. Elles sont la seule source d'information pour le Store. Nous pouvons les envoyer au Store en utilisant `store.dispatch()`.
- **Reducer** : Les reducers (réducteurs) spécifient comment l'état de l'application change en réponse aux actions envoyées au magasin(Store). Comme les actions ne décrivent que ce qui s'est passé, les reducers vont changer l'état de l'application par rapport aux directives dictées par les actions.
- **Store** : Le magasin (Store) est l'objet qui rassemble les actions et les reducers. Il a les responsabilités suivantes :
  - Maintenir l'état de l'application;
  - Permettre l'accès au "state" via `getState()`;
  - Permettre à l'état(state) d'être mis à jour via `dispatch(action)`;
  - Inscrir les listeners via `subscribe(listener)`

Une fois vues les notions basiques de Redux, nous allons expliquer un peu plus en détail les quatre étapes mentionnées précédemment.

### 18.2.1 Étape 1 : Créer un "Store"

Pour cette première étape, nous avons besoin d'importer la méthode `createStore` directement de Redux, ce qui va nous permettre de créer une nouvelle Store.

```
import { createStore } from 'redux'; //méthode nécessaire pour l'étape 1
```

FIGURE 18.4 – Import de la méthode createStore()

Dans la figure ci-dessous nous pouvons voir un code minimaliste pour la création d'un "Store".

```
//Etape 1: Créer un Store prends en paramètre un reducer et un état prédefini
const store = createStore(reducer,"Etat");
```

FIGURE 18.5 – Crédit du Store

### 18.2.2 Étape 2 : Créer un "Reducer"

Dans la figure ci-dessous nous pouvons voir un code minimaliste pour la création d'un "Reducer".

```
//Etape 2: Créer un Reducer prends state(etat) et action comme paramètres
const reducer = function (state, action) {
    if (action.type === "NOUVELLE_ETAT") {
        return action.payload
    }
    if (action.type === "UN_AUTRE_ETAT") {
        return action.payload
    }
    //si pas de changement d'état
    return state;
}
```

FIGURE 18.6 – Crédit d'un reducer

### 18.2.3 Étape 3 : Utiliser la méthode "Subscribe"

Dans la figure ci-dessous nous pouvons voir un code minimaliste pour l'utilisation de la méthode "Subscribe" qui nous permet de récupérer la valeur actuelle de notre état dans le Store.

```
//Etape 3: Subscribe
store.subscribe(() => {
  console.log("Votre nouvelle état est: ", store.getState());
});
```

FIGURE 18.7 – Utilisation de la méthode subscribe()

### 18.2.4 Étape 4 : Utiliser la méthode "Dispatch"

Dans la figure ci-dessous nous pouvons voir un code minimaliste pour l'utilisation de la méthode "Dispatch".

```
//Etape 4: Dispatch qui prend une action et un payload(données utiles) comme paramètres
store.dispatch({type: "NOUVELLE_ETAT", payload: "Etat 1"});
store.dispatch({type: "UN_AUTRE_ETAT", payload: "Etat 2"});
```

FIGURE 18.8 – Utilisation de la méthode dispatch()

**R** *Dans le but d'expliquer d'une manière simplifiée le fonctionnement de Redux nous avons codé toutes les étapes décrites précédemment dans un seul fichier, mais il est conseillé de répartir ces étapes dans des fichiers différents. Dans la figure suivante nous pouvons observer la manière dont laquelle nous avons implémenté ces étapes dans notre application.*

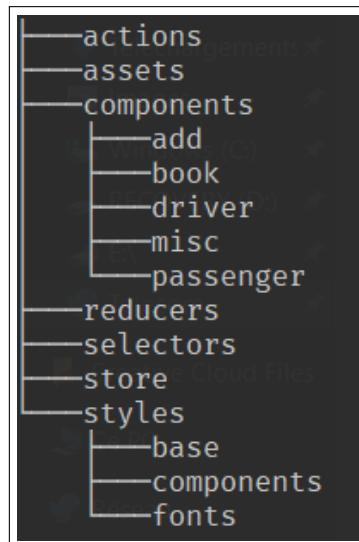


FIGURE 18.9 – Répartition des étapes de création Redux dans notre application

```

userLink = {
    secondaryLink,
    dren,
    userAvatar,
    ne,
    {
      className={styles.container}
    }
  }

  includeAvatar && (
    <UserDetailsCardOnHover
      user={user}
      delay={CARD_HOVER_DELAY}
      wrapperClassName={styles.avatarContainer}
    >
      <Avatar user={user} />
    </UserDetailsCardOnHover>
  )
}



<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
  <Link
    to={{ pathname: buildUserUrl(user) }}
    className={classNames(styles.name, {
      [styles.alt]: type === 'alt',
      [styles.centerName]: !secondaryLink,
      [styles.secondaryLink]: secondaryLink
    })}
  >
    {title}
  </Link>
</UserDetailsCardOnHover>


```

## 19. Carte interactive

### 19.1 Introduction

Nous avons besoin d'un système de carte interactive afin de visualiser les différents arrêts à l'intérieur de la commune et de les sélectionner. Notre choix s'est porté sur Mapbox<sup>1</sup>, c'est un service de cartes en ligne proposant une personnalisation approfondie ainsi qu'une API afin de l'intégrer sans notre application web. Il est entre autres utilisé par Uber, Facebook, Snapchat.



FIGURE 19.1 – Logo de Mapbox

Nous souhaitions initialement utiliser Google Maps Platform<sup>2</sup>. Pour intégrer un système de carte interactive dans notre application. Le service ayant une meilleure documentation et une plus grande communauté. Nous avons dû trouver une alternative, car Google Maps Platform requiert l'enregistrement d'une carte de crédit pour utiliser leur API. Mapbox était une bonne alternative, relativement bien documenté, compatible avec React, entièrement gratuit jusqu'à une certaine limite de connexion par jours, mais un peu plus complexe à mettre en place.

Notre carte interactive doit posséder les fonctionnalités suivantes :

- Géolocalisation (pour que l'utilisateur ait une idée de sa distance par rapport aux arrêts)
- Zoom, dézoom, déplacement, recentrer
- Emplacement des arrêts externes indiqués au moyen de symboles non interactifs
- Emplacement des arrêts internes au moyen de marqueurs interactifs
- Possibilité de sélectionner des arrêts internes

Pour une question de manque de temps, nous avons choisi de permettre le choix que d'un seul arrêt. À terme, il faudra qu'un conducteur puisse choisir plusieurs arrêts à desservir et le passager un seul arrêt comme point d'embarcation/de dépôt.

1. <https://www.mapbox.com/>

2. <https://cloud.google.com/maps-platform/?hl=fr>

## 19.2 Customisation de Mapbox

Mapbox offre sur leur site la possibilité de customiser une carte tant au niveau esthétique qu'au niveau des informations qu'elle contient.

C'est grâce à cet outil nommé Mapbox Studio<sup>3</sup><https://studio.mapbox.com/> que nous avons créé une carte respectant nos codes visuels et ayant les différents arrêts externes mis en évidence avec des symboles.

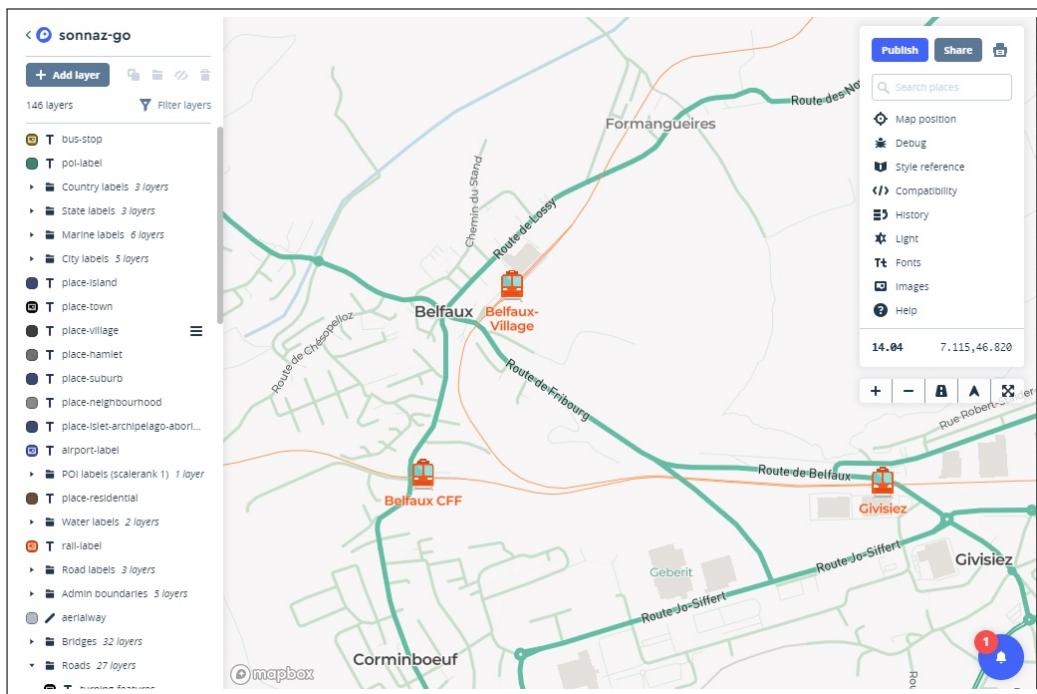


FIGURE 19.2 – Mapbox Studio

## 19.3 Implémentation de Mapbox

Nous avons utilisé la documentation officielle de Mapbox afin d'assurer l'implémentation : <https://docs.mapbox.com/mapbox-gl-js/overview/>

Mapbox s'installe facilement comme un package NPM : `npm i mapbox-gl`

L'implémentation commentée est dans le fichier `Map.js` dans le dossier `src/components/misc`.

La structure du fichier est la suivante :

- Appel à la clé pour la connexion l'API Mapbox (crée sur le compte Mapbox)
- Définition des props liées à l'utilisation de la carte (longitude, latitude, zoom, arrêts)
- Appel de la fonction `componentDidMount` (appelé après qu'un composant est monté) qui "construit" la carte
- Dans cette méthode, définition des marqueurs de localisation interne dans une variable `geojson`
- Crédit de la carte en soi et définition de son style (grâce au style, crée en FIGURE 19.2 )

3. \unskip\penalty\@M\vrulewidth\z@height\z@depth\dp ,

- Placement des différents marqueurs, on leur donne à chacun une action et un popup déclenché par un click.

#### 19.4 Dépendance

Il est important de souligner que dans notre implémentation, nous sommes malheureusement dépendants d'un service de carte interactive pour notre application web de covoiturage. Il faut être conscient que le service devient payant si le nombre d'accès à la carte dépasse des 50'000 par mois. (tarification : <https://www.mapbox.com/pricing/>), mais cela ne devrait pas poser de problème dans l'immédiat, l'application étant limitée à une petite commune d'environ 1000 habitants.

Si le service venait à fermer, il faudrait également trouver une alternative, se tourner vers Google Maps Platform par exemple.

Cette dépendance pourrait être un obstacle dans le futur, car nous dépendons d'une source externe sur laquelle nous n'avons aucun pouvoir. Nous avons prouvé dans l'immédiat, grâce à notre projet, que c'est une solution viable et fonctionnelle.

#### 19.5 License

Mapbox est distribué sous license "3-Clause BSD license"<sup>4</sup>. Cette license nous autorise toutes modifications du codes source, distribution et utilisation commerciale et nécessite la présence du copyright visible.<sup>5</sup>

---

4. <https://github.com/mapbox/mapbox-gl-js/blob/master/LICENSE.txt>

5. <https://opensource.org/licenses/BSD-3-Clause>

# VI

## Test et validation

<b>20</b>	<b>Test empirique .....</b>	<b>86</b>
20.1	Test de 5 secondes	
20.2	Test d'utilisabilité	

```

userLink = {
    secondaryLink,
    dren,
    userAvatar,
    ne,
    {
      className={styles.container}
    }
}

includeAvatar && [
  <UserDetailsCardOnHover
    user={user}
    delay={CARD_HOVER_DELAY}
    wrapperClassName={styles.avatarContainer}
  >
    <Avatar user={user} />
  </UserDetailsCardOnHover>
]

div
  className={classNames(
    styles.LinkContainer,
    inline && styles.inlineContainer
  )}

```

```

<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
  <Link
    to={{ pathname: buildUserUrl(user) }}
    className={classNames(styles.name, {
      [styles.alt]: type === 'alt',
      [styles.centerName]: !secondaryLink,
      [styles.secondaryName]: secondaryLink
    })}
  >
    {title}
    </a>
  </Link>
  <div
    className={styles.footer}
    style={{ position: 'absolute', bottom: 0, left: 0, width: 100% }}
  >
    <UserFooterSub>{title}</UserFooterSub>
  </div>
</UserDetailsCardOnHover>

```

## 20. Test empirique

“ Empirique : Qui s'appuie sur l'expérience, l'observation et non sur la théorie.<sup>1</sup>

*Dictionnaire*

”

Nous avons fait faire des tests à un panel de 5 utilisateurs afin de recueillir des informations visant à améliorer et peaufiner notre application. Nous avons à chaque fois testé la version mobile ainsi que la version web.

### 20.1 Test de 5 secondes

Un test de 5 secondes permet de recueillir la première impression que les utilisateurs se font de manière spontanée d'une application.

A la suite de ce test, cinq questions étaient posées aux testeurs :

- A quoi sert l'application présentée ?
- A qui est destinée l'application ?
- Que pensez-vous de l'esthétique de l'application ?
- Quel est le nom de l'application web ?
- Faites un croquis de l'interface que vous venez de voir

#### A quoi sert l'application présentée ?

Les réponses n'ont pas été très variées, elles tournent toutes autour de la notion de "proposer et réserver des trajets" ou "aller d'un pour A à un point B en voiture". La notion d'arrêts est aussi ressorti. La plus part des testeurs ont réussi à déduire qu'il s'agissait d'une application de covoiturage.

Ce test est donc concluant et répond à nos attentes.

#### A qui est destinée l'application ?

Ici des réponses plus mitigées, personne n'a réussi à deviner que l'application était destiné à un public très restreint (la commune de la Sonnaz). Les réponses étaient dans la veine de "des personnes ayant recours au covoiturage".

Il est selon nous pas nécessaire que cela soit flagrant que l'application soit réservée aux habitants de la commune de la Sonnaz, l'application sera de toute façon distribuée de manière restreinte. Le nom de l'application possédant le nom de la commune dedans est suffisant selon nous.

1. <https://www.linternaute.fr/dictionnaire/fr/definition/empirique/>

### Que pensez-vous de l'esthétique de l'application ?

Beaucoup de compliments ont été reçus à cette question : "joli, propre, stylé, bon contraste, lisible". Il a été fait remarqué que le bouton d'ajout de trajet devrait être visible directement sans avoir à défiler au fond de la page. Nous sommes également déçu que la notion d'écologie ne soit pas ressortie. Une personne aurait souhaité voir le cartes blanches plutôt que anthracite pour améliorer la lisibilité.

### Quel est le nom de l'application web ?

Trois personnes ont retenu "Sonnaz Go", une a dis "Sonnaz... quelque chose" et la dernière n'a pas fait attention au nom. Nous sommes un peu mitigé, nous aurions espéré que notre logo (contenant le nom de l'application) attire plus l'oeil et soit bien retenu et mémorisé. C'est peut être dû au fait qu'il s'intègre "trop" bien dans l'interface et qu'il passe inaperçu.

### Faites un croquis de l'interface que vous venez de voir

Toutes les personnes ont redessiné avec brio notre interface ! Ont voit bien que ce qui ressort pour les testeurs sont les "cartes" contenant les informations sur le trajet. C'est justement l'élément sur lequel nous souhaitons focaliser l'attention. Le bouton pour ajouter un trajet est également jamais oublié.

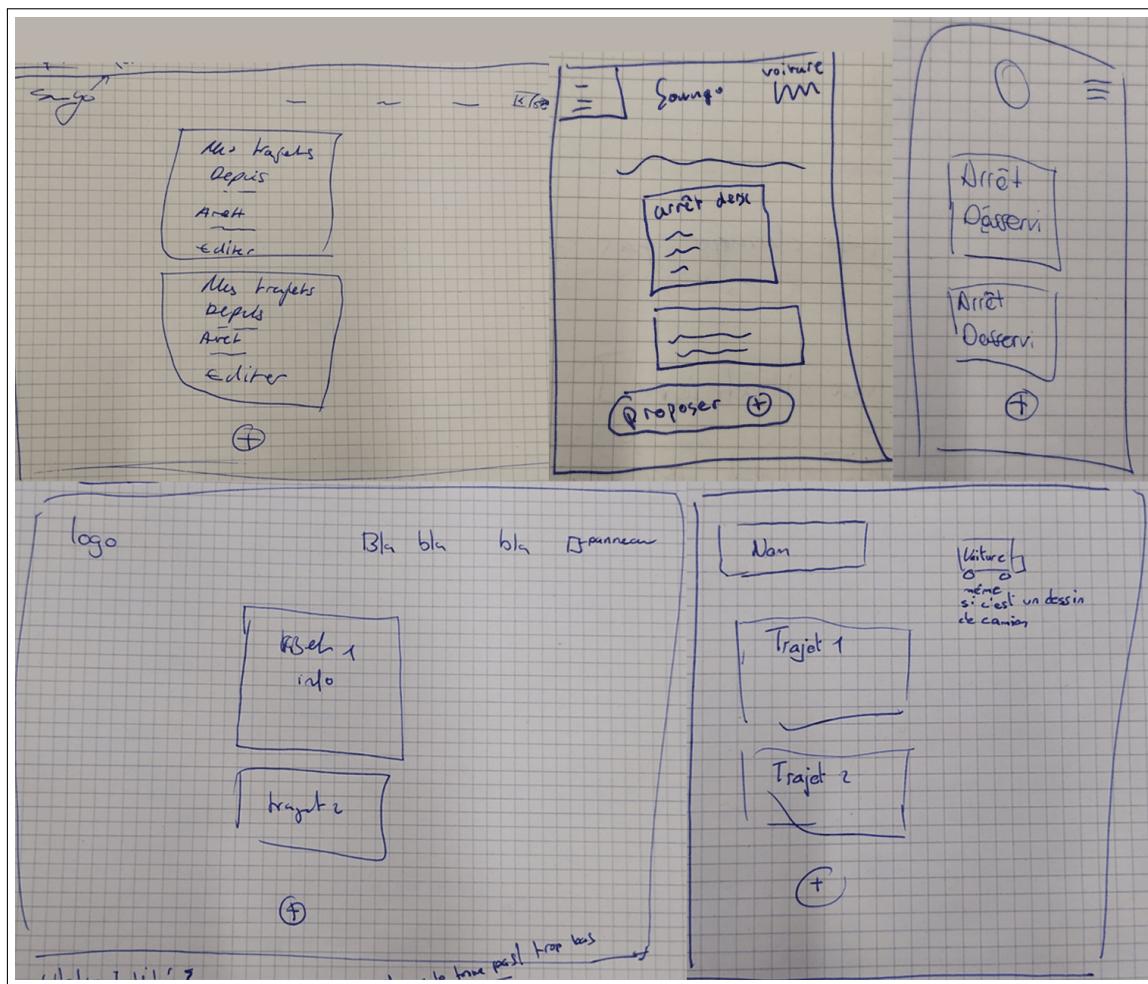


FIGURE 20.1 – Croquis des testeurs

## 20.2 Test d'utilisabilité

Un test d'utilisabilité permet de vérifier le bon fonctionnement des fonctionnalités, de voir si l'utilisateur comprend les démarches et parvient à les effectuer sans encombre. C'est une étape indispensable.

Nous avons demandé aux utilisateurs d'effectuer cinq tâches au cœur de notre application afin de mesurer l'aisance avec laquelle ils les accomplissent.

- Se connecter en tant que passager avec comme nom de profil "Patrick"
- Supprimer un trajet
- Ajouter un trajet en tant que conducteur, destination, départ et direction imposé
- Devenir passager en étant conducteur
- Se déconnecter

### Se connecter en tant que passager avec comme nom de profil "Patrick"

Test passé haut la main pour tous les testeurs. Une fonctionnalité permettant d'appuyer sur la touche "enter" pour valider le login serait appréciée.

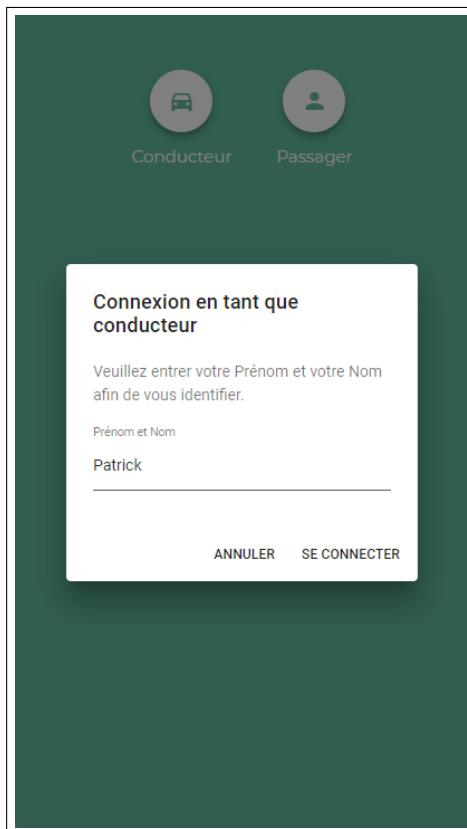


FIGURE 20.2 – Connexion comme Conducteur avec le nom "patrick"

### Supprimer un trajet

Tout le monde a passé le test avec succès, l'icône de la poubelle parle à tout le monde.

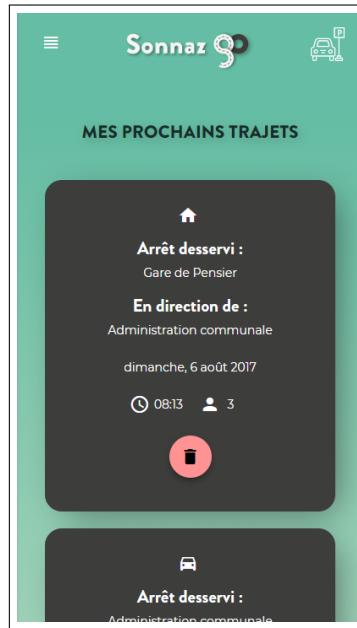


FIGURE 20.3 – Bouton de suppression en rouge quand on le "hover"

#### Ajouter un trajet en tant que conducteur, destination, départ et direction imposé

Pas de problème pour trouver le bouton pour ajouter un nouveau trajet, ni pour choisir une direction et remplir le formulaire.

Pour la version mobile, certains utilisateurs n'ont pas compris qu'il fallait défiler au bas de la page pour voir le bouton pour passer à la page suivante.

Dans les deux cas, au moment du choix de la destination interne, certaines personnes ont essayé de cliquer sur les gares. Nous avons résolu ce problème en changeant la couleur des marqueurs en rouge afin qu'ils attirent le regard.

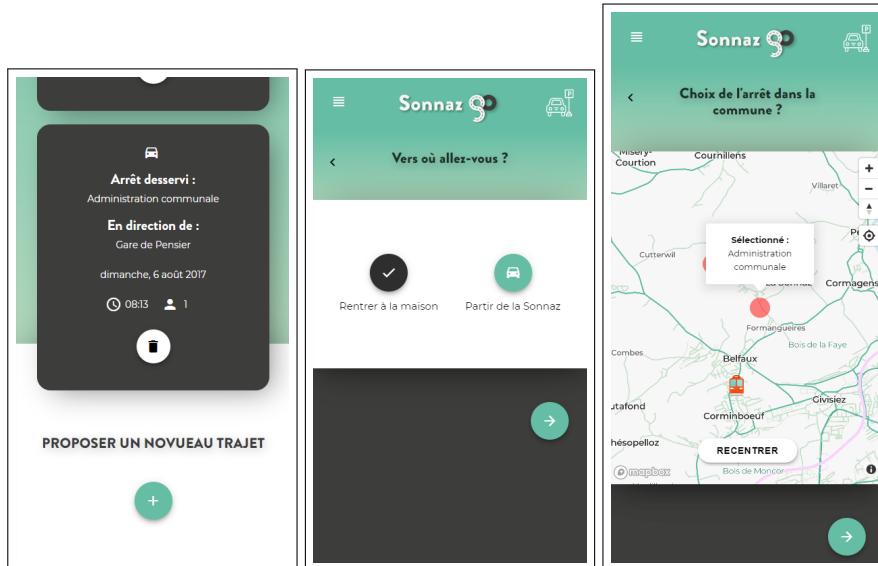


FIGURE 20.4 – Nouveau trajet, choix de la direction, choix de destination interne

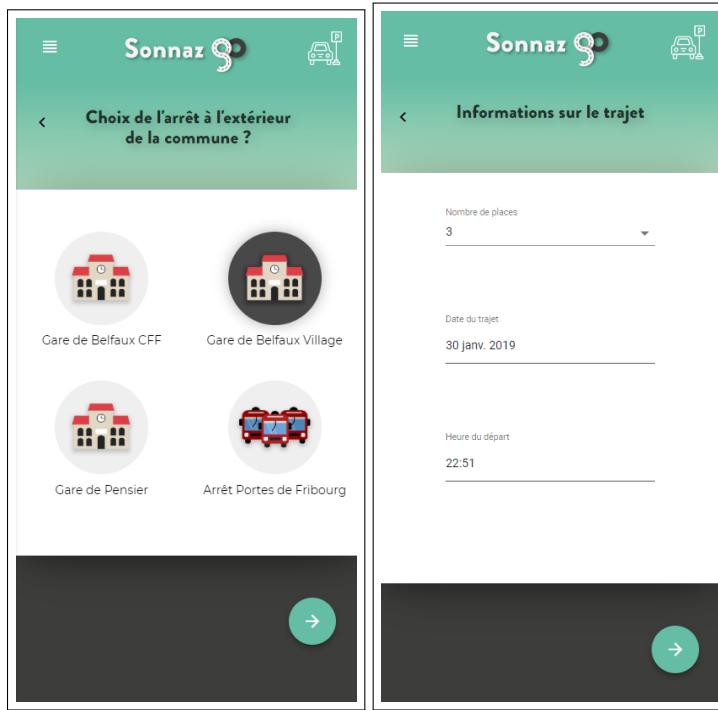


FIGURE 20.5 – Choix de destination externe et Formulaire d’informations sur le trajet

#### Devenir passager en étant conducteur

Une personne a dû chercher quelque secondes pour le bouton car il ne se mettait pas en surbrillance comme le reste des items. Nous avons corrigé cela comme vous pouvez le voir ci-dessous.

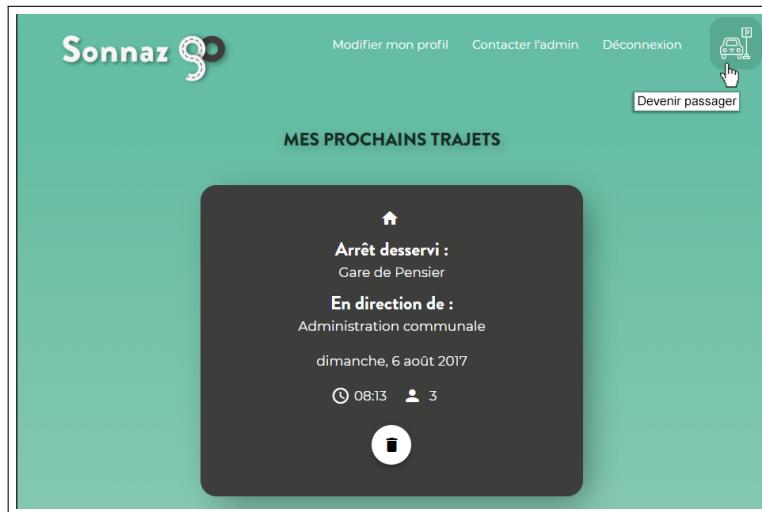


FIGURE 20.6 – Devenir passager avec le bouton en haut à droite

**Se déconnecter**

Tout le monde y es parvenu rapidement.

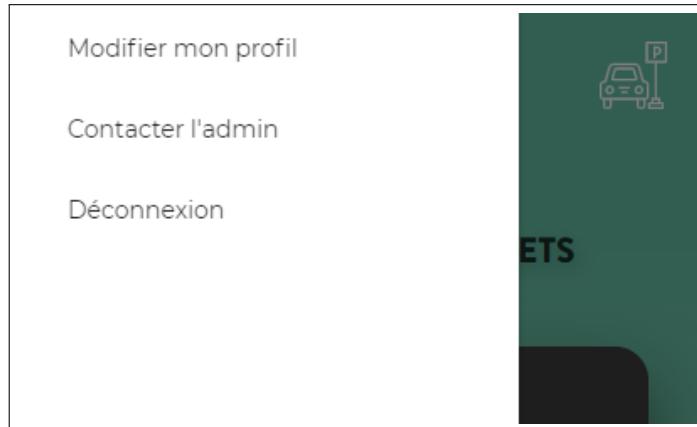


FIGURE 20.7 – Bouton déconnexion dans le hamburger menu

# Conclusions



<b>21</b>	<b>Conclusions du projet . . . . .</b>	<b>93</b>
21.1	Validation des objectifs	
21.2	Problèmes rencontrées et solutions apportées	
21.3	Perspectives Futures	
21.4	Remerciements	
21.5	Conclusion du projet	
21.6	Conclusion personnelle	
<b>22</b>	<b>Conclusions du document . . . . .</b>	<b>97</b>
22.1	Licences	
22.2	Déclaration d'honneur	
	<b>Glossaire . . . . .</b>	<b>103</b>
	<b>Acronymes . . . . .</b>	<b>104</b>
	<b>Table de figures . . . . .</b>	<b>107</b>
22.3	Planning	
22.4	Tâches heures par heures	



## 21. Conclusions du projet

### 21.1 Validation des objectifs

Nos objectifs principaux étaient séparées en trois catégories :

- 1. Analyse technologique
- 2. Analyse besoins User experience design (UX) et marketing
- 3. Prototype front-end

#### 1. Analyse technologique

Tous les objectifs spécifiées dans cette partie ont été remplis et les livrables livrés.

#### 2. Analyse besoins UX et marketing

Tous les objectifs spécifiées dans cette partie ont été remplis et les livrables livrés.

#### 3. Prototype front-end

Nous avons dû remanier un peu les objectifs. Nous avons choisi, avec nos responsables de projet, de sortir le système de badge du prototype 1 et donc de ne pas le réaliser pour notre projet de semestre 5.

La cause de ce changement a été une mauvaise estimation de notre part. Nous avons pris plus de temps que nous l'avions estimé pour se familiariser avec React et l'emploi soudain et imprévu de Redux dans notre projet nous a pris beaucoup de temps.

A part ça, tous les objectifs spécifiées dans cette partie ont été remplis et les livrables livrés.

### 21.2 Problèmes rencontrés et solutions apportées

#### 21.2.1 Redux

Nous avions comme "contrainte" de s'intéresser uniquement à l'aspect front-end de l'application. Cela s'est révélé être un problème lorsque nous avons dû implémenter l'aspect logique de notre application web. Nous avons été contraint de se tourner vers Redux afin d'implémenter cette logique, chose que nous n'avions pas prévue initialement.

Nous avons implémenté Redux avec succès et avons compris son fonctionnement. C'était très intéressant de le découvrir car c'est un outil très puissant.

### 21.2.2 Restructuration du projet

Nous nous sommes rendu compte, avant de vouloir implémenter Redux, que la structure de fichiers de notre application n'était pas optimale. En effet, les composants n'étaient pas cohérents au niveau de leur noms, ils n'avaient pas de vrai structure, le CSS était dans un seul fichier.

La cause de cela est que nous écrivions les composants au fur et à mesure de notre apprentissage de React et que nous ne prenions pas le temps de les structurer correctement. Nous travaillons également à deux sur le projet et avions des méthodes de nommage différentes.

Nous avons entièrement restructuré notre projet, réfléchissant à sa structure au préalable et en recommençant tout depuis zéro. Cela a nécessité un investissement en temps non négligeable mais cela était indispensable pour mener à bien notre projet.

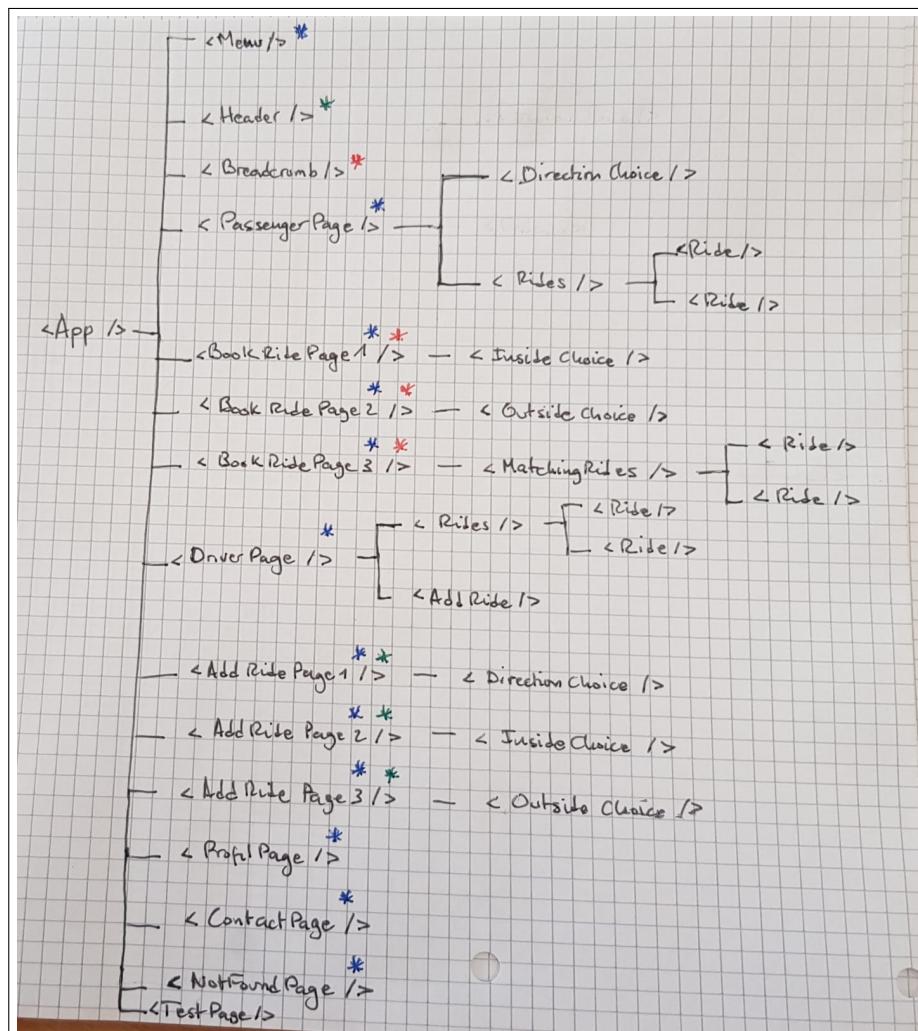


FIGURE 21.1 – Bouton déconnexion dans le hamburger menu

### **21.2.3 Cohérence des labels**

En présentant notre avancement lors des séances de projet ainsi qu'en donnant à tester notre application, nous nous sommes rendu compte que certains labels de notre application n'étaient pas assez clair. Nous avons fait au mieux pour les rendre cohérent à travers toute l'application et pour que leur signification ne soit pas ambiguë ou incomprise.

### **21.2.4 Organisation**

Nous avons pris du retard (de un peu plus d'une semaine) par rapport à notre planning à cause des périodes d'examen. Nous avons discuté de la situation avec nos responsables de projet et il a été décidé de rédiger une liste de tâches individuelles associées à un nombre d'heure nécessaire pour les réaliser. Elle nous offre une bien meilleure granularité des choses à réaliser

Nous avons finalement rattrapé notre retard, la liste de tâches se trouve en annexe.

### **21.2.5 Petits problèmes UI persistants**

- Marker qui saute de place dans la carte Mapbox

## **21.3 Perspectives Futures**

Nous avons définis beaucoup d'objectifs pour la suite dans nos objectifs secondaires (prototype 2). Il faut bien sur continuer de développer tout le frontend, il manque beaucoup d'interfaces afin de remplir complètement le cahier des charge des mandants.

Il faudra ensuite se pencher sur tout l'aspect backend, stocker les trajets dans une base de données au lieu d'utiliser le Store Redux qui est limité.

## **21.4 Remerciements**

Nous tenons à remercier les personnes qui nous ont encadré, renseigné et aidé durant ce projet :

Monsieur Joye Philippe et Madame Ingram Sandy, professeurs à la HEIA-FR, pour leur suivi et leur supervision en tant que responsables de projet durant l'entier du semestre.

Merci pour leur précieuses remarques et informations qui nous ont permis d'améliorer notre travail et d'en être fier.

## **21.5 Conclusion du projet**

Comme indiqué ci-dessus, les objectifs définis dans le cahier des charges ont été remplis. Le projet est, selon nous, une réussite. Nous avons créé un prototype utilisable que les mandants pourront tester afin de décider si oui ou non, il faut continuer le développement.

Nous avons choisi une approche "design first" lors du développement. Créant en premier tout l'aspect graphique de l'application web avant d'y implémenter la logique. C'est un choix qui a été pris afin que les mandants puissent rapidement avoir du concret à juger.

Nous avions, pendant une longue partie du développement, deux projets distincts. Un avec uniquement la logique mais avec aucun style et l'autre avec tout le côté graphique mais avec aucun fonctionnement "dynamique". Nous avons fini par intégrer la logique dans la partie graphique.

Nous avons, selon nous, atteint un des buts du projet étant, entre autre, d'utiliser React pas seulement pour générer des composants mais aussi d'utiliser ses fonctionnalités clés comme l'utilisation des props et des states. Nous sommes maintenant à l'aise avec cette technologie.

L'application web Sonnaz Go est robuste, élégante, fonctionnelle, et utilisable. Nous sommes fier de son rendu final.

## **21.6 Conclusion personnelle**

Ce projet a été une vraie aubaine pour nous de nous informer et de nous former sur différentes technologies web modernes. Nous avons pu mettre à profit nos compétences nouvellement apprises dans un projet concret et intéressant.

Ce fut assez challengeant d'apprendre autant de nouvelles technologies en si peu de temps. Nous avons pris un temps considérable afin de nous former à React et à Redux et cela a entraîné un retard sur le planning et une obligation de modifier les objectifs du projet.

Nous avons eu des grosses périodes de stress et de doute vers la fin du semestre. Cela contrastait beaucoup avec notre grande confiance du début. Nous avons appris comme leçon, pour nos futurs projets, de mieux accepter les remarques lors des séances, de ne pas délaisser le projet en périodes d'exams et de se tenir très fermement au planning.

Nous ressortons grandit de ce premier projet de semestre. Nous, Nestor et Patrick, sommes fier de vous présenter le fruit de notre travail.



## 22. Conclusions du document

### 22.1 Licences

Package	Version	Licence
@date-io/date-fns	^1.0.1	MIT © Sasha Koss
@material-ui/core	^3.8.3	MIT
@material-ui/icons	^3.0.2	MIT
date-fns	^2.0.0-alpha.25	MIT © Sasha Koss
firebase-tools	^6.2.2	MIT
g	^1.0.1	MIT
mapbox-gl	^0.52.0	3-Clause BSD license
material-design-icons	^3.0.1	Apache-2.0
material-ui-pickers	^2.1.2	MIT
moment	^2.24.0	MIT
node-sass	^4.11.0	MIT
normalize.css	^8.0.1	MIT
prop-types	^15.6.2	MIT
react	^16.7.0	MIT
react-addons-shallow-compare	^15.6.2	MIT
react-dates	^12.7.0	MIT
react-dom	^16.7.0	MIT
react-materialize	^2.6.0	MIT
react-redux	^5.0.5	MIT
react-reveal	^1.2.2	MIT
react-router-dom	^4.3.1	MIT
react-router-redux	^4.0.8	MIT
react-scripts	^2.1.3	MIT
redux	^3.7.2	MIT
sw-precache-cra	^1.0.0-alpha.2	MIT
uuid	^3.3.2	MIT

Comme nous pouvons observer dans le tableau ci-dessus nous avons utilisé des paquets ayant un total de 3 licences :

- MIT(<https://kossnocorp.mit-license.org/>)
- 3-Clause BSD license(<https://opensource.org/licenses/BSD-3-Clause>)
- Apache-2.0(<https://www.apache.org/licenses/LICENSE-2.0.html>)

Ce machin c'est juste pour que tu voies comment ça marche les acronymes et le glossaire :

## **22.2 Déclaration d'honneur**

Je, soussigné, Nestor Peña López, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

Je, soussigné, Patrick Audriaz, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

Fribourg, 31 janvier 2019

---

Nestor Peña

---

Patrick Audriaz



## Bibliographie

- [Ahm18] Kamran AHMED. *Modern Frontend Developer in 2018*. 2018. URL : <https://medium.com/tech-tajawal/modern-frontend-developer-in-2018-4c2072fa2b9c> (visité le 14/10/2018).
- [Ano18a] ANONYME. *Covoiturage*. 2018. URL : <https://www.ate.ch/themes/voiture/autopartage/covoiturage/> (visité le 10/11/2018) .
- [Ano18b] ANONYME. *Documentation Material Design*. 2018. URL : <https://material.io/design/introduction/#principles> (visité le 11/11/2018).
- [Ano18c] ANONYME. *Documentation Materialize*. 2018. URL : <https://materializecss.com/about.html> (visité le 12/11/2018).
- [Ano18d] ANONYME. *Documentation React*. 2018. URL : <https://reactjs.org/docs/introducing-jsx.html> (visité le 01/12/2018).
- [Ano18e] ANONYME. *Documentation React*. 2018. URL : <https://reactjs.org/docs/state-and-lifecycle.html> (visité le 01/12/2018).
- [Ano18f] ANONYME. *Documentation React*. 2018. URL : <https://reactjs.org/docs/components-and-props.html> (visité le 01/12/2018).
- [Ber18] Sébastien BERGIA. *Aide au choix d'un framework JavaScript*. 2018. URL : <http://blog.ippon.fr/2018/03/12/aide-au-choix-dun-framework-javascript> (visité le 19/10/2018).
- [Boc17] Vincent BOCQUET. *Votre premier composant React - Partie 1*. 2017. URL : <https://medium.com/@vincent.bocquet/votre-premier-composant-react-partie-1-e3c5047402f4> (visité le 03/12/2018) .
- [Bot16] Marco BOTTO. *The Hitchhiker's guide to the modern front end development workflow*. 2016. URL : <https://marcobotto.com/blog/the-hitchhikers-guide-to-the-modern-front-end-development-workflow> (visité le 15/10/2018).
- [Buc17] Owen BUCKLEY. *A Modern Day Front-End Development Stack*. 2017. URL : <https://www.linux.com/blog/learn/2017/7/modern-day-front-end-development-stack> (visité le 12/10/2018).
- [Cas18] Laurène CASTOR. *Utilisez le framework "Materialize CSS"*. 2018. URL : <https://openclassrooms.com/fr/courses/3936801-composez-des-interfaces-utilisateurs-en-material-design/4392371-utilisez-le-framework-materialize-css> (visité le 03/11/2018).
- [Cho18] Hitesh CHOUDARY. *What is JavaScript and where can we use it*. Youtube. 2018. URL : <https://www.youtube.com/watch?v=d1frWbYk1v0>

- [Cla18] Open CLASSROOM. *Des applications ultra-rapides avec Node.js*. 2018. URL : <https://openclassrooms.com/fr/courses/1056721-des-applications-ultra-rapides-avec-node-js/1056866-node-js-mais-a-quoi-ca-sert> (visité le 12/10/2018).
- [Com18] L'équipe COMMUNIDÉE. *4 étapes pour créer l'identité visuelle de votre entreprise*. 2018. URL : <http://www.communidee.com/lidentite-visuelle/> (visité le 13/11/2018).
- [F17] Mégane F. *5 étapes pour mettre en place son identité visuelle*. 2017. URL : <https://toile-de-marque.com/strategie/5-etapes-identite-visuelle/> (visité le 13/11/2018)
- [Fri18] Etat de Fribourg. *Covoiturage*. 2018. URL : <https://www.fr.ch/smo/mobilite-et-transport/en-voiture/covoiturage> (visité le 08/11/2018)
- [Gee18] GEEKYANTS. *Progressive Web Apps—The Next Step in Web App Development*. 2018. URL : <https://hackernoon.com/progressive-web-apps-the-next-step-in-web-app-development-372235bf9a99> (visité le 01/11/2018)
- [Kap18] Shruti KAPOOR. *Progressive Web Apps 101 : the What, Why and How*. 2018. URL : <https://medium.freecodecamp.org/progressive-web-apps-101-the-what-why-and-how-4aa5e9065ac2> (visité le 02/11/2018)
- [Kum18] Arun KUMAR. *A quick introduction to Material Design using Materialize*. 2018. URL : <https://medium.freecodecamp.org/an-quick-introduction-to-material-design-using-materialize-8a9b223c64f1> (visité le 08/11/2018)
- [Lim18] Henry LIM. *Zero to 15 — Building a Nothing PWA in 15 minutes*. 2018. URL : <https://dev.to/henrylim96/zero-to-15--building-a-nothing-pwa-in-15-minutes-258j> (visité le 30/10/2018)
- [Loy17] Fabrice LOYOLA. *Les étapes de la création d'une charte graphique*. 2017. URL : <http://artiste2d3d.com/etapes-de-creation-dune-charte-graphique/> (visité le 13/11/2018)
- [M17] Manjunath M. *Stateful vs. Stateless Functional Components in React*. 2017. URL : <https://code.tutsplus.com/tutorials/stateful-vs-stateless-functional-components-in-react--cms-29541> (visité le 03/12/2018)
- [Met18] Vladimir METNEW. *Best UI Frameworks for your new React.js App*. 2018. URL : <https://hackernoon.com/the-coolest-react-ui-frameworks-for-your-new-react-app-ad699fffd651> (visité le 07/11/2018)
- [Mos18] Programming with Mosh. *What is JavaScript*. Youtube. 2018. URL : <https://www.youtube.com/watch?v=upDLs1sn7g4>
- [Neu17] Jens NEUHAUS. *Angular vs. React vs. Vue : A 2017 comparison*. 2017. URL : <http://blog.ippon.fr/2018/03/12/aide-au-choix-dun-framework-javascript> (visité le 20/10/2018)
- [Nic14] Raymond Yerly NICOLAS SCHROETER Sladjan Milanovic. *SwissMobil : service offert à l'usager*. 2014.
- [One18] ONEJOHI. *Introducing Geolocation in your PWAs*. 2018. URL : <https://medium.com/@onejohi/introducing-geolocation-in-your-pwas-65713faba51a> (visité le 01/11/2018)
- [Ove18] Stack OVERFLOW. *Developer Survey Results 2018*. 2018. URL : <https://insights.stackoverflow.com/survey/2018> (visité le 01/10/2018)

- [Por18] Christophe PORTENEUVE. *Réalisez une application web avec React.js*. 2018. URL : <https://openclassrooms.com/fr/courses/4664381-realisez-une-application-web-avec-react-js/4664388-decouvrez-lutilite-et-les-concepts-cles-de-react> (visité le 03/12/2018)
- [Sch15] Code SCHOOL. *What is JavaScript*. Youtube. 2015. URL : <https://www.youtube.com/watch?v=nITsSTwBvSU> (cf. page 99).
- [Ser17] Gaël SERVAUD. *Comment lier React et le DOM*? 2017. URL : <https://www.apprendre-react.fr/tutorial/debutant/render-et-virtual-dom/> (visité le 03/12/2018)
- [Tec18a] Maruti TECH. *Why Progressive Web App is the future of web development?* 2018. URL : <https://www.marutitech.com/progressive-web-app/> (visité le 20/10/2018)
- [Tec18b] TECHNO-SCIENCE.NET. *Framework définition*. 2018. URL : <https://www.techno-science.net/definition/1471.html> (visité le 18/10/2018).
- [wag15] Le WAGON. *Javascript pour les débutants*. Youtube. 2015. URL : <https://www.youtube.com/watch?v=cQZOfeKrWDs>.
- [Web17] Bitfumes WEBNOLOGIES. *What is a Framework and Why we use Frameworks?* 2017. URL : <https://www.youtube.com/watch?v=1a5VKUc0AUc> (visité le 19/10/2018)
- [Wel16] Anthony WELC. *5 raison de choisir Javascript*. Youtube. 2016. URL : <https://www.youtube.com/watch?v=-0duzjYZ-zM>
- [wha18] WHATWEBCANDO.TODAY. *What Web Can Do Today*. 2018. URL : <https://whatwebcando.today/> (visité le 02/11/2018).
- [Wik18a] WIKIPEDIA. *CSS-framework*. 2018. URL : <https://en.wikipedia.org/wiki/CSS-framework> (visité le 02/11/2018).
- [Wik18b] WIKIPEDIA. *Flat Design*. 2018. URL : [https://fr.wikipedia.org/wiki/Flat\\_design](https://fr.wikipedia.org/wiki/Flat_design) (visité le 12/11/2018).
- [Wik18c] WIKIPEDIA. *HTML5*. 2018. URL : <https://fr.wikipedia.org/wiki/HTML5> (visité le 18/10/2018).
- [Wik18d] WIKIPEDIA. *JavaScript*. 2018. URL : <https://fr.wikipedia.org/wiki/JavaScript> (visité le 01/10/2018).
- [Wik18e] WIKIPEDIA. *Material Design*. 2018. URL : [https://fr.wikipedia.org/wiki/Material\\_design](https://fr.wikipedia.org/wiki/Material_design) (visité le 13/11/2018).
- [Zak13] Nicholas C. ZAKAS. *Developer Survey Results 2018*. 2013. URL : <https://humanwhocodes.com/blog/2013/10/07/node-js-and-the-new-web-front-end> (visité le 10/10/2018).



## Glossaire

**backend** partie "invisible" (logique) d'une application. 16

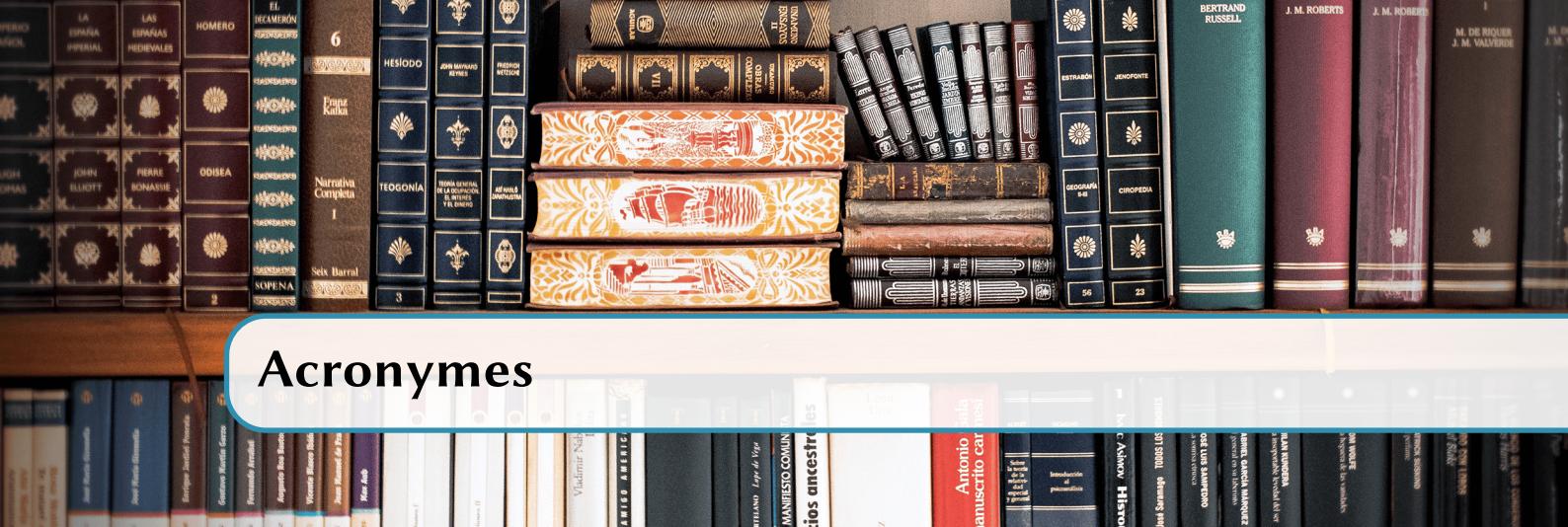
**framework** un ensemble de fonctions et conventions que l'on peut réutiliser et standardiser pour faciliter la création de tout ou une partie d'un système logiciel. 15  
**frontend** partie d'une application qu'un utilisateur peut voir et avec laquelle il peut interagir. 16

**isomorphique** code sur toutes les plateformes. 26

**librairie** collection de sources à intégrer. 15

**skeuomorphisme** style de design d'interface graphique. 40

**w3c** organisme de standardisation. 61



## Acronymes

**API** Application programming interface. 25

**BPMN** Business process model and notation. 6, 51, 52

**DOM** Document Object Model. 61

**HTML** Hypertext Markup Language. 23

**IDE** Integrated development environment. 20

**PHP** Hypertext Preprocessor. 18

**PWA** Progressive Web App. 15

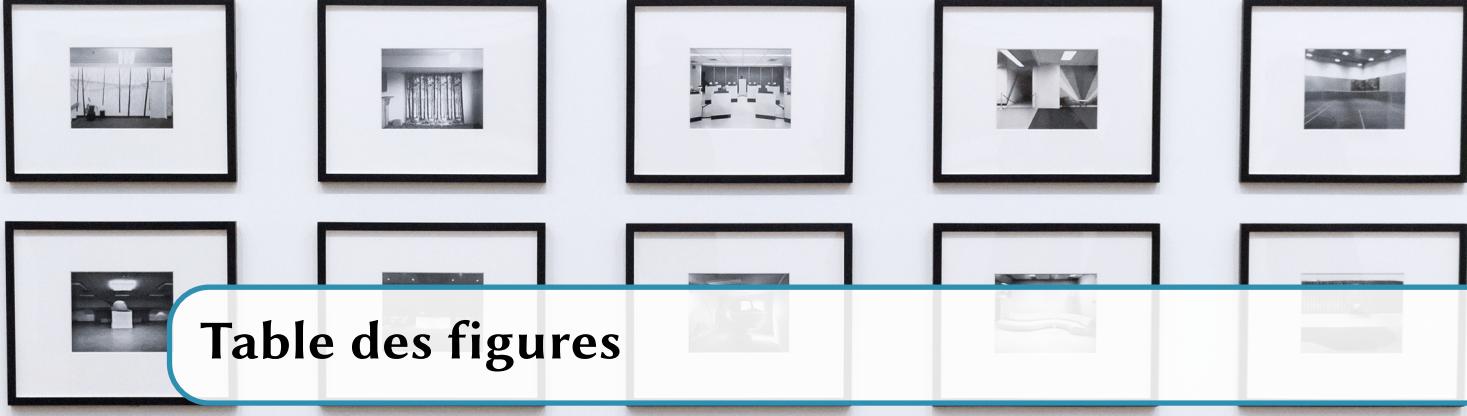
**UI** Interface Utilisateur. 62

**UML** Unified Modeling Language. 12

**URL** Uniform Resource Locator. 22

**UX** User experience design. 93

**VR** Réalité virtuelle. 17



# Table des figures

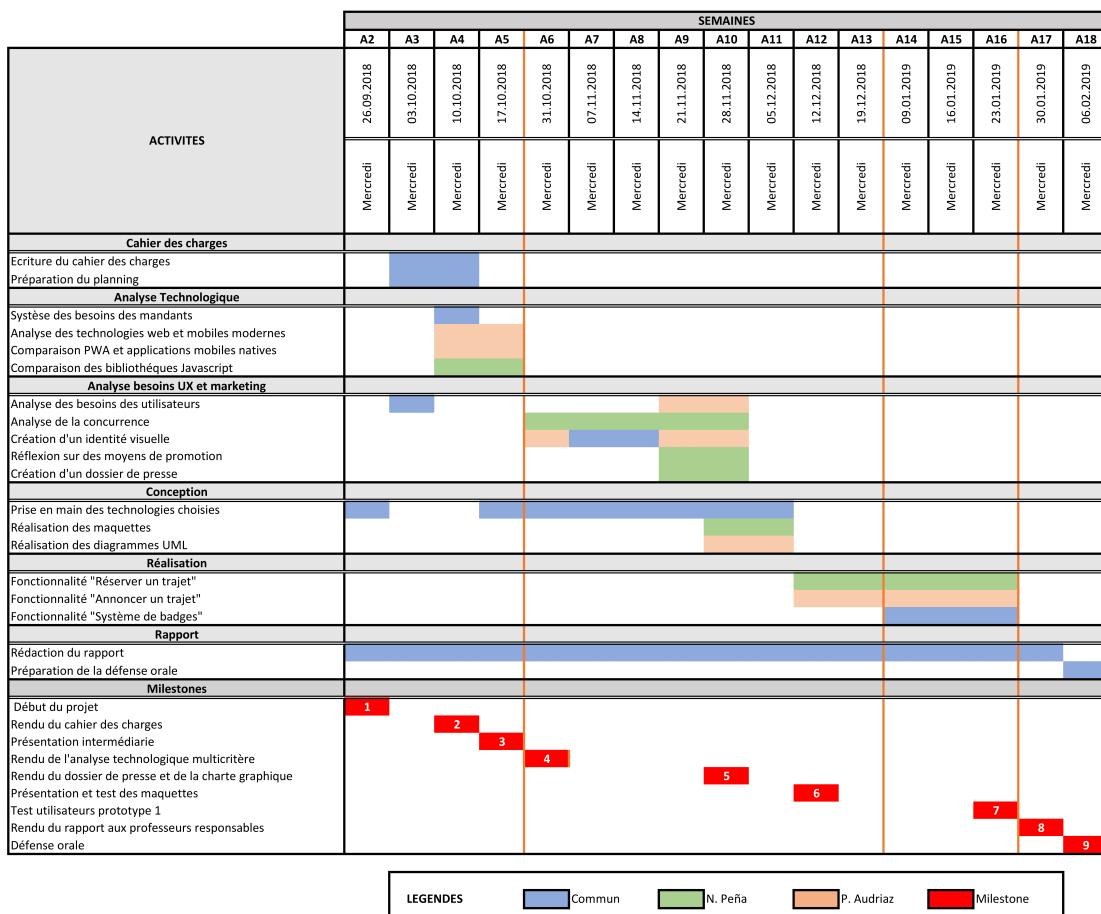
5.1 Popularité des langages sur StackOverflow . . . . .	16
5.2 Architecture sans Node.JS où tout le travail frontend se faisait dans le navigateur . . . . .	17
5.3 Communication Client / serveur "traditionnelle" avec PHP sur le server et JavaScript chez le client . . . . .	18
5.4 Architecture type avec Node.JS . . . . .	18
5.5 Serveur utilisant Node.JS . . . . .	19
5.6 Différences de structure et de balises entre l'ancien et le nouveau HTML . . . . .	20
5.7 Exemple d'animation CSS faisant "pulser" un carrés . . . . .	20
5.8 État de l'écosystème JavaScript . . . . .	21
6.1 Les 4 piliers d'une PWA . . . . .	23
6.2 Comparaison de l'engagement de l'utilisateur entre le web et les applications . . . . .	24
6.3 Exemple de fichier manifest.json . . . . .	24
6.4 Schéma de fonctionnement d'un service worker . . . . .	25
7.1 Nombre de téléchargements effectués entre 2015 et 2018 dans NPM . . . . .	28
8.1 WAAD dans sa globalité . . . . .	33
9.1 Applications de covoiturage en Suisse . . . . .	35
9.2 Réservation d'un trajet depuis la Sonnaz avec Frimobility . . . . .	38
9.3 Réservation d'un trajet depuis la Sonnaz avec Fribourg-Covoiturage . . . . .	38
10.1 Exemple de skeuomorphisme dans iOS6 comparé à du flat design . . . . .	40
10.2 Évolution du logo de Apple vers le flat design . . . . .	41
10.3 Principes du material design par Google . . . . .	41
10.4 Comparaison de flat design et material design . . . . .	42
10.5 Logo et résumé de Materialize . . . . .	42
10.6 Recherche de logo en ayant initialement comme nom "Via Sonnaz" et étant ensuite passé à "Sonnaz Go", croquis et vectorisation . . . . .	44

10.7	Premier jet de logo : rejeté . . . . .	45
10.8	Second test de logo au propre . . . . .	46
10.9	Logo final de Sonnaz Go avec les modifications . . . . .	46
10.10	Brandon Grotesque Font . . . . .	47
10.11	Identité visuelle de Sonnaz Go . . . . .	48
10.12	Communiqué de presse Sonnaz Go . . . . .	50
11.1	BPMN du processus de connexion et d'enregistrement . . . . .	52
11.2	BPMN du processus de proposition de trajet . . . . .	52
11.3	BPMN du processus de réservation d'un trajet . . . . .	53
12.1	Diagramme de cas d'utilisation . . . . .	54
13.1	Chargement, Enregistrement et Connexion . . . . .	55
13.2	Créer un compte (conducteur), Numéro de téléphone et Confirmation numéro de téléphone	
56		
13.3	Guide 1, 2 et 3 . . . . .	56
13.4	Accueil conducteur et "Hamburguer" Menu . . . . .	57
13.5	Choix arrêt externe, interne et formulaire . . . . .	58
13.6	Accueil passager, "Hamburguer" menu et Devenir conducteur . . . . .	58
13.7	Point d'embarquement et Choix du trajet . . . . .	59
14.1	Schématisation du DOM virtuel . . . . .	61
14.2	Schématisation du fonctionnement de JSX . . . . .	62
14.3	Résumé des deux types de composants React . . . . .	63
15.1	Structure du projet . . . . .	64
15.2	Exemple d'erreur du serveur de développement dans la console . . . . .	65
15.3	index.js . . . . .	66
15.4	App.js . . . . .	67
15.5	Exemple de lien sur un bouton et un label . . . . .	67
15.6	Enregistrement d'un service worker . . . . .	67
15.7	Validation que le service worker est en place et fonctionne . . . . .	68
15.8	Fichier sw-config définissant un fonctionnement "network first" pour le service worker	69
15.9	Tableau de bord Firebase . . . . .	69
16.1	Utilisation du composant Provider . . . . .	70
16.2	Props pour la création d'un nouveau trajet . . . . .	71
16.3	Utilisation de scrollTo . . . . .	71

16.4 Utilisation de momentjs . . . . .	71
16.5 Affichage conditionnelle du button . . . . .	72
16.6 Composant parent . . . . .	72
16.7 Composant fils . . . . .	72
17.1 Logo Material-UI . . . . .	73
17.2 Fichiers SCSS dans notre structure de fichier . . . . .	74
17.3 Exemple de personnalisation d'un bouton Material-UI . . . . .	75
17.4 Exemple de mise en place de la grille sur la plage d'accueil du conducteur . . . . .	76
17.5 Exemple simple de media queries . . . . .	76
18.1 Logo Redux . . . . .	77
18.2 Illustration du fonctionnement de React avec Redux . . . . .	78
18.3 Illustration du fonctionnement de Redux . . . . .	79
18.4 Import de la méthode createStore() . . . . .	80
18.5 Création du Store . . . . .	80
18.6 Création d'un reducer . . . . .	80
18.7 Utilisation de la méthode subscribe() . . . . .	81
18.8 Utilisation de la méthode dispatch() . . . . .	81
18.9 Répartition des étapes de création Redux dans notre application . . . . .	81
19.1 Logo de Mapbox . . . . .	82
19.2 Mapbox Strudio . . . . .	83
20.1 Croquis des testeurs . . . . .	87
20.2 Connexion comme Conducteur avec le nom "patrick" . . . . .	88
20.3 Bouton de suppression en rouge quand on le "hover" . . . . .	89
20.4 Nouveau trajet, choix de la direction, choix de destination interne . . . . .	89
20.5 Choix de destination externe et Formulaire d'informations sur le trajet . . . . .	90
20.6 Devenir passager avec le bouton en haut à droite . . . . .	90
20.7 Bouton déconnexion dans le hamburger menu . . . . .	91
21.1 Bouton déconnexion dans le hamburger menu . . . . .	94

# Annexes

## 22.3 Planning



## 22.4 Tâches heures par heures

Tâches	Détail / remarque	Estimation d'heures	Fini ?
Analyse besoins UX et marketing	Décompte des tâches	Heures par heures	0.5
	Améliorations et finalisation logo	Selon critiques	0.75
	Justification identité graphique		1.25
	Charte graphique	Regroupement de l'ID visu	0.5
	Analyse des besoins	Selon documents M. Ingram	1.5
	Analyse des problèmes	Selon document M. Schroeter	0.5
	Analyse de la concurrence	Technique	2
	Réflexion sur la promotion		1
	Création du dossier de presse	Avec concurrence et ID visu	2
Conception	Moyens de motivation à l'utilisation		1
	React : explication composants et tendance déclarative		2.5
	Réflexions sur l'UML		1
	Diagramme de use case		1
	Diagramme BPMN		2
	Maquette de la page d'accueil		2
	Maquette de la réservation d'un trajet		3
	Maquette de l'annonce de trajet		2
	Reste des maquettes		3
Réalisation	Formation sur React et Materialize		3
	Accueil réserver un trajet		4
	Accueil proposer un trajet		4
	Routage React (+ page 404)		1
	Hosting Firebase		1
	Service Worker		1.5
	React Materialize		1.5
	API Mapbox + intégration		4
	Données dans fichiers JSON (mock obj)		3
	Modifs selon directives Material Design		1
	Utilisation des props pour les compo		5
	Processus de proposition d'un trajet		6
	Redux		15
	Processus de réservation d'un trajet		6
	Fonctionnalité "système de badges"		4
	Rédaction du rapport		6
	Power Point présentation		5
<b>HEURES TOTALES</b>			<b>93.5</b>

**LEGENDES**

Commun	N. Peña	P. Audriaz
--------	---------	------------