**HEIA-FR**

Projet de semestre 5

Rapport

Via Sonnaz



Application de covoiturage

**⬩ Etudiants ⬩**

Nestor Peña Lopez

Patrick Audriaz

**⬩ Responsables ⬩**

Sandy Ingram

Philippe Joye

2018 – 2019

TABLE DES MATIERES

[1~ DOCUMENT 4](#_Toc528663425)

[1.1. Métadonnées 4](#_Toc528663426)

[1.2. Organisation des documents 4](#_Toc528663427)

[1.3. Historique 4](#_Toc528663428)

[2~ INTRODUCTION 5](#_Toc528663429)

[2.1. Introduction 5](#_Toc528663430)

[2.2. Contexte 5](#_Toc528663431)

[2.3. Plan 5](#_Toc528663432)

[2.4. Objectifs et activités 6](#_Toc528663433)

[2.4.1. Objectifs principaux 6](#_Toc528663434)

[2.4.2. Objectifs secondaires 7](#_Toc528663435)

[2.5. Gestion de projet 7](#_Toc528663436)

[2.5.1. Planning et milestones 7](#_Toc528663437)

[3~ ANALYSE TECHNOLOGIQUE 8](#_Toc528663438)

[3.1. Synthèse des souhaits et besoins des mandants 8](#_Toc528663439)

[3.2. Analyse des technologies web et mobiles modernes 9](#_Toc528663440)

[3.2.1. JavaScript 9](#_Toc528663441)

[3.2.2. Backend 11](#_Toc528663442)

[Node.JS 12](#_Toc528663443)

[3.2.3. Frontend 13](#_Toc528663444)

[HTML5 13](#_Toc528663445)

[CSS3 14](#_Toc528663446)

[Ecosystème Javascript 14](#_Toc528663447)

[3.3. Comparaison des Framework Javascript 15](#_Toc528663448)

[Qu’est-ce que c’est un framework ? 15](#_Toc528663449)

[Différences entre un framework et une librairie ? 15](#_Toc528663450)

[Comparaison des frameworks et une librairie 16](#_Toc528663451)

[[…] 16](#_Toc528663452)

[3.3.1. Analyse multicritère 16](#_Toc528663453)

[3.4. Comparaison des applications mobiles natives, des PWA et des sites responsive 17](#_Toc528663454)

[Site web responsive 17](#_Toc528663455)

[Application mobile native 17](#_Toc528663456)

[Progressive Web App (PWA) 18](#_Toc528663457)

[Fonctionnement des PWA 19](#_Toc528663458)

[3.4.1. Analyse multicritère 21](#_Toc528663459)

[4~ ANALYSE BESOINS UX ET MARKETING 23](#_Toc528663460)

[4.1. Synthèse des besoins des utilisateurs 23](#_Toc528663461)

[4.2. Problèmes potentiels qui peuvent être rencontrés 23](#_Toc528663462)

[5~ CONCEPTION 24](#_Toc528663463)

[6~ REALISATION 25](#_Toc528663464)

[7~ TESTS ET VALIDATION 26](#_Toc528663465)

[8~ CONCLUSION 27](#_Toc528663466)

[8.1. Synthèse 27](#_Toc528663467)

[8.2. Conclusion 27](#_Toc528663468)

[8.3. Déclaration d'honneur 27](#_Toc528663469)

[9~ RÉFÉRENCES 28](#_Toc528663470)

[10~ ANNEXES 30](#_Toc528663471)

[10.1. Installation de REACT et de ses modules 30](#_Toc528663472)

[10.2. Création d'un nouveau projet React 30](#_Toc528663473)

DOCUMENT

* 1. Métadonnées

**Auteurs** : Nestor Peña Lopez, Patrick Audriaz

**Responsables** : Sandy Ingram, Philippe Joye

**Date** **d'édition** : 3 octobre 2018

Version : 1.0

* 1. 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>

Le code source du projet est déposé sur le dépôt Git disponible à l’adresse suivante : <https://gitlab.forge.hefr.ch/patrick.audriaz/projetsem5-pena-audriaz>

* 1. Historique

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Modification | Date | Auteur |
| 1.0 | Création du rapport | 3.10.2018 | AU, PE |

# INTRODUCTION

* 1. Introduction

Nous, Nestor Peña Lopez et Patrick Audriaz, nous sommes vu attribué 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.

* 1. 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’emprunte CO2 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.

* 1. 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 commenceront une analyse en profondeur des besoins, de la concurrence et des technologies. Cette analyse pourra nous permettre d’enchainer 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.

* 1. Objectifs et activités
     1. Objectifs principaux

**1) Analyse technologique**

- Synthèse de besoins des mandants pour appuyer nos critères d’analyse

- Quel sont les technologies modernes permettant de créer une application web et mobile (isomorphique) efficiente et robuste ?

- Comparaison entre les progressives web app et les applications mobiles natives

- Comparaison des différentes bibliothèques JavaScript

**🡪 Livrable** : Analyse comparatifs multicritères détaillé

**2) Analyse des besoins UX et marketing**

- Analyse des besoins des utilisateurs pour la conception et l’interaction de l’objectif 3

- Etude de la concurrence, des potentielles alternatives et de l’état de l’art

- Création d’une identité visuelle comprenant une charte graphique, un nom et un logo en accord avec le projet

- Création d’un dossier de presse pour présenter le projet à divers parties externes

- Réflexion sur des moyens de promotion à l’échelle de la commune

**🡪 Livrables :** Maquette interactive, dossier 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 UML des fonctionnalités

- Tests intermédiaires avec de potentiels utilisateurs au moyen des maquettes

- Réalisation d’une version intermédiaire (prototype 1) livrable comprenant les fonctionnalités suivantes :

* Réserver un trajet (passager)
* Annoncer un trajet (conducteur)
* Système de "badges" (conducteurs)

**🡪 Livrables :** Prototype 1

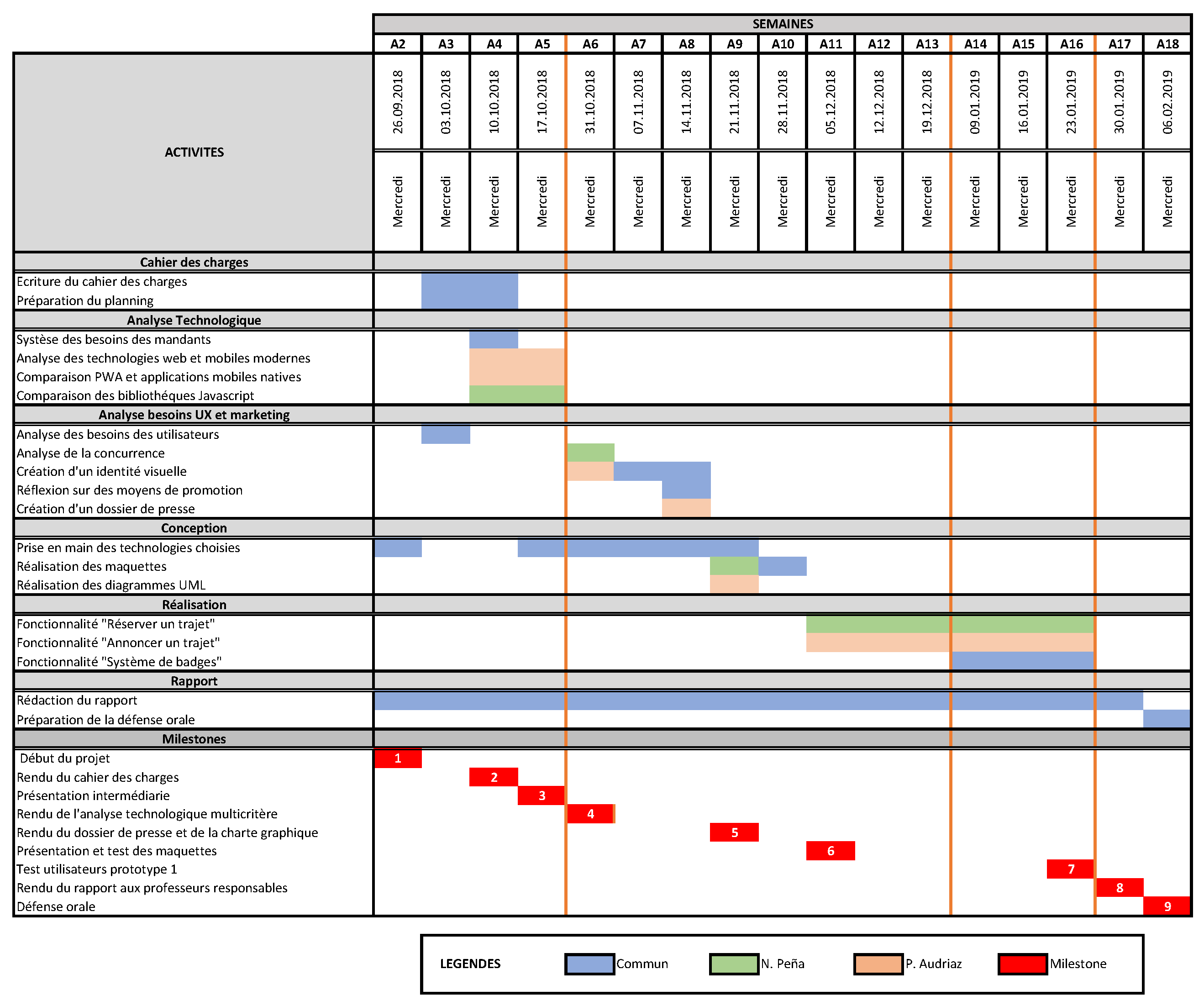
* + 1. Objectifs secondaires

1. **Étoffer le front-end pour y intégrer d’autres fonctionnalités présentes dans la maquette.**

- Réalisation d’un prototype 2 comprenant toutes les fonctionnalités de la maquette :

* Login/logout/inscription
* 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

🡪 **Livrables :** Prototype 2

* 1. Gestion de projet
     1. Planning et milestones

ANALYSE TECHNOLOGIQUE

* 1. 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 optimal.

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 librairie/framework 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 pus 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 co-voiturage sera effectuée dans la partie 4 (ANALYSE BESOINS UX ET MARKETING).

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

* Multi-plateforme (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, dossier de presse pour des acteurs externes)
* Feedback (critiques, remarques ou commentaires de la part des chauffeurs ou des passagers)
* Administration (ajout d'arrêts, gestion des conducteurs, contrôle des utilisateurs …)
  1. Analyse des technologies web et mobiles modernes

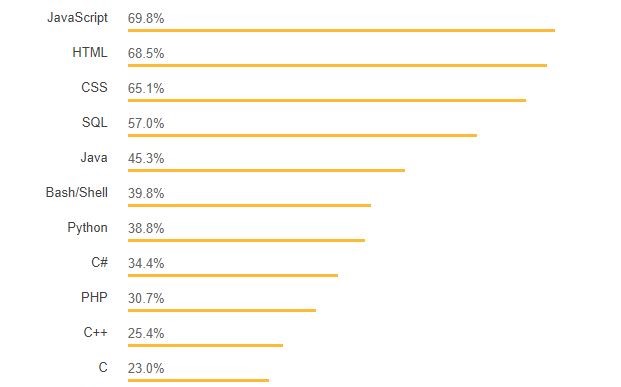
Nous allons ici nous concentrer sur l'aspect frontend du développement web mais ce n'est pas pour autant que nous n'allons pas commencer par expliquer des concepts essentiels au développement web moderne dont, le côté backend entre autres. Nous allons commencer par citer des notions de base avant de montrer une architecture types 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.

* + 1. JavaScript

*"JavaScript est un langage de programmation de scripts côté client orienté objet principalement employé dans les pages web interactives". (https://fr.wikipedia.org/wiki/JavaScript)*

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 apparaitre, disparaitre, bouger des éléments de la page, mettre à jour des éléments sans recharger la page, gérer des actions effectuées par l'utilisateur (click, clavier, souris…) pour y réagir…

JavaScript est 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.

Popularité des langages sur 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 développée 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 virtuel (VR) et même programmer les comportements des robots.

Au fil des années et grâce à une adoption massive (les navigateurs ont grandement optimisé son traitement), des outils ont été développés pour faciliter son utilisation et le rendre plus performant.

**Evolutions notables :**

**JQuery** est une bibliothèque créée pour faciliter l'écriture de scripts Javascript du côté client dans le code HTML.

**AJAX** permet de mettre à jour une des éléments d'une page ou une partie de la page sans recharger toute la page.

**ES6** est unenouvellesyntaxe pour Javascript, elle a été pensée pour créer facilement des applications web maintenables. Elle est rétrocompatible. C'est une extension de la bibliothèque standard qui ajout de nouvelles méthodes ainsi que la nouvelle syntaxe.

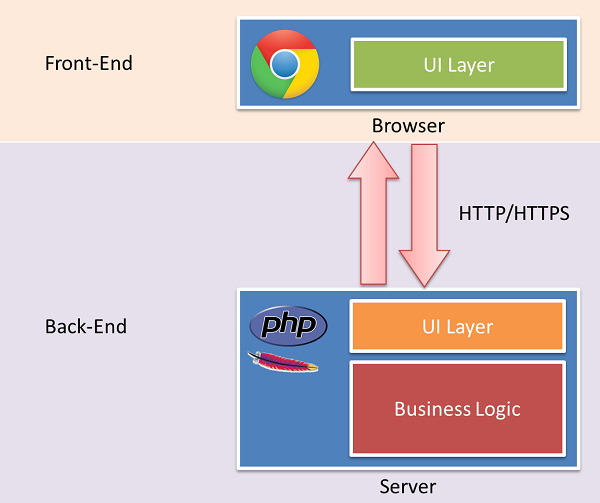
**Node.JS** permet simplement d'utiliser Javascript sur un serveur. (Plus de détails ci-dessous)

Il en existe de nombreuses autres évolutions mais nous avons fais un tour d'horizon des choses pouvant être utiles à notre projet. Nous nous étendrons sur les possibilités offertes par le Javascript "modern" plus loin dans ce rapport.

* + 1. Backend

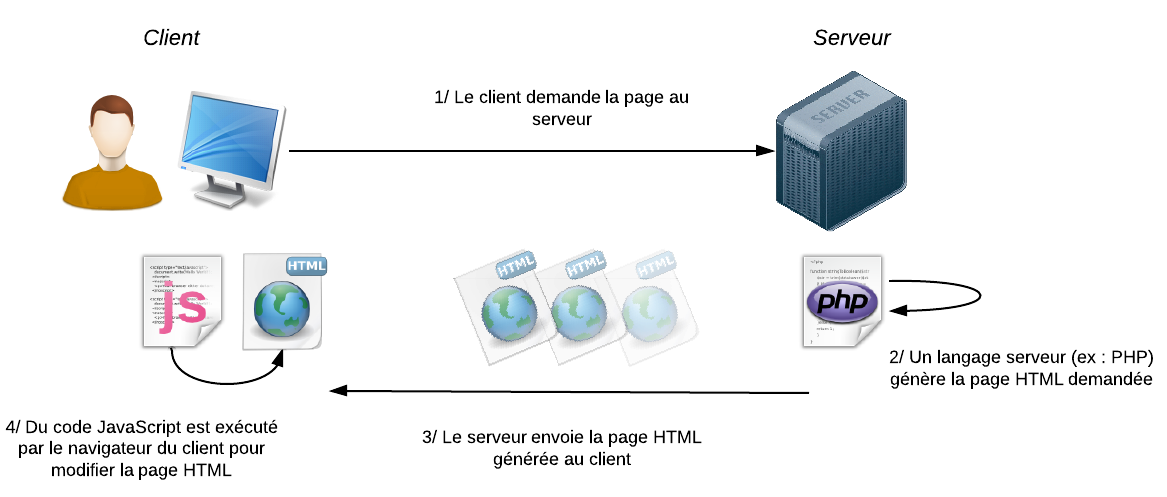
Le backend qualifie la "partie immergée" d'une application web, elle est invisible pour l'utilisateur mais représente la clé de voute de celle-ci. Une belle interface est une coquille vide sans un backend solide. Il est composé de trois piliers essentiels à un bon fonctionnement : un serveur d'hébergement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



Architecture sans Node.JS ou tout le travail frontend se faisait dans le navigateur

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



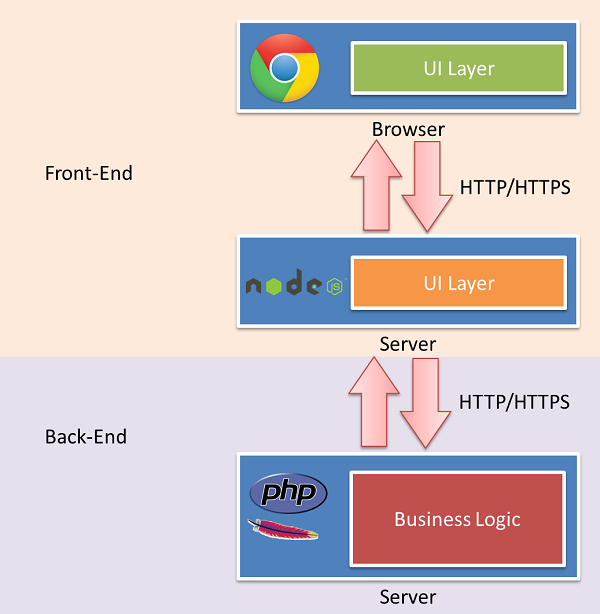
Communication Client / serveur "traditionnelle" avec PHP sur le server et Javascript chez le client

<https://user.oc-static.com/files/421001_422000/421108.png>

Node.JS

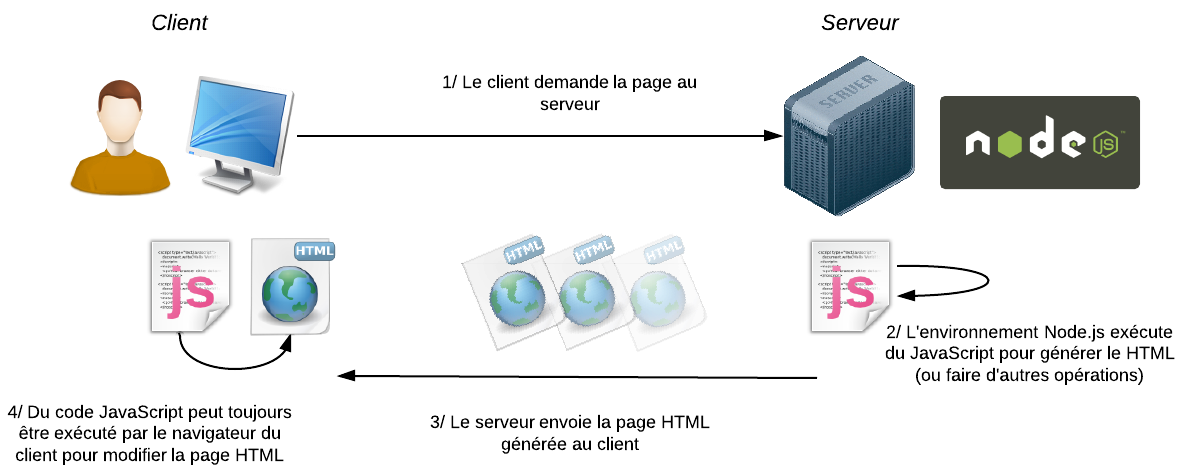
NodeJS vient résoudre ce "problème" en proposant un environnement bas niveau Javascript côté serveur. Il vient remplacer les langages dit "serveur" comme PHP. L'avantage face à ceux-ci est que, contrairement aux autres plateformes, il a une approche non-bloquante, il permet de gérer des actions et des évènements de manière asynchronee (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 évènements. Il possède également un gestionnaire de paquets (NPM) permettant des télécharger des librairies afin d'en accroitre les possibilités.

NodeJS permet de libérer la partie du backend consacré 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.



Architecture type avec Node.JS

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



Serveur utilisant Node.JS

<https://user.oc-static.com/files/421001_422000/421109.png>

* + 1. Frontend

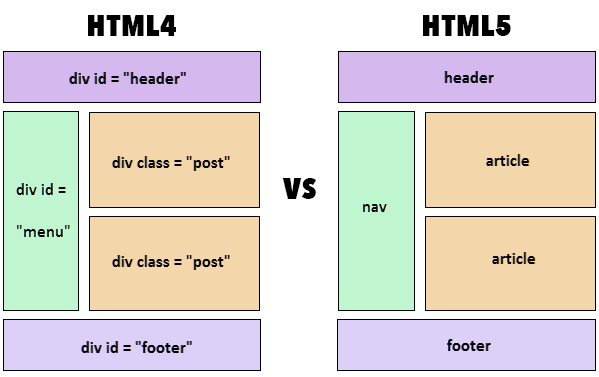
Le frontend est toute la partie visible de l'application web, tous les éléments avec lesquels l'utilisateur peu interagir, toute la partie destinée à être manipulée. Les principaux domaines touchés par le front end et donc par extension, les compétences requises pour faire une application agréable sont 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.

HTML5

(Successeur de HTML 4.0.1) HTML5 est la dernière version en date de HTML, elle offre principalement d'avantage 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, des nouveaux types de contrôle de formulaires, de nouvelles balises (header, article, nav, footer, cancas, menu…)

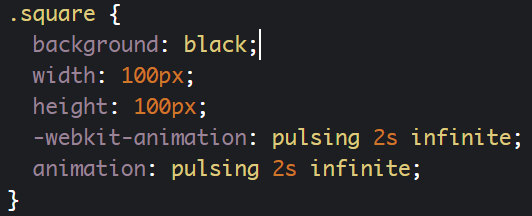


Différences de structure et de balises entre l'ancien et le nouveaux HTML

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

CSS3

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

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 : Bootstrap, Foundation, Bulma, Sematic UI…

Exemple d'animation CSS faisant "pulser" un carré

Ecosystème 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 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 fournis 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 rafraichissement automatique du navigateur quand le fichier est modifié, l'utilisations 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) transcompilé en Javascript pour améliorer la sécurité et la production de code pour celui-ci.

Finalement, il est fréquent de se reposer sur des framework Javascript, qui sont des bibliothèques pré-écrites en Javascript, pour accélérer et faciliter le développement d'applications web. Il existe différents framework frontend que nous allons décrire dans la partie 3.3 (COMPARAISON DES FRAMEWORK 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 évènements, du DOM (interface de programmation d'applications), des formulaires…



Etat de l'écosystème Javascript

<https://lh3.googleusercontent.com/0jP6hKKh1Z0IBuxEbCWNJeu0BXlqFww9eUiGynrwTepqKhjxyu5bwbcMySafteaSwmsBl_EEu_OmuDInSZQvna6SBMmZMjUz9huJOY39Z7Bj5eHtRr1xLOIL20K2xGUvCcKgeJs1>

* 1. Comparaison des Framework Javascript

Nous allons comparer différents framework 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.

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. »*

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 anglaises « 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.

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 nôtre 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 dis qu’avec un framework nous appelons notre code depuis cette structure prédéfini.

Nous voulons vous présenter cette différence en nous servant de l’analogie suivante : un framework est comme une école et une librairie est comme une maison. Dans notre école nous avons beaucoup de règles à respecter tandis que dans notre maison nous avons encore des règles mais dans un sens moins strict du terme. Avec les librairies et les frameworks il se passe exactement la même chose, 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 actuelle et nous allons les utiliser main à main l’un et l’autre.

Comparaison des frameworks et une librairie

[…]

* + 1. Analyse multicritère
  1. Comparaison des applications mobiles natives, des PWA et des sites responsive

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

Site web responsive

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

* Accessible facilement depuis un navigateur via une URL
* Développement web (HTML, CSS, JS, PHP…)
* Capacités multiplateforme
* 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

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 icone 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

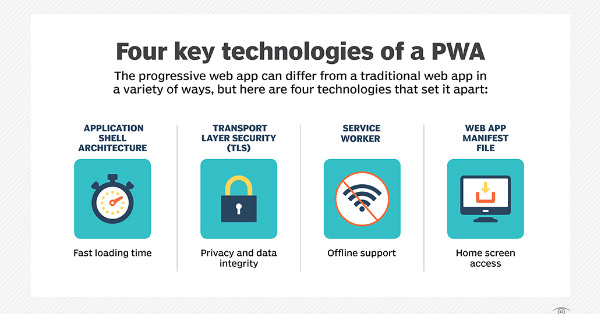
Progressive Web App (PWA)

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 accessible 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 stockés 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).

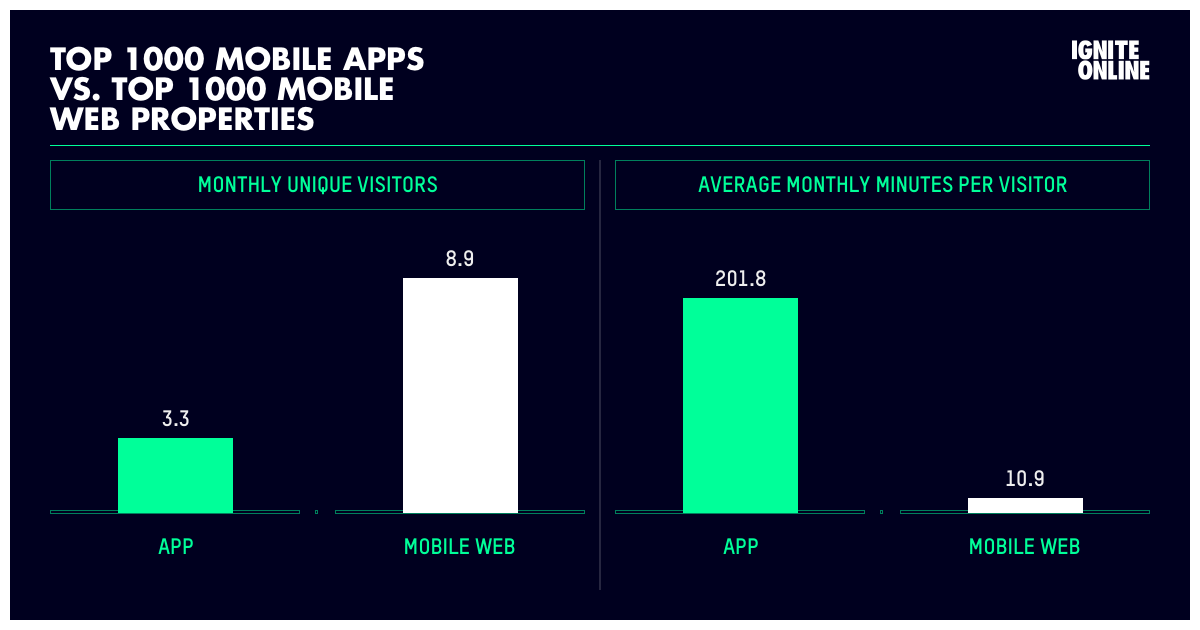
Côté développeur, les PWA représentent un gain de temps en nous permettant de faire le développement qu'une seule fois pour toutes les plateformes grâce à ses propriétés héritées des site web (accessible depuis un navigateur web et utilise les technologies de développement web comme HTML5, CSS3 et JS) au contraire des applications mobiles natives.



Les 4 piliers d'une PWA

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

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 application mobiles natives dédiées mais il y a beaucoup plus de personnes qui accèdent aux site 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 nombre d'utilisateurs des site web dans le PWA.



Comparaison de l'engagement de l'utilisateur entre le web et les applications

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

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 laceur 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 :

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

Exemple de fichier manifest.json

**Service Worker**

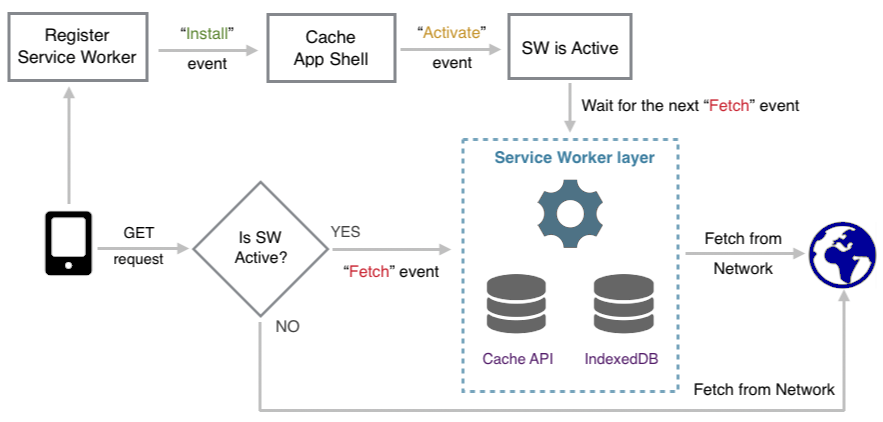
Les Service Worker sont des scripts basés sur des évènements qui permettent d'accéder à l'application en mode hors-ligne, d'envoyer des notifications ou de mettre à jour le contenu en arrière-plan. Ils fonctionnent en arrière-plan et en parallèle de l'application web. Concrètement, ce sont des scripts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

Schéma de fonctionnement d'un service worker

<http://webagility.com/posts/how-progressive-web-apps-make-the-web-great-again/_/image/f2bdbfdc-e271-4977-94ba-e9320f21eef3:a5510468926c3cb9c06e5f07a329bd577344f5cb/full/pwa-7.png>

**Notifications Push**

Une notification push est un message qui apparait sur le périphérique de l'utilisateur. Elles peuvent être envoyer 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 PWA, elles sont mises en œuvre grâce à deux 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 server 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).

* + 1. 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 notés 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). A 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.

Les critères que nous avons choisis sont les suivants :

**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.

**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'information même ne l'absence d'un réseau.

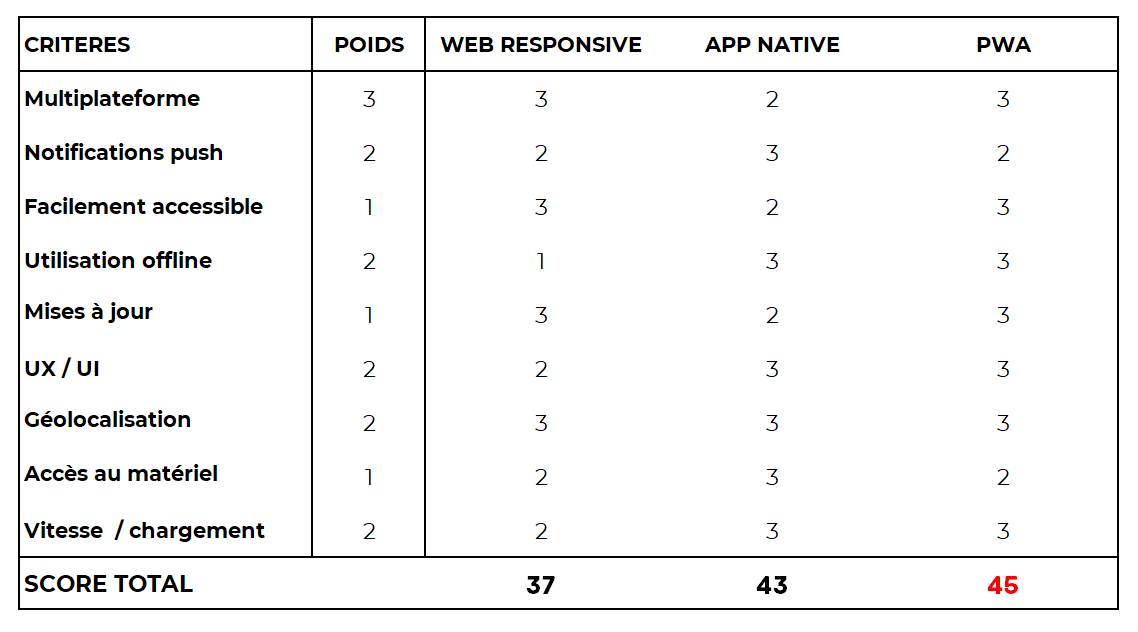
**Mises à jour :** Le système de mises à jour est transparent pour l'utilisateur et n'encombre pas l'emploi de l'application.

**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 économe en ressource.



Ce n'est pas une surprise les PWA arrivent en premier ! Notre analyse préalable du sujet nous avait déjà montré que les PWA rassemblent beaucoup de 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, comme nous le pensions donc, sont bien les PWA. En rassemblant les avantages des applications natives et des site 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.

ANALYSE BESOINS UX ET MARKETING

* 1. Synthèse des besoins des utilisateurs
     1. Analyse des besoins
     2. Problèmes potentiels qui peuvent être rencontrés
  2. Analyse concurrentielle
  3. Identité visuelle
     1. Framework CSS

La notion de framework a déjà été analysé 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 dirigé 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 framework CSS mais à la différence qu'il suit justement les principes du material design. Il fournit de nombreux composants prêt à l'emploi afin de reproduire en un temps réduit des applications web respectant le style material design.



Logo et résumé de Materialize

<http://idboard.net:10000/web/wp-content/uploads/sites/2/2018/07/55da385022302664040110.png>

* + 1. Material design vs Flat design

Principes du flat design

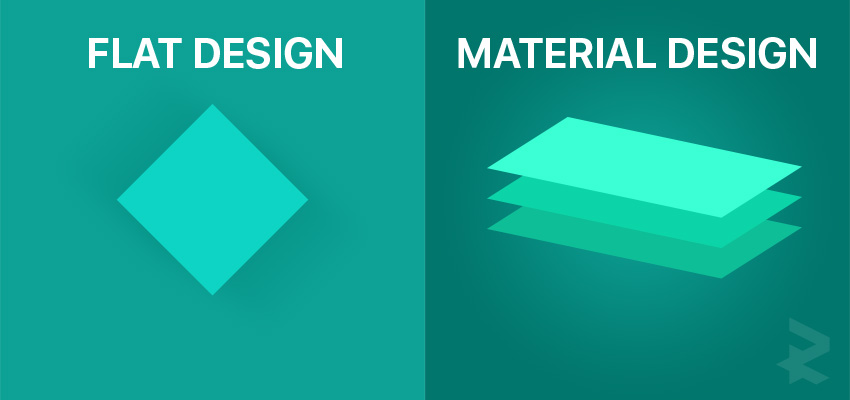
 « Less is More. »

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 (pas de profondeur, peu de textures et d'ombres). Il a donc des signifiances moins flagrantes.

Il est le total opposé du skeuomorphisme (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, ce qui est la capacité d'une design à évoquer sa fonction, à appeler à l'action. A cause de son côté (trop) minimaliste, 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.



<https://www.zetkay.com/wp-content/uploads/2018/03/FlatVsMaterial.jpg>

Principes du material design

* + 1. Elaboration et croquis initiaux

**Notre nom** : Sonnaz Go

**Nos valeurs** : le partage, le transport et l'écologie.

Il faut bien sur avant tout choisir un nom sur lequel tout 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 rural, pas d'ambitions internationales. Quelque chose de simple, direct, compréhensible et percutant à la première lecture par tout un chacun est donc primordiale 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.

La processus de création du 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 plait 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.

----------------------------------------------------

Le logo final a été réalisé dans le but de respecter nos valeurs ainsi que la direction material design que nous avons choisi, 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 étudiés dans les esquisses afin de le mettre au propre. Nous avons fait une première version que nous avons soumis 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 et esthétique bien que cette critique soit subjective.

Les couleurs utilisés font parties de la palette de couleurs material design inclus dans le framework CSS Materialize.

Elle a été désignée en 2010 par Hannes von Döhren et a largement été influencé par les styles géométriques et sans-serif qui étaient populaires dans les années 20 et 30. La typo est basée sur des formes géométriques qui ont été corrigés 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.\\

**4.3) identité visuelle**

**Framework CSS :**

<https://en.wikipedia.org/wiki/CSS_framework>

<https://materializecss.com/>

<https://openclassrooms.com/fr/courses/3936801-composez-des-interfaces-utilisateurs-en-material-design/4392371-utilisez-le-framework-materialize-css>

<https://medium.freecodecamp.org/an-quick-introduction-to-material-design-using-materialize-8a9b223c64f1>

**Material design vs Flat design:**

<https://material.io/design/introduction/#principles>

<https://materializecss.com/about.html>

<https://fr.wikipedia.org/wiki/Flat_design>

<https://fr.wikipedia.org/wiki/Material_design>

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

Conception

* 1. Notions essentiels React
     1. DOM virtuel

React ne crée pas d'HTML et n'interagit pas avec le DOM (interface de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web) <https://fr.wikipedia.org/wiki/Document_Object_Model> 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 nœuds 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 virtual DOM afin d'alléger l'arbre. Ce sont des éléments React, légers, sans état et immutable qui constituent une représentation virtuel d'un élément DOM, c'est un DOM virtuel. <https://gist.github.com/sebmarkbage/fcb1b6ab493b0c77d589> Cela se matérialise comme cela : Le virtual DOM va comparer la page HTML finale au DOM virtuel et il va en déduire les opérations minimales à effectuer pour y arriver.

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>

* + 1. JSX

C'est une extension de syntaxe JavaScript crée 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.

Il est recommandé de l'utiliser avec React afin de décrire à quoi l'UI ressemblera. Il permet de résoudre le problème des appels "render()" qui peuvent vite être trop verbeux et difficile 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) <https://fr.wikipedia.org/wiki/Compilateur_source_%C3%A0_source>. Nous utiliserons pour cela Babel. <https://babeljs.io/>

* + 1. 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 apparaitre à l'écran. Cela nous permet de diviser l'UI en différentes petites pièces indépendantes et réutilisables que nous pouvons concevoir en isolation

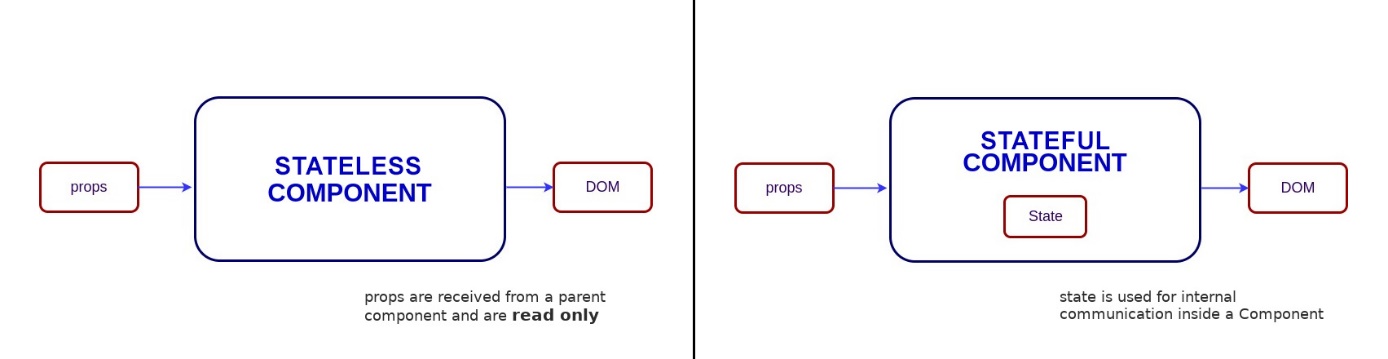
React possède deux types de composants :

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.

Composants stateful (ou à état)

Classe qui reçois 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 composant dynamiques et interactifs. Nous pouvons modifier cet état avec un simple appel à la fonction setState().



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).

* + 1. 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, quel que soit le moment et le contexte de l'appel.

<https://fr.wikipedia.org/wiki/Programmation_d%C3%A9clarative>

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

<https://medium.com/@vincent.bocquet/votre-premier-composant-react-partie-1-e3c5047402f4>

<https://openclassrooms.com/fr/courses/4664381-realisez-une-application-web-avec-react-js/4664388-decouvrez-lutilite-et-les-concepts-cles-de-react>

<https://www.apprendre-react.fr/tutorial/debutant/render-et-virtual-dom/>

<https://www.apprendre-react.fr/tutorial/debutant/les-composants/>

<https://code.tutsplus.com/tutorials/stateful-vs-stateless-functional-components-in-react--cms-29541>

<https://www.synbioz.com/blog/introduction_a_react>

<https://reactjs.org/docs/introducing-jsx.html>

<https://reactjs.org/docs/state-and-lifecycle.html>

<https://reactjs.org/docs/components-and-props.html>

CONCEPTION

# REALISATION

TESTS ET VALIDATION

CONCLUSION

* 1. Synthèse
  2. Conclusion
  3. 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, 25 octobre 2018

**Nestor Peña Patrick Audriaz**

RÉFÉRENCES

**3.2) Analyse des technologies web et mobiles modernes :**

**Javascript :**

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

« 5 raison de choisir Javascript » - Anthony Welc <https://www.youtube.com/watch?v=-0duzjYZ-zM>

« Javascript pour les débutants » - Le wagon <https://www.youtube.com/watch?v=cQZOfeKrWDs>

« What is JavaScript » - Code School <https://www.youtube.com/watch?v=cQZOfeKrWDs>

« What is JavaScript » - Programming with Mosh <https://www.youtube.com/watch?v=upDLs1sn7g4>

« What is JavaScript and where can we use it  » - Hitesh Choudary <https://www.youtube.com/watch?v=dlfrWbYk1v0>

Images - <https://insights.stackoverflow.com/survey/2018>

**Backend :**

<https://humanwhocodes.com/blog/2013/10/07/node-js-and-the-new-web-front-end/>

<https://openclassrooms.com/fr/courses/1056721-des-applications-ultra-rapides-avec-node-js/1056866-node-js-mais-a-quoi-ca-sert>

**Frontend :** <https://medium.com/tech-tajawal/modern-frontend-developer-in-2018-4c2072fa2b9c>

<https://marcobotto.com/blog/the-hitchhikers-guide-to-the-modern-front-end-development-workflow/>

<https://www.linux.com/blog/learn/2017/7/modern-day-front-end-development-stack>

<https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade>

<https://fr.wikipedia.org/wiki/HTML5>

**3.3) Comparaison des framework Javascript**

Framework définition - Techno-Science.net <https://www.techno-science.net/definition/1471.html>

What is a Framework and Why we use Frameworks? - Bitfumes Webnologies <https://www.youtube.com/watch?v=1a5VKUc0AUc>

**3.4) Comparaison des applications mobiles natives et des PWA :**

<https://www.marutitech.com/progressive-web-app/>

<https://dev.to/henrylim96/zero-to-15--building-a-nothing-pwa-in-15-minutes-258j>

<https://medium.com/@onejohi/introducing-geolocation-in-your-pwas-65713faba51a>

<https://hackernoon.com/progressive-web-apps-the-next-step-in-web-app-development-372235bf9a99>

<https://medium.freecodecamp.org/progressive-web-apps-101-the-what-why-and-how-4aa5e9065ac2>

**4.1) Synthèse des besoins des utilisateurs**

SwissMobil : Service offert à l'usager (Nicolas Schroeter)

ANNEXES

* 1. Installation de REACT et de ses modules

Setting up bases :

- Install NodeJS sur Ubuntu : <https://websiteforstudents.com/install-the-latest-node-js-and-nmp-packages-on-ubuntu-16-04-18-04-lts/>

- Create a new react app with facebook : <https://github.com/facebook/create-react-app>

- Bootstrap : <https://react-bootstrap.github.io/getting-started/introduction/>

- Router : <https://github.com/hamza-mirza/react-router>

Installation :

> cd "project-name"

> npm install --save react-bootstrap

🡪 in your **index.js** file before CSS import :

import 'bootstrap/dist/css/bootstrap.min.css';

> npm install --save react react-dom

> npm install --save react-router

🡪 create a new folder “components”

🡪 import your component in your App.js file

🡪 import About from "./components/About";

* 1. Création d'un nouveau projet React