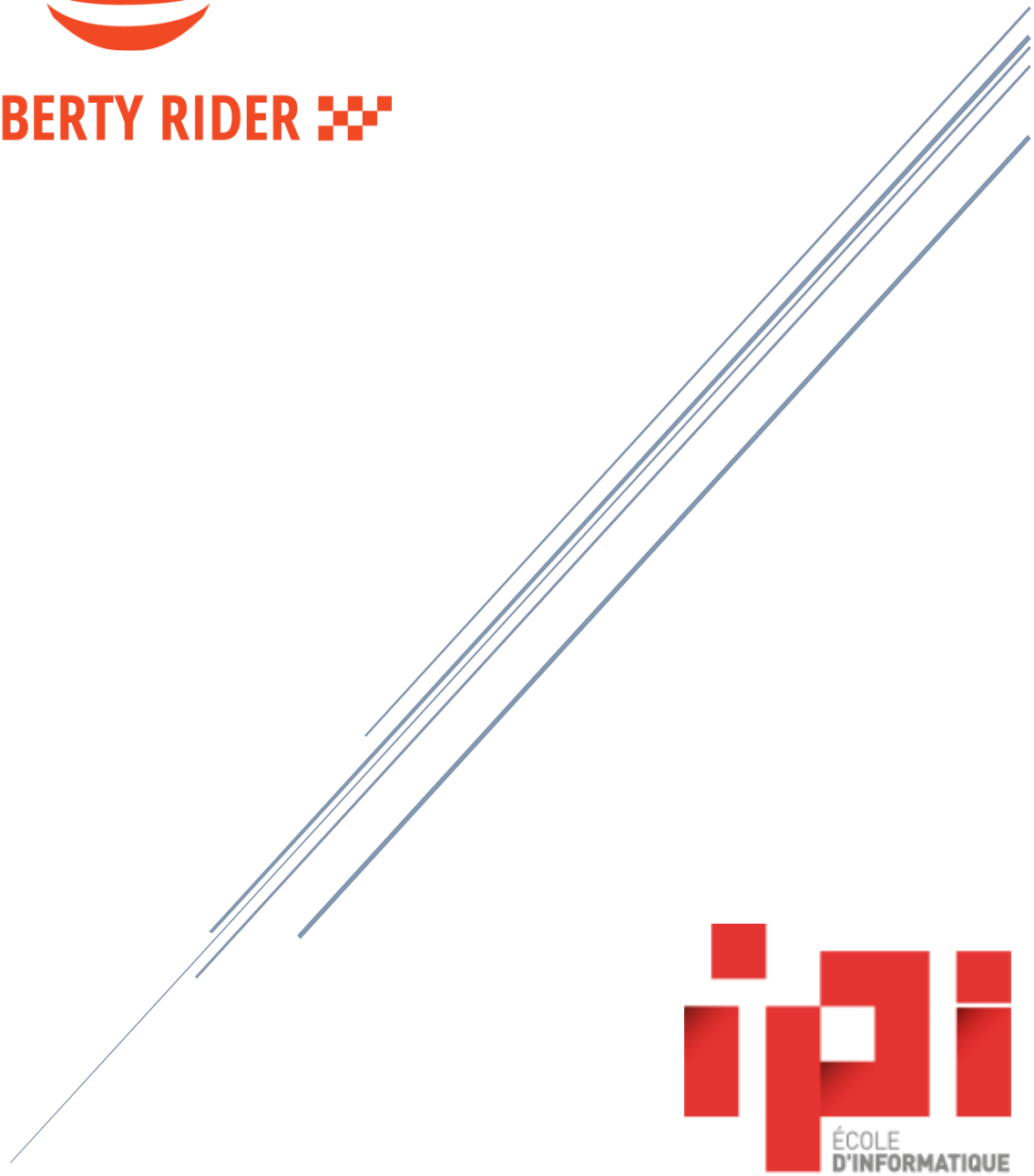
RAPPORT D'ALTERNANCE

Titre de Chef de projet ingénierie logicielle



Institut de Poly-Informatique, Toulouse

Julien Sergent

PAGE A SUPPRIMER

LEGENDE

Lorem ipsum	Texte à revoir, ajouter ou à supprimer
[image:label]	Image à insérer, décrite selon le label
[annexe:numero:label]	Insérer une annexe complétant la section

Remerciements

Grille de compétences

Gérer les processus et la qualité

- Pratiquer des audits en respectant une méthode .....
- Être capable de gérer et d'optimiser les procédures existantes en respectant les normes ISO 20 000 .....
- Formaliser des procédures et garantir leur respect en respectant les normes ISO 900 .....
- Architecturer et gérer un réseau d'entreprise.....
- Concevoir et proposer des solutions innovantes de reconstruction des processus .....

Gérer les ressources du projet

- Prévoir des ressources humaines, recruter des collaborateurs .....
- Manager une équipe, évaluer les collaborateurs .....
- Prévoir et trouver les moyens logistiques nécessaires au projet .....
- Maîtriser les aspects juridiques des contrats à passer .....

Gérer le budget des projets

- Élaborer, faire valider un budget .....
- Contrôler les tableaux de bord de suivi budgétaire.....
- Savoir rendre compte de l'état d'avancement du budget.....

Communiquer

- Élaborer un cahier des charges ou y répondre 19
- Organiser des réunions, capacité à négocier 19
- Mobiliser l'équipe pour favoriser l'avancement du projet, savoir pratiquer une écoute active et gérer les conflits .....
- Rédiger des documents et des présentations en français et en anglais.....

Manager le projet

- Être capable d'utiliser une méthode Agile.....
- Transmettre les informations et accompagner le changement .....
- Prévoir les charges de travail et le planning de réalisation, lancer l'ordonnancement et assurer son suivi.....
- Contrôler les écarts de délai et le respect des contraintes.....
- Synthétiser les indicateurs des tableaux de bord, prendre ou faire prendre des décisions pour garantir les bonnes fins du projet.....

Technique métiers spécifiques

- Posséder des compétences métier spécifiques associées aux projets à gérer.....
- Être capable de maintenir ses compétences ou connaissances métier .....

Glossaire

Entreprise

CEO : Chief Executive Officer <tiret quadratin + explications>

R&D : Recherche et Développement

B2B : Business to Business <tiret quadratin + explications>

B2C : Business to client <tiret quadratin + explications>

CPO : Chief Product Owner <tiret quadratin + explications>

CMO : Chief Marketing Officer <tiret quadratin + explications>

IMA : Inter Mutuelles Assistance

LEAN STARTUP : Est une approche spécifique du démarrage d'une activité économique et du lancement d'un produit. Dans cette optique les entreprises, en particulier les startups, cherchent à concevoir des produits et services qui satisfont au mieux la demande de leurs consommateurs, avec un investissement initial minimal.

Propre à Liberty Rider

FLOOZ : Système monétaire virtuel de l'application n'ayant aucune valeur en euro

Gestion de projet

DAILY MEETING : Réunion courte (n'excédant pas 15min) tous les matins afin de faire le point entre les différents acteurs du projet, sur ce qu'ils vont faire durant la journée et les éventuels points bloquant qu'ils ont rencontré.

SPRINT : Intervalle de temps court (1 mois maximum, souvent appelé itération), pendant lequel l'équipe de développement va concevoir, réaliser et tester de nouvelles fonctionnalités.

Technique

BACKEND : Partie d'une application qui concerne le serveur, l'architecture, base de données, etc. Tout ce qui n'est pas visible ni tangible par le client

FRONTEND : Partie d'une application qui concerne la partie graphique, l'interface utilisateur

API : Application Programming Interface. Interface disponible pour le client (mobile, application web, etc.) afin de pouvoir interagir avec le serveur.

FULLSTACK : Backend + frontend

JAVASCRIPT : Langage de programmation

SWIFT : Langage de programmation

JAVA : Langage de programmation

DEADLINE : Date limite / Date butoir (souvent appliquées pour parler d'une date limite pour rendre/terminer quelque chose)

NODEJS : Plateforme logicielle permettant d'exécuter du JavaScript côté serveur

FIREBASE : Plateforme logicielle en ligne simplifiant toute la partie base de données pour un projet

SGBD : Système de gestion de base de données

SNIPPET : Bout de code réutilisable

Outils

SLACK : Outils permettant de communiquer sur différents canaux avec les membres de notre équipe

CIRCLECI : Outils permettant d'exécuter des commandes et d'effectuer un déploiement continue

GITHUB : Plateforme en ligne permettant de versionner son code avec une stratégie git

Sommaire

Introduction

Dans le cadre de mon mastère au sein de l'école IPI, j'ai effectué deux années consécutives en alternance, dans deux entreprises différentes. Lors de mon mastère 1, j'ai effectué une année d'alternance au sein du groupe Capgemini, puis au sein de Liberty Rider pour mon mastère 2. J'ai pu changer d'entreprise car Capgemini n'avait pas désiré me prendre deux années consécutives en alternance, ce qui m'a finalement permis de pouvoir découvrir le monde de la startup en plus de celui de la SSII, afin de pouvoir effectuer un choix plus adapté à mon fonctionnement lors de la prise de mon premier emploi.

J'ai donc effectué une année d'alternance chez Liberty Rider, où j'ai occupé le poste de développeur web Fullstack. Durant cette année d'alternance, j'ai pu contribuer à plusieurs projets, qui m'ont permis d'approfondir mes connaissances techniques, mes compétences à travailler en équipe et à gérer un projet.

Les débuts ont été quelques peu difficiles, à commencer par comprendre comment fonctionnait l'entreprise, apprendre l'environnement global de l'application et savoir prendre les bonnes décisions. Rapidement, j'ai pu m'adapter à toutes ces contraintes et nouveautés afin de pouvoir apprendre pleinement tout ce dont Liberty Rider avait à m'offrir, mais aussi de pouvoir contribuer et faire évoluer cet environnement.

Liberty Rider

L'histoire

Liberty Rider prend naissance un soir, lorsque Emmanuel Petit, CEO (Chief Executive Officer) et porteur originel de l'idée, rentrait chez ses parents en moto, en empruntant des chemins sinueux, dangereux et peu fréquentés. Il s'est alors dit que s'il venait à chuter à cet endroit, il aurait de grandes chances de ne pas s'en sortir. L'idée lui est donc venu de créer une startup, avec trois de ses amis, à savoir : Julien LE, Martin D'Allens et Jérémie Fourmann.

Liberty Rider était donc né, à la tête quatre fondateurs, tous prêt à porter le projet jusqu'au bout.

Les débuts de Liberty Rider ont été quelques peu houleux. Tout d'abord, Emmanuel Petit, en charge de la gestion financière, rencontre beaucoup de difficultés à trouver des investisseurs. Une autre problématique majeure est de définir la fonctionnalité primaire de l'application, dans une démarche de lean startup.
Cette démarche, bien souvent sous-estimée, est pourtant critique pour la suite du projet, étant donné que sa mauvaise prise en compte menace la survie de la startup à moyen terme.

A force de volonté et de persévérance, la startup a pu intégrer les locaux de AtHome, un incubateur sur Toulouse. C'est à partir de ce moment que tout prit forme. Les premiers stagiaires et employés ont été recrutés, les premiers rôles ont été attribués, c'était donc les prémisses de la jeune startup comme on a la connaît actuellement.

La startup

Effectif et organisation

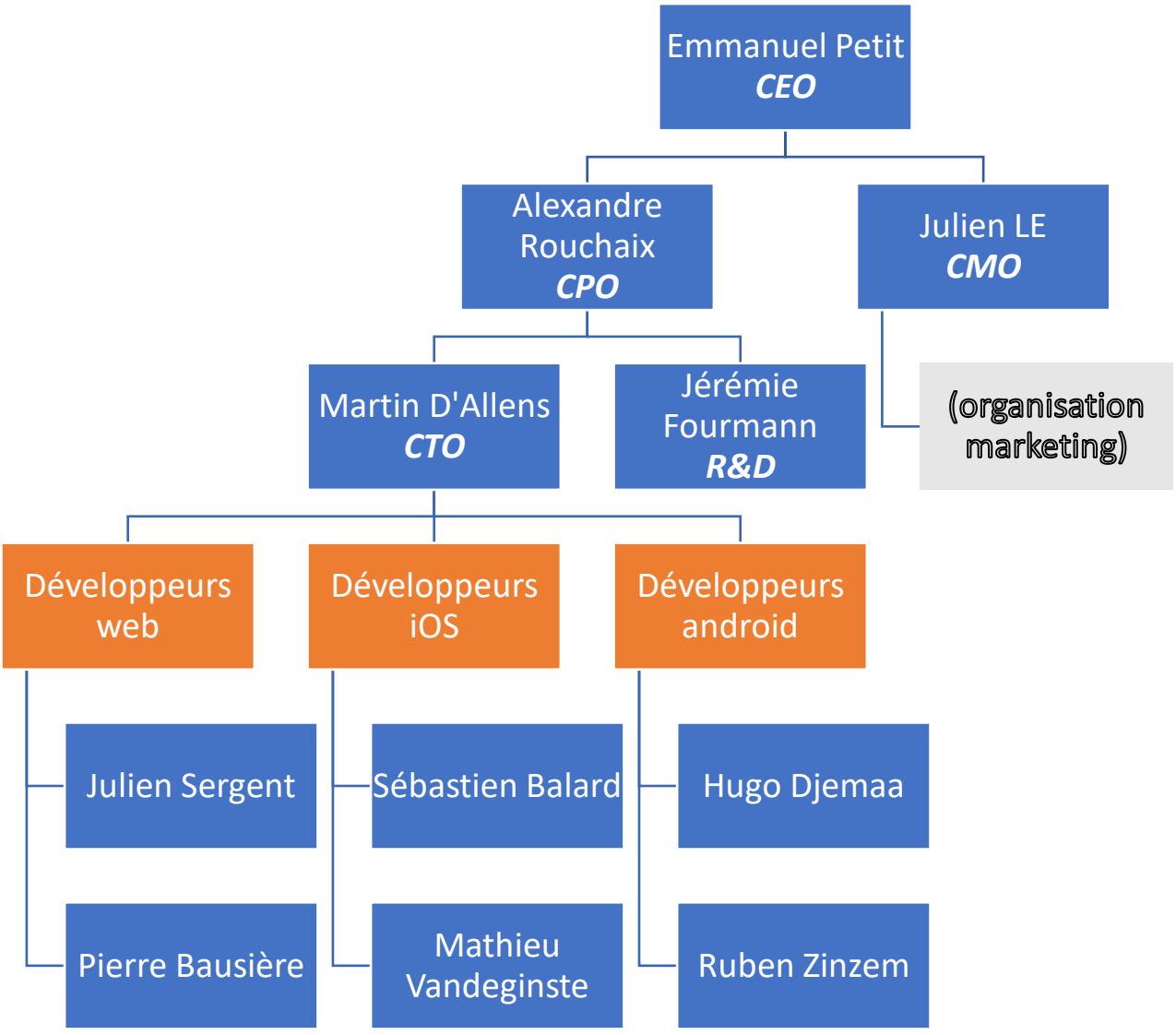
Liberty Rider est actuellement composé de dix-sept personnes ; dont le CEO, trois développeurs web, quatre développeurs mobiles, un CPO (Chief Product Owner), un responsable R&D (Recherche & Développement), quatre personnes chargées de la partie B2C (Business To Client), deux personnes chargées de la partie B2B (Business To Business) et un happiness officer.

La startup est répartie en trois grands pôles d'activités :

- Le secteur développement (regroupe les développeurs web, mobile et R&D)
- Le secteur commercial / client (regroupe les commerciaux B2B et B2C)
- Le secteur de l'entreprise elle-même (regroupe le happiness officer, le CEO)

<à compléter>

Hiérarchie


Emmanuel Petit
CEO
Alexandre Rouchaix
CPO
Julien LE
CMO
Martin D'Allens
CTO
Jérémie Fourmann
R&D
(organisation marketing)
Développeurs web
Développeurs iOS
Développeurs android
Julien Sergent
Sébastien Balard
Hugo Djemaa
Pierre Bausière
Mathieu Vandeginste
Ruben Zinzem

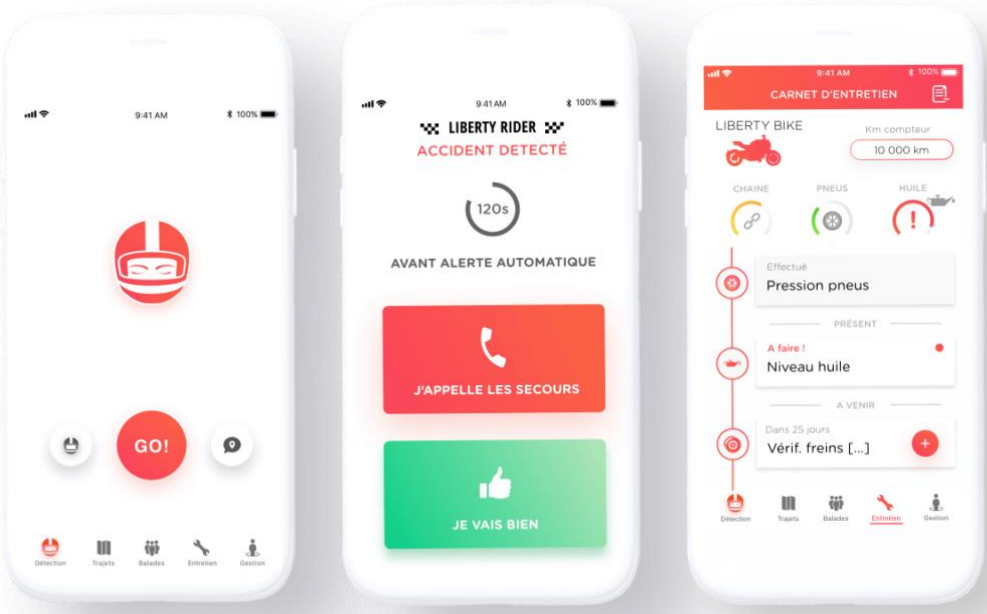
L’application

Liberty Rider est à la fois la société mais aussi l’application, cette application est donc le cœur de métier de la startup.

L’objectif principal de l’application est la détection de chute à moto. Pour ce faire, un motard peut donc télécharger gratuitement sur n’importe quel store (App Store, Play Store), s’inscrire, remplir quelques informations sur son profil et il sera enfin en capacité de pouvoir démarrer une session durant laquelle Liberty Rider sera son ange gardien. Au démarrage de la session, un sms est envoyé à certains contacts du motard (choisi par ses soins), leur indiquant qu’il vient de démarrer une session, un lien de suivi de trajet sur le web leur est fourni dans le message, ils peuvent donc observer tout type d’évènement durant la session.

En cas d’accident détecté par l’application, l’équipe de Liberty Rider est immédiatement alertée et prévient les secours, en indiquant la position GPS de l’accident et porter assistance au motard.

Cependant, l’application n’a pas pour seule fonctionnalité la détection d’accident. Elle permet aussi de pouvoir gérer le carnet d’entretien de sa moto, d’être alerté quand il est nécessaire de faire la vidange, ou tout autre changement nécessaire. L’application permet aussi d’organiser des balades avec plusieurs motards, par conséquent de tous partager le même trajet. Pour rendre l’application plus attractive, un système de ludification a été mis en place. Ce système consiste à gagner de la monnaie virtuelle, appelée « flooz ». Le motard peut gagner des flooz lors de ses trajets, du remplissage de son profil, de sa participation à des balades, ou bien en être l’organisateur. Les flooz obtenus permettront au motard, de commander des cadeaux sur le site de la boutique Liberty Rider (des casques, tasses, blousons, etc.).



Vision, mission et ambition

Le projet Liberty Rider est basé sur un secteur nouveau, avec peu de concurrents, mais aussi où le marché n'est pas encore à l'écoute ni habitué à voir naître ce type d'application dans leur écosystème. C'est pourquoi la vision de la startup est de devenir le principal leader dans le domaine de la détection de chute à moto, mais aussi dans le domaine plus général de la moto et notamment de la sécurité.

Sa mission est donc de savoir faire éclore ce nouveau marché, convaincre la grande majorité des motards (français dans un premier temps) d'installer l'application afin d'assurer leur sécurité, mais aussi de pouvoir rassurer leurs proches. Tout ceci s'accompagnera d'une croissance de communauté, qui permettra de pouvoir faire naître de nouvelles fonctionnalités, comme le partage et proposition de balades au sein de la communauté.

Aujourd'hui l'application ne fonctionne qu'exclusivement en France, pour des raisons de proximité avec les secours et d'échanges avec les utilisateurs, mais il est envisagé, à moyen terme, d'une expansion en Europe de l'application, grâce notamment à un nouveau partenariat avec IMA (Inter Mutuelles Assistance) qui permettrait à Liberty Rider de pouvoir agir en termes d'intervention des secours en cas d'accident détecté dans des pays d'Europe.

Clients et partenaires

Liberty Rider est une application gratuite destinée aux particuliers, elle repose sur un modèle B2C, cependant comme le marché n'est pas encore pleinement ouvert pour ce nouveau type d'application, Liberty Rider collabore avec des partenaires déjà présents sur différents types de marchés de la moto (Honda, Scorpion, GS27, etc.).

De cette manière, la startup peut donc utiliser les données récoltées auprès de leurs utilisateurs pour les proposer à ses partenaires afin que ces derniers puissent mieux cibler leur clientèle, promouvoir des produits aux bonnes dates et ainsi optimiser leurs campagnes publicitaires.

La startup s'entoure aussi d'investisseurs, lui permettant de pouvoir augmenter ses effectifs régulièrement et assurer l'augmentation de charge des nouveaux utilisateurs qui s'accompagne de plus de bugs, de mécontentement, d'accidents et de nouvelles fonctionnalités à développer.

Tout ceci engendre une importante charge de travail pour les développeurs, qui se retrouvent submergés de travail et il devient donc nécessaire de recruter plus d'effectif.

Contexte et problématiques

Gestion de projet

L'organisation au sein d'une startup est bien souvent chaotique, et Liberty Rider n'y échappe pas.

A mes débuts dans la startup, j'ai commencé par le projet *Gamification*, le but du projet était de fournir une boutique dans l'application, où le motard pouvait y dépenser ses flooz gagner durant ses sessions avec l'application, le tout afin de rendre l'application plus ludique, et favoriser la rétention des utilisateurs. Ce projet était malheureusement dirigé par une personne dont les compétences n'était pas la gestion de projet, ce qui est bien souvent le problème dans une startup, on ne peut pas se permettre d'embaucher des compétences à chaque besoin. J'ai rapidement ressenti les problématiques engendrées dès les débuts du projet : manque de spécifications qui nécessitait des allers-retours constants avec le chef de projet afin de connaître le cadre des fonctionnalités à développer, manque de connaissances globales qui engendrait un retard conséquent sur les deadlines, et bien souvent des parties de codes à totalement refaire car elles n'étaient pas conformes aux attentes du projet.

En ce qui concerne l'organisation au sein de l'équipe de développeurs, on s'organisait le plus souvent par oral, en mettant le plus possible de comptes rendus sur Slack, tout en essayant de se mettre d'accord entre les équipes web et mobiles.

Il est donc convenable de dire qu'au début de mon alternance au sein de Liberty Rider, la gestion de projet était quelque peu empirique, mais fort heureusement tout ceci à rapidement évoluer.

Environnement technique

Les solutions techniques adoptées chez Liberty Rider étaient les suivantes :

[image:technologies logos]

WEB : NodeJS, Express, JavaScript, Firebase

MOBILE : Java, Swift

DEVOPS : CircleCI, GitHub, Asana

ARCHITECTURE : Amazon Web Services (AWS)

La startup avait donc fait le choix d'utiliser des technologies modernes, mais tout ceci n'était pas sans contrepartie, puisque en effet l'utilisation de Firebase a engendré de lourdes problématiques quant à la manière de pouvoir traiter les données utilisateur (filtrer, réorganiser, etc.), il est même devenu impossible de pouvoir récupérer certains nœuds de données sous crainte de faire tomber la base de données pour dix minutes, rendant l'application totalement inutilisable.

De plus, la maintenabilité et l'évolutivité étaient parfois compromises dues aux dettes techniques trainantes. Tout ceci sera rapidement remis en question et de lourds projets de migration viendront réorganiser le système.

<Page de garde de l'annonce du projet>

Nouvelle version de détection d'accidents

Contexte et objectifs

Liberty Rider est avant tout une application de détection d'accident, son but est donc d'être l'application la plus performante possible sur ce sujet, et pour ce faire des algorithmes complexes doivent être mis en place.

J'ai donc commencé chez Liberty Rider sur ce premier projet, la nouvelle version de détection d'accident. Ce projet avait pour but d'améliorer les contraintes et problématiques présentent, à savoir des fausses détections, ou parfois même des non détections alors qu'il y avait accident. Raison supplémentaire, la startup était en passe de concrétiser un partenariat avec IMA, qui serait en charge d'intervenir sur le lieu de l'accident une fois que nous leur aurions transmis l'alerte de l'accident et la localisation.

Les nouveaux enjeux de l'entreprise étaient donc importants pour cette dernière, car il s'agit du cœur même de l'application qui en fait sa réputation, mais aussi de la conséquence directe de leur première levée de fond ayant comme objectif de pouvoir en avoir d'autres par la suite.

D'autres objectifs étaient attendus à l'issu de ce projet. Une nouvelle architecture du code côté serveur, afin de rendre hautement disponible l'API d'accident, qui est un des points clés de l'application et qui ne peut pas se permettre d'être indisponible pour une raison quelconque sous peine de rendre l'application privée de sa fonction première et par conséquent l'incapacité à pouvoir secourir des motards accidentés.

Un autre objectif de ce projet était d'amorcer la migration et l'insertion d'un nouveau backend, afin de commencer à se détacher au fur et à mesure de Firebase. Il était important de garder à l'esprit que Firebase coûtait 700€/mois, ce qui n'est pas négligeable quand on est une startup, mais aussi du fait de son manque de souplesse et de prise de contrôle, il provoquait des problèmes de performance sur tout le réseau de l'application car certaines des requêtes dans l'application (par exemple la possibilité à un utilisateur de pouvoir visionner tous ses anciens trajets) entrainé un ralentissement de Firebase du fait d'un traitement de beaucoup données, cette problématique avait pour conséquence de rendre l'application indisponible pendant dix minutes environ, un problème majeur.

Liberty Rider étant dans les locaux de l'incubateur AtHome, au milieu de beaucoup de personnes et d'autres startup, nous étions souvent dérangés par le milieu de travail, afin de

pouvoir mener à bien ce projet et pouvoir être concentré pleinement dessus, nous avons décidé de partir une semaine, une dev week, dans une maison isolé à Pau. Cette semaine avait donc pour but de dégrossir le projet, en produire les spécifications et commencer à les implémenter afin d'obtenir une première version bêta sur mobile.

Travail R&D en amont

Comme expliqué dans le chapitre précédent, la détection d'accident est le cœur de l'application, et n'est pas une des parties les plus simple à réaliser. En effet il s'agit d'algorithmes complexes visant à analyser correctement les signaux envoyés par le téléphone dans un but de détecter et reconnaître un vrai accident d'un faux.

Le problème est que de tels algorithmes requièrent un important travail préparatoire, il a donc fallu réaliser des recherches, préparer tous les aspects des nouveaux algorithmes, anticiper les contraintes techniques, et rédiger des documentations techniques afin de simplifier l'implémentation pour les développeurs.

Ces travaux de recherches ont été effectués pendant une année avant leur implémentation définitive, des tests en amont ont été réalisés pour ajuster les paramètres des algorithmes et les améliorer.

Jérémie Fourmann, responsable R&D et co-founder, était en charge de la réalisation des travaux.

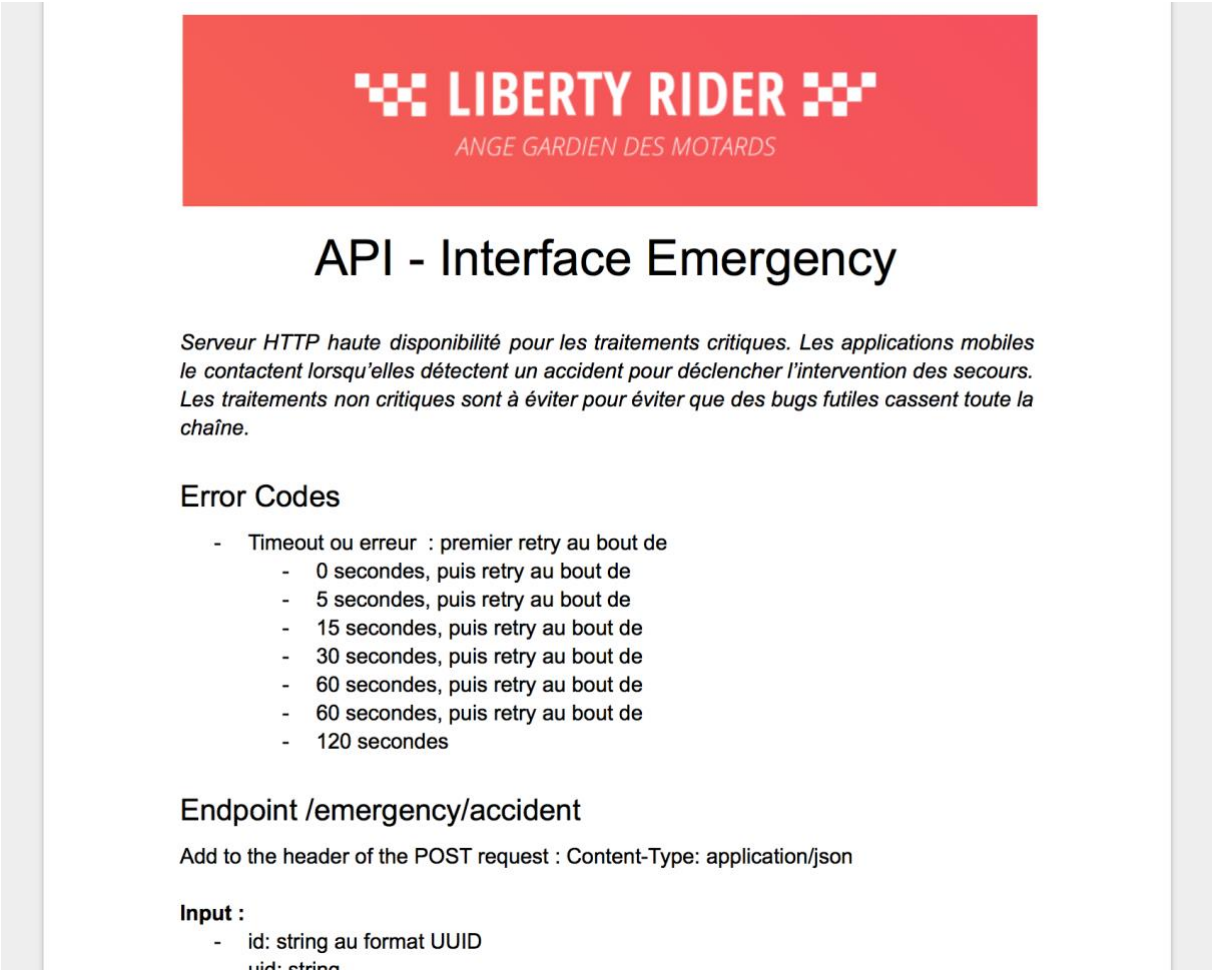
Point technique : Les recherches ont été réalisés à l'aide du langage de programmation Python.

Spécifications

Rédaction

Avant de commencer le projet, et afin de mieux le réaliser, nous avons décidé de rédiger des spécifications, pour mieux prévoir tous les aspects de ce dernier, les éventuels points bloquants, le temps que nous allions y consacrer et des documents en anglais et français (anglais pour la partie technique, axés développeurs, et français pour la partie commerciale) permettant de recenser la manière dont serait menés les travaux, les temps attribués aux différentes tâches, et les différents comportements du nouvel algorithme.

Pour mieux réaliser ces spécifications, nous avons donc effectué plusieurs réunions, chacune avait pour but de mieux prévoir toutes les frontières de la nouvelle version de détection d'accidents (ce qui allait devoir être migré, modifié et ajouté), et aussi se mettre d'accord sur les aspects techniques, et comment nous allions procéder.
J'ai donc assisté et animé des réunions de planifications, contribué à rédiger les spécifications du cahier des charges, ainsi que savoir discuter avec l'équipe des éventuels conflits, problèmes, que nous aurions pu rencontrer lors du développement de la solution.



LIBERTY RIDER
ANGE GARDIEN DES MOTARDS

API - Interface Emergency

Serveur HTTP haute disponibilité pour les traitements critiques. Les applications mobiles le contactent lorsqu'elles détectent un accident pour déclencher l'intervention des secours. Les traitements non critiques sont à éviter pour éviter que des bugs futiles cassent toute la chaîne.

Error Codes

- Timeout ou erreur : premier retry au bout de
  - 0 secondes, puis retry au bout de
  - 5 secondes, puis retry au bout de
  - 15 secondes, puis retry au bout de
  - 30 secondes, puis retry au bout de
  - 60 secondes, puis retry au bout de
  - 60 secondes, puis retry au bout de
  - 120 secondes

Endpoint /emergency/accident

Add to the header of the POST request : Content-Type: application/json

Input :

- id: string au format UUID
- uid: string

Aperçu du document des spécifications de l'API du serveur en charge de gérer les accidents (voir Annexe 2)

Architecture

Une des résultantes des spécifications est l'architecture des services web exposés par les serveurs de Liberty Rider.

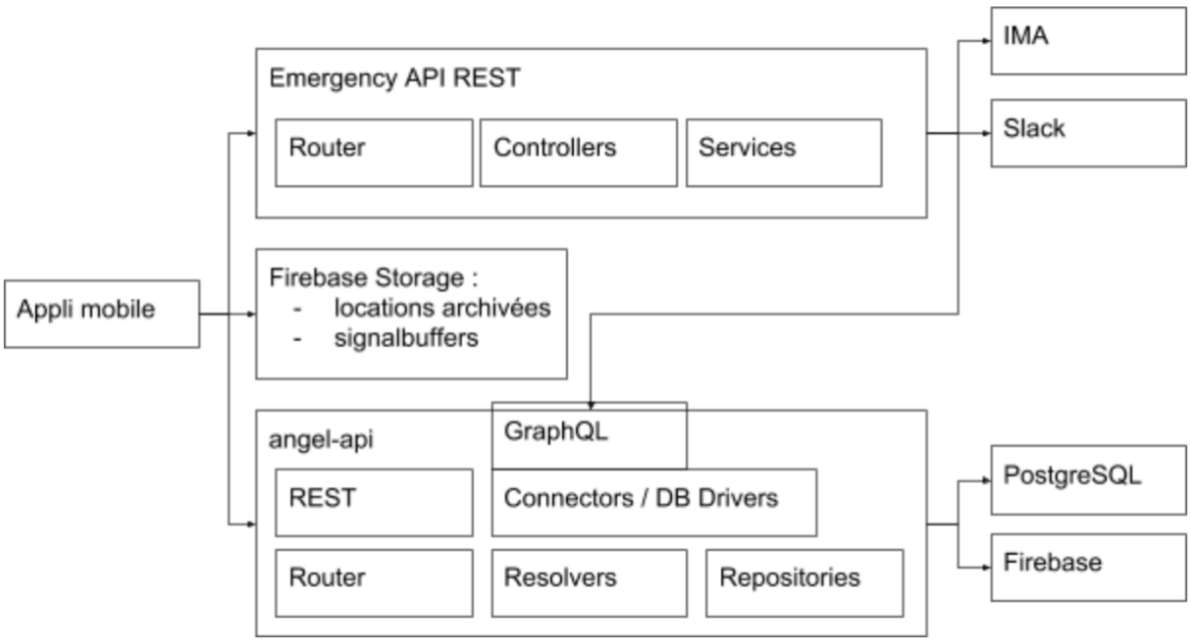


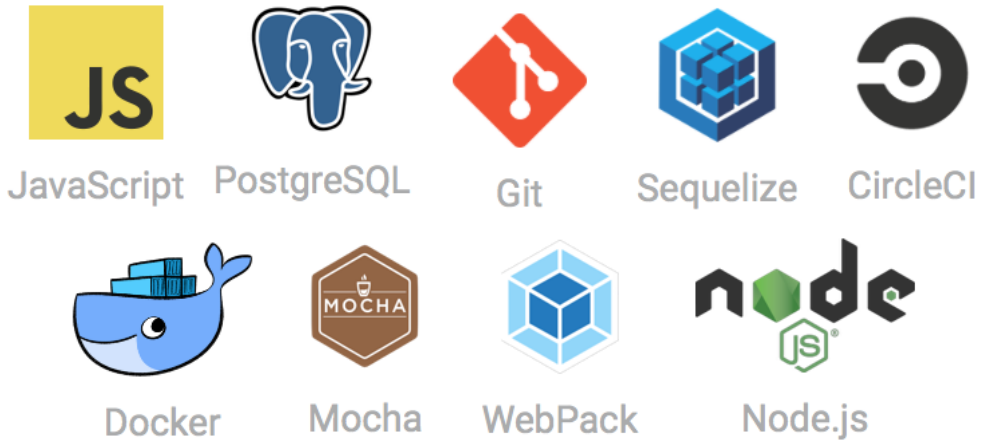
Schéma de l'architecture des différents services web exposés par les serveurs de Liberty Rider

Cette architecture est l'issue de plusieurs réunions de l'équipe web.

Dans le précédent schéma on représente donc les clients qui utilisent les services web par la cadre « Appli mobile ». Ces clients vont pouvoir consommer les services qui leur sont exposés, à savoir la partie accident avec le serveur « Emergency », les services globaux de l'application avec le serveur « angel-api », et tout ce qui concerne le stockage d'images sera disponible avec le service « Firebase Storage ». On remarque également qu'en sortie du schéma, nous avons « PostgreSQL », qui est un SGBD (Système de gestion de base de données), et « Firebase », qui est un système de base de données temps réel. Ces deux sorties, qui font le même travail, sont la preuve de la migration du système de Liberty Rider, cet aspect sera présenté en détails dans la présentation du projet **Migration et nouvelles fonctionnalités**.

Environnement technique

L'environnement technique choisi pour ce projet est le suivant :



L'environnement présenté ci-dessus concerne uniquement l'environnement web choisi par l'équipe web.

Nous avons décidé de garder quelques éléments déjà utilisés dans des projets de Liberty Rider (à savoir Docker, CircleCI, JavaScript, WebPack et Git). En ce qui concerne les nouveaux éléments techniques ajoutés (Sequelize, Mocha,et PostgreSQL), ils ont été choisi de manière élaboré. Leur choix provient de leur popularité au sein de la communauté des développeurs ; Sequelize est le nouvel ORM (Object-Relational Mapping) de tous les systèmes de Liberty Rider, du fait de sa popularité au sein de la communauté des développeurs ; PostgreSQL est un SGBD fiable et robuste, Mocha est un outil permettant d'exécuter des tests écrit avec le langage JavaScript.

Les raisons de ces choix sont dans un but précis. Dans un premier objectif, harmoniser les outils sur tout le système de Liberty Rider, dans un second temps, utiliser des outils robustes et fiables, et dernièrement, entrer dans un cadre DevOps pour aligner les différentes équipes grandissantes (par exemple l'utilisation de Docker permet d'augmenter l'environnement DevOps).

Réalisation et gestion

Équipe

Pour réaliser ce projet nous étions répartis en trois équipes.
L'équipe R&D a dû effectuer des travaux en amont, permettant de pouvoir mettre en avant les différents aspects du projet et les acteurs qui seraient impliqués.
Dans les différents acteurs, nous retrouvons l'équipe web, constituée de Martin D'Allens et moi-même, chargée de la réalisation de la partie services web.
Une dernière équipe était nécessaire, l'équipe mobile. Son but était de réaliser et implémenter le nouvel algorithme de détection d'accident.

Les équipes projet étaient donc les suivantes :

- **Web**
- **Mobile**
- **R&D**

Développement

Les spécifications étant prêtes, c'était donc le de commencer à développer le projet. Mon ressenti, à la suite des premiers jours de développement, était plutôt pessimiste. En effet il s'agissait d'un de mes premiers projets avec Liberty Rider, j'avais donc peu de connaissances métiers sur leur système, il m'était difficile de pouvoir prendre des décisions seul face aux questions et demandes des autres équipes. Pour mieux appréhender la suite, à chaque problème ou doute rencontrée, je posais systématiquement la question à Martin, de cette manière j'ai pu rapidement évoluer sur l'écosystème Liberty Rider afin de pouvoir travailler en pleine autonomie.

Le développement du projet s'est finalement bien déroulé, j'ai pu mettre en pratique mes connaissances pour structurer les différents projets qui allaient accueillir les différents services web, mais aussi accroître grandement mes compétences sur la partie DevOps, puisque nous possédons une gestion complète de conteneurs, via Docker, de tous les projets web.

En ce qui concerne la gestion du projet, nous avons mis en place une stratégie semblable au daily meeting. J'avais proposé cette solution à Martin à mes débuts dans l'entreprise, car elle n'était pas pratiquée et je trouve sa pratique importante pour mieux savoir où vont les équipes, et par conséquent, le projet.

<Parler de l'automatisation des tests, avec CircleCI : Unit, integ, system>

Revue de planning et des spécifications

Aucune stratégie de gestion de projet n'était réellement appliquée, mais on peut considérer que notre méthode appliquée durant ce projet, se rapprochait d'une gestion agile. En effet, tous les matins, nous faisions un retour sur le planning, permettant de savoir si les tâches prévues la veille ont pu être terminées, et les tâches restantes.

Comme les spécifications n'étaient pas figées ni complètes, des réunions étaient organisées durant lesquelles on devait présenter ce que nous allons ajouter et sous quel format, afin d'en débattre et se mettre d'accord entre les différentes équipes. J'ai donc pu rédiger des spécifications en anglais, et les présenter au reste de l'équipe, et savoir défendre mon point de vue quand cela était nécessaire.

Ces échanges réguliers ont permis une avancée rapide et précise du projet, les équipes savaient communiquer et débattre sur un sujet afin d'améliorer la qualité du projet.

Contraintes

Cependant, la réalisation de ce projet ne s'est pas déroulée sans accrocs. Plusieurs contraintes se sont révélées lors de la réalisation de ce dernier, notamment quant à la gestion de projet. En effet, aucune gestion de projet n'était prévue, des stratégies de suivis et d'amélioration de qualité ont été mises en place, mais sans structure réelle, ni responsable de leur gestion.

Sans planning réel, de sprints et de tâches, il devenait rapidement difficile de prévoir la fin d'une tâche, ayant pour conséquence de devenir pratiquement impossible de savoir si le projet était en retard ou en avance sur les maigres planifications prévues. Il en découlait cependant un avantage, la rapidité d'exécution. Même si les gestions de projet agile sont pertinentes et répondent à un grand nombre de besoin, il n'en est pas moins qu'à très court termes elles retardent légèrement le projet du fait du temps investi dans tous les préparatifs de ces différentes phases (sprint planning, rétrospectives, sprint review). Dans le cadre de la dev week, il s'agissait d'avoir une version bêta en pratiquement une semaine.

D'autres contraintes sont venues perturbées l'équilibre du projet, et notamment le manque de spécifications claires et constructives. Cela a mené à des distorsions au sein des équipes mobiles (équipe iOS et équipe Android), qui n'avançaient pas à la même vitesse, il arrivait donc parfois qu'une équipe vienne remettre en cause des spécifications après leur

implémentation par l'équipe précédente, ce qui avait pour conséquence un retard imprévu sur le projet.

Mise en production

La mise en production est une étape cruciale dans le cycle de vie d'un projet. Cette étape, bien souvent sous-estimée en termes de temps et de charge, se révèle être un vrai casse-tête pour les développeurs, pour qui tout marche en local mais quand on déploie le projet en production, rien ne fonctionne comme prévu, voire rien du tout.

Architecture Amazon Web Services

Pour les besoins du projet, nous avons dû prévoir deux serveurs sur AWS (Amazon Web Services), un serveur pour gérer uniquement la chaine de détection d'accident (ce serveur est isolé afin d'éviter tout disfonctionnement de la chaine d'alerte si une erreur survient sur le serveur), et un autre serveur pour la gestion des services web.

Dans le cadre de la mise en production, j'étais en charge de réaliser l'architecture des serveurs sur la plateforme AWS. J'ai donc alloué des machines virtuelles en haute disponibilité selon la politique de AWS, accompagné d'un équilibreur de charge permettant de répartir les charges serveurs, tout ceci dans un but d'assurer une disponibilité continue du serveur en charge de la chaine d'alerte pour les accidents.
Ces machines virtuelles sont regroupées par groupe, appelés cluster. Chaque groupe correspond à un environnement dédié (l'environnement de développement, de validation et de production), nous avons au total trois environnements différents. Dans ces groupes, nous retrouvons des services, qui démarrent des tâches. Ces tâches sont chargées de démarrer les conteneurs Docker et de les surveiller. Ces conteneurs contiennent tout le code de l'application. Je les ai utilisés afin d'améliorer l'environnement DevOps, et de maintenir un environnement similaire entre celui des développeurs et de production, de cette manière on s'assure de réduire les erreurs d'environnement lors de la mise en production.

Une fois l'architecture réalisée, j'ai dû mettre en place la passerelle permettant à notre outil d'intégration continue de pouvoir publier les conteneurs Docker sur les machines AWS. L'outil d'intégration continue qui est utilisé dans ce contexte est CircleCI.

La mise en production est bien souvent difficile, et Liberty Rider ne fait pas exception. Nous avons éprouvé des difficultés à obtenir le même résultat en production que sur l'environnement local des développeurs, mais grâce à un écosystème DevOps s'appuyant sur la technologie Docker, les problèmes étaient moindres et nous sommes rapidement arrivés au résultat attendu.

<diagramme archi aws dire que les explications seront dans la section Mise en production, page xx>

Analyse des erreurs

Savoir analyser ses propres erreurs et, de manière plus générale, les erreurs d'un projet, est une pratique connue mais peu pratiquée. En effet, il est, chez certains, peu agréable de se remettre en doute, ainsi que ses pratiques, et pourtant cette étape permet de s'améliorer continuellement afin de parfaire les prochaines pratiques.

Dans le cadre de ce projet, c'est ce que nous avons fait. Dans le contexte d'une startup, où l'on cherche encore nos habitudes de travail, où l'on essaie de construire un environnement stable et durable, il est primordial de savoir analyser ses erreurs et en tirer profit pour les fois suivantes.

En ce qui concerne ce projet, nous avons réalisé des réunions de retours de projet, permettant à chacun de s'exprimer sur les côtés positifs et négatifs et surtout les axes d'amélioration à approfondir.

Pour ma part, j'ai pu en discuter plus longuement avec Martin. Je lui ai fait part de plusieurs points bloquant durant le projet, mais surtout d'axes d'amélioration à suivre.

Un premier point bloquant que j'ai pu identifier est : **les délais**.
Nous n'étions pas dans une gestion agile, aucune notion de sprint n'était présente, nous n'avions aucun moyen de suivre un quelconque support permettant de la visibilité sur le projet (date de fin, vélocité de l'équipe, sommes-nous en avance ou en retard sur le projet).

Un autre problème que j'ai identifié : **la surcharge**.
Induit directement du premier point bloquant, la surcharge découle d'une mauvaise gestion du temps, nous avons donc dû réaliser des surcharges de travail afin de pouvoir terminer le projet dans les « délais » prévus initialement.

Un dernier point bloquant que j'ai identifié : **complexité et difficulté inégales selon la plateforme**.
Dans une équipe agile, chaque membre doit avoir un minimum connaissance sur chacun des domaines, permettant une meilleure prise de conscience de la difficulté sur chaque domaine, des tâches à réaliser.

Pour éviter à nouveau ces problèmes, j'ai fait part à Martin qu'il serait bon de mettre en place une gestion agile (daily meeting, sprints) pour les prochains projets, de telle manière à pouvoir mieux estimer les charges, les délais, et favoriser la collaboration inter-équipes.
Cette solution a été mise en place dans le prochain projet, que je présenterai, **Migration et nouvelles fonctionnalités**.

<Page de garde de l'annonce du projet>

Migration et nouvelles fonctionnalités (20p)

Contexte et objectifs (0.5p)

Après plus de deux ans de vie, l'application Liberty Rider a pris de l'ampleur, son écosystème s'est enrichi, de bons et de mauvais aspects. Personne n'est capable de faire les bons choix dès le début, il arrive souvent de vouloir remettre en question nos anciennes décision. Durant ces deux premières années, Liberty Rider a accumulé des dettes techniques, c'est à dire de mauvais choix technique, et avec la croissance des effectifs il était important de corriger ces défauts pour repartir sur de meilleures bases. Il s'agissait du premier objectif de ce projet.

À la suite des levées de fonds, des résultats étaient attendus par les investisseurs. Il était donc indispensable de produire de nouvelles fonctionnalités pour la communauté, afin d'augmenter la rétention des utilisateurs et le nombre de clients (notons qu'un utilisateur est une personne utilisant une application, un client est un utilisateur qui paie pour utiliser l'application, dans le cas de l'application Liberty Rider, un client est un utilisateur premium).

Ce projet a donc été réalisé dans le but de répondre à deux objectifs :

- Migrer l'écosystème actuel vers de meilleures bases
- Apporter de nouvelles fonctionnalités pour augmenter le nombre de clients pour assurer la survie de la startup

Préparations (3p)

Cahier des charges

En ce qui concerne la partie migration du projet, dans un but de mieux prévoir cette partie, qui est lourd d'un point de vu développement mais aussi d'un point de vue gestion, un document de recensement et de suivi a été écrit sous forme de tableau.
Ce document a pour but de pouvoir vérifier à tout moment l'avancement de la migration et ce de manière précise, puisque le cahier des charges recense toutes les fonctionnalités existantes, et leur état d'avancement dans la migration.

	Table	Field	Sub	Objectif	Schema	Resolvers	Android	iOS	Admin	Webapp	Tracking	Worker	API v1	Import	Commentaire
6	users								read						
7		traveledDistance		remove											
8		level		remove						read				later: import badge avant mise en prod de gamificationgraphql ; la poule ou l'oeuf ??	
9		points		remove											
10		firstConnection		remove										à importer depuis firebase auth (et renommer UserCreation ?)	
11		lastConnection		remove											inutile
12		provider_email		remove											inutile
13		firebaseToken		remove											
14		fcm_instance_id_token		remove											
15		previous_uid		remove											
16		auth_provider		remove											à remplacer par currentIdentity.provider, pour customer care
17		phone		remove											
18		emails		remove											
19		rank		remove											
20		profile_country		remove											
21		profile_first_name		prio1	prod	prod	prod	pr		read	read	read		TODO	TODO ios et android : stop using /profiles
22		profile_last_name		prio1	prod	prod	prod	pr		read	read	read		TODO	
23		profile_email		prio1	prod	prod	prod	pr		read		read		TODO	
24		profile_gender		prio1	prod	prod	prod	pr							
25		coins		prio1	W=TOD	R=prod W=TODO	blocked			write					
26		coverPictureUrl		prio1	prod	prod	prod								
27		profilePictureUrl		prio1	prod	prod	prod	pr		read	read				
28		profile_phone		prio1	prod	prod	prod	pr	read						
29		profile_birth_date		prio1	prod	prod	prod	pr							
30		bike_type		prio2	prod	prod	prod								
31		bike_brand		prio2	prod	prod	prod								
32		bike_series		prio2	prod	prod	prod								
33		bike_engine		prio2	prod	prod	prod								
34		bike_year		prio2	prod	prod	prod								
35		bike_km		prio2	prod	prod	prod			TODO badge ?					
36		bike_buy_date		remove											
37		profile_health_info		prio2	prod	prod	PR#930								TODO api : prendre en compte pref "store_health_data" ?

Aperçu du document chargé de recenser et suivre l'avancement de la migration (Annexe 5)

J'ai été en charge de compléter la partie web de ce document avant le début de la migration, puis de le mettre à jour par la suite lors de notre progression durant le projet.

Cependant, ce document n'est pas la seule ligne de conduite du projet, car la seconde partie, « nouvelles fonctionnalités », n'est pas enclin à être suivie de cette manière. C'est pour cette raison qu'une méthodologie agile a été mise en place, pour diriger de manière uniforme et en harmonie les deux parties du projet. Je présenterai plus en détails cette méthodologie dans la partie **Réalisation et gestion**.

Recrutement de compétences

Pour faire face à ce projet conséquent, Liberty Rider a dû recruter de nouveaux collaborateurs. Le besoin le plus important était la nécessité de recruter un CPO. En effet ce dernier est en charge de planifier les stratégies d'évolution du produit, il devient donc le référent de toute décision métier, et permet de mieux diriger les décisions du fait de sa vision globale du produit.

Dans un second temps, il était nécessaire de recruter de la main d'œuvre technique pour mieux assurer les charges de développement déjà présentent, mais surtout celles à venir. Liberty Rider possède une politique de communication interne importante, permettant aux salariés de pouvoir exprimer leur avis sur une grande quantité de sujets. Grâce à cette politique, j'ai pu assister aux entretiens de recrutement des futurs développeurs avec le CTO, et par la suite donner mon avis sur l'étude du candidat. A l'issue de ces entretiens, deux candidats ont rejoint la startup : Sébastien Balard (développeur iOS) et Pierre Bausière (développeur web).

Roadmap

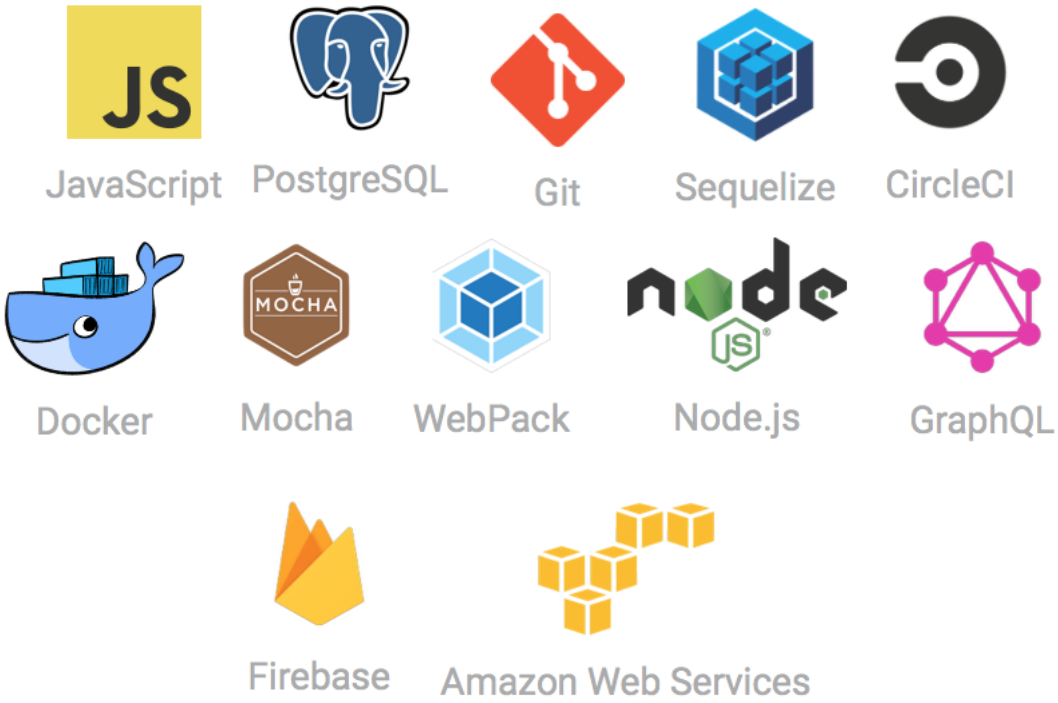
Un des premiers travaux effectués par le CPO était la roadmap. Cette feuille de route est la première de Liberty Rider. Elle fournit une vision à moyen termes sur comment va évoluer l'application et son écosystème. La stratégie derrière cette roadmap est d'augmenter la rétention des utilisateurs sur l'application, apporter de nouvelles fonctionnalités et faire accroitre la communauté de Liberty Rider pour gagner en notoriété.

Roadmap de Liberty Rider

Mise en place de l'environnement technique (3p)

Environnement technique

En ma qualité de développeur, j'ai eu la responsabilité (en collaboration avec Martin), de pouvoir choisir les technologies qui allaient être utilisées dans pour la partie backend du projet :



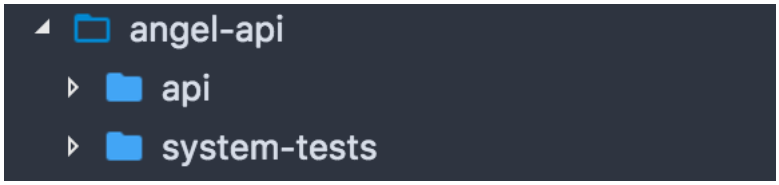
Étant un développeur passionné et faisant de la veille technologique, j'ai donc été force proposition quant à l'utilisation de nouvelles technologies. En effet, dans un contexte de startup ainsi qu'un nouveau projet, il me semblait pertinent de choisir ces technologies, pour plusieurs raisons :

- Technologies récentes mais ayant fait leurs preuves
- Communauté grandissante pour chacune d'elle (en cas de problème on peut compter sur le soutien de cette dernière)
- La plupart sont développées par des grands noms du web : GraphQL a été créé par Facebook, Amazon Web Services par Amazon et Firebase par Google
- Modernes et permet de répondre à plus de problématiques

On a cependant été contraint de devoir utiliser Firebase et PostgreSQL en même temps, pour des raisons de migration. En effet il était impossible de migrer complètement et de

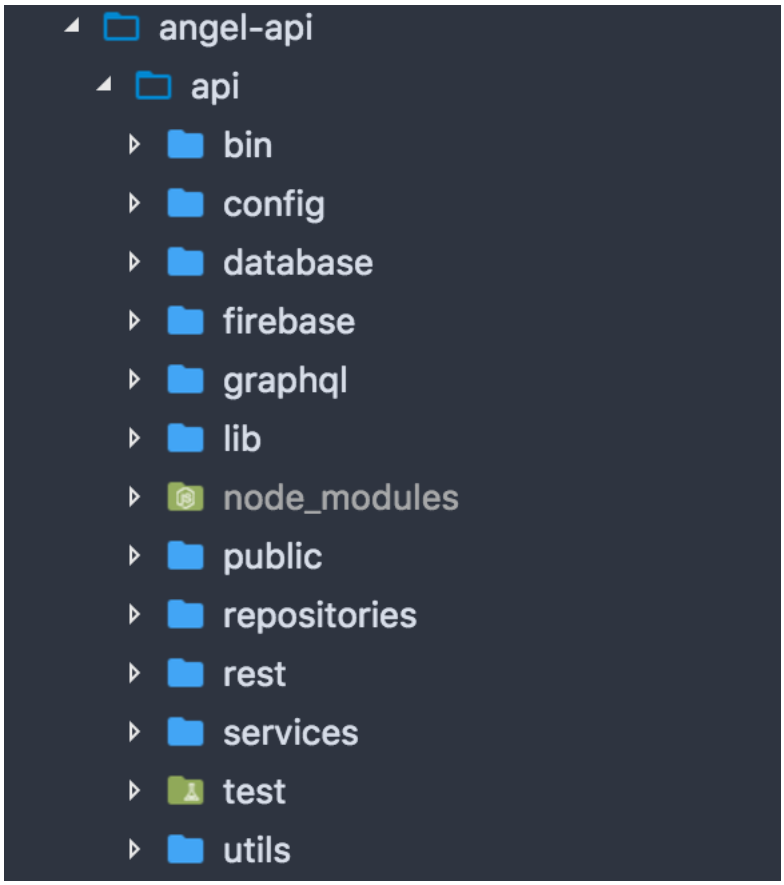
manière rapide le système Liberty Rider, nous avons donc choisi de réaliser une migration douce qui regrouperait les deux mondes dans une premier temps, puis se dirigerai au fur et à mesure vers la nouvelle solution.

Structure du projet



Aperçu de la racine du projet

Voici la structure générale du projet backend. Dans le dossier `api` se trouve le code métier de l'application, c'est à dire le code propre à Liberty Rider, et qui répond à ses besoins. Dans le dossier `system-tests`, on va retrouver tous les tests système des services web exposés par l'API. Un test système a pour but de tester l'intégralité une chaine du programme de bout en bout, dans notre cas, tester qu'un service web réagisse de la manière attendue.



Aperçu de la structure du dossier contenant le code métier de l'API

Cette image montre la structure interne du dossier `api`. Le dossier `bin` est en charge de contenir le programme qui démarre le serveur de l'API. Le dossier `config` contient les diverses configurations pour le bon fonctionnement du serveur, ainsi que des configurations spécifiques aux différents environnements. Le dossier `database` regroupe l'abstraction de la logique permettant d'accéder à la base de données, et dans ce projet il s'agit de l'ORM Sequelize. Le dossier `firebase` permet la même logique que le dossier `database`, mais son rôle est de fournir la passerelle avec l'ancien monde, pour garantir une migration en douceur. Le dossier `graphql` permet d'exposer les services web (ici on parle de « query »), pour les clients (applications mobile, applications web, etc.). Les autres dossiers contiennent du code permettant de réaliser la tuyauterie interne et mettre en relations les différents composants du projet et faire fonctionner l'écosystème.

Automatisation des tests

Durant le cycle de vie d'une application, on est souvent confronté à des régressions de comportement, c'est à dire un changement d'un comportement que l'on pensait encrer et définitif. Ce type de régression est souvent induit lorsqu'on développe ou met à jour une partie du code de l'application et que cette dernière ne contient pas de tests. En effet les tests vont venir figer, en quelque sorte, les comportements développés dans l'application. De cette manière on s'assure que toute modification future ne viendra pas perturber ces comportements, ou bien les tests seront là pour indiquer toute régression induite.

Cependant, chaque développeur n'aura pas nécessairement le réflexe d'exécuter tous les tests de l'application avant d'en déployer une nouvelle version. C'est ainsi qu'intervient l'automatisation des tests, grâce à ce que l'on appelle de l'intégration continue.

Pour automatiser nos tests, nous avons mis en place l'outil CircleCI, qui est un outil d'intégration continue. De cette manière, à chaque fois que nous publions du code sur GitHub (qui est une plateforme de partage de code, basé sur le système de version de code Git), une tâche est démarrée sur CircleCI, réalisant toutes les tâches définies dans un fichier de configuration. Parmi ces tâches, se trouvent une tâche exécutant tous les tests de l'application, assurant qu'aucune régression n'est causé par le nouveau code publié.

Intégration continue et déploiement

<image du circuit : développeur + GitHub + CircleCI + AWS>

Réalisation et gestion (12p)

Équipe (0.5p)

Gestion de projet (8p)

Parler des burndown/up chart + diagrammes

Méthode Kanban

Sprint planning (mettre le planning sous forme de diagramme de gantt)

Utilisation de Asana

Daily meeting

Rétrospective de sprint

Revue de planning

Suivi des KPI

Développement (1-2p)

<Mettre des screens de config, etc. (à voir), parler des tests et de la qualité du projet grâce aux tests>

Contraintes (1p)

Compréhension inter-équipes

Respect de la méthodologie agile

Délicatesse de la migration

Mise en production (1p)

Procédure

Contraintes

Évolution

Retours sur investissement (1p)

Augmentation des performances

Réduction des coûts

Transformation des processus (3p)

<A intégrer dans les chapitres des projets>

Nouvel environnement agile

Proposition et mise en place d'une gestion agile

Daily meeting

Sprints

Force de proposition pour des outils plus adaptés

Jira

Nouvel environnement technique

Solidité

Évolutivité

<Page de garde de l'annonce du projet>

Projet personnel

SnipHub

Problématique récurrente et naissance de l'idée

En tant que développeur, et comme la majorité dans mon cas, j'ai toujours connu un problème récurrent, le fait de devoir chercher de manière continue des bouts de code que j'ai déjà utilisé ou codé plusieurs fois auparavant.

Par exemple, j'ai régulièrement constaté que j'avais besoin de valider une adresse email ou un numéro de téléphone et qu'il m'était nécessaire à chaque fois d'effectuer une recherche sur internet, afin de pouvoir trouver ma réponse. Il apparaissait de manière évidente qu'il y avait un manque d'organisation à ce sujet et qu'une idée de projet devenait possible.

Souvent, les bouts de code réutilisables (snippets), sont éparpillés sur les sites web (StackOverflow par exemple), et sont parfois difficiles à trouver, il devient donc rapidement laborieux et chronophage de chercher et trouver le bon snippet.

Parti de ce constat, j'ai eu l'idée de formaliser et regrouper tous ces snippets, de manière organisée et universelle, afin que tout type de développeur puisse facilement retrouver un snippet sans devoir perdre du temps. J'ai fait pars du projet à un ami développeur, qui lui aussi avait effectué le même constat.

Nous avons donc décidé de se mettre en collaboration pour ce projet.

Étude du marché et analyse des concurrents

Avant de commencer à développer le projet, nous avons dû effectuer une étude du marché afin d'évaluer quels étaient les clients potentiels, ainsi que leurs attentes. Sans surprise, le résultat de l'étude était le suivant : Un grand nombre de développeurs passent leur temps à chercher des snippets sur des sites tels que GitHub, StackOverflow. Il parfois qu'ils perdent leur temps à chercher plusieurs fois le même snippet d'un projet à l'autre. On constate donc qu'un réel besoin est exprimé implicitement, et donc notre idée de projet est d'autant plus renforcée.

En quelques chiffres :

- **21 millions** de développeurs dans le monde
- **49 572 724** questions sur StackOverflow (communauté active)
- **1 243** questions taguées avec « code-snippets »
- **80 millions** de projets sur GitHub donc environ 40% sont des snippets, soit environ **32 millions** de snippets sur GitHub.

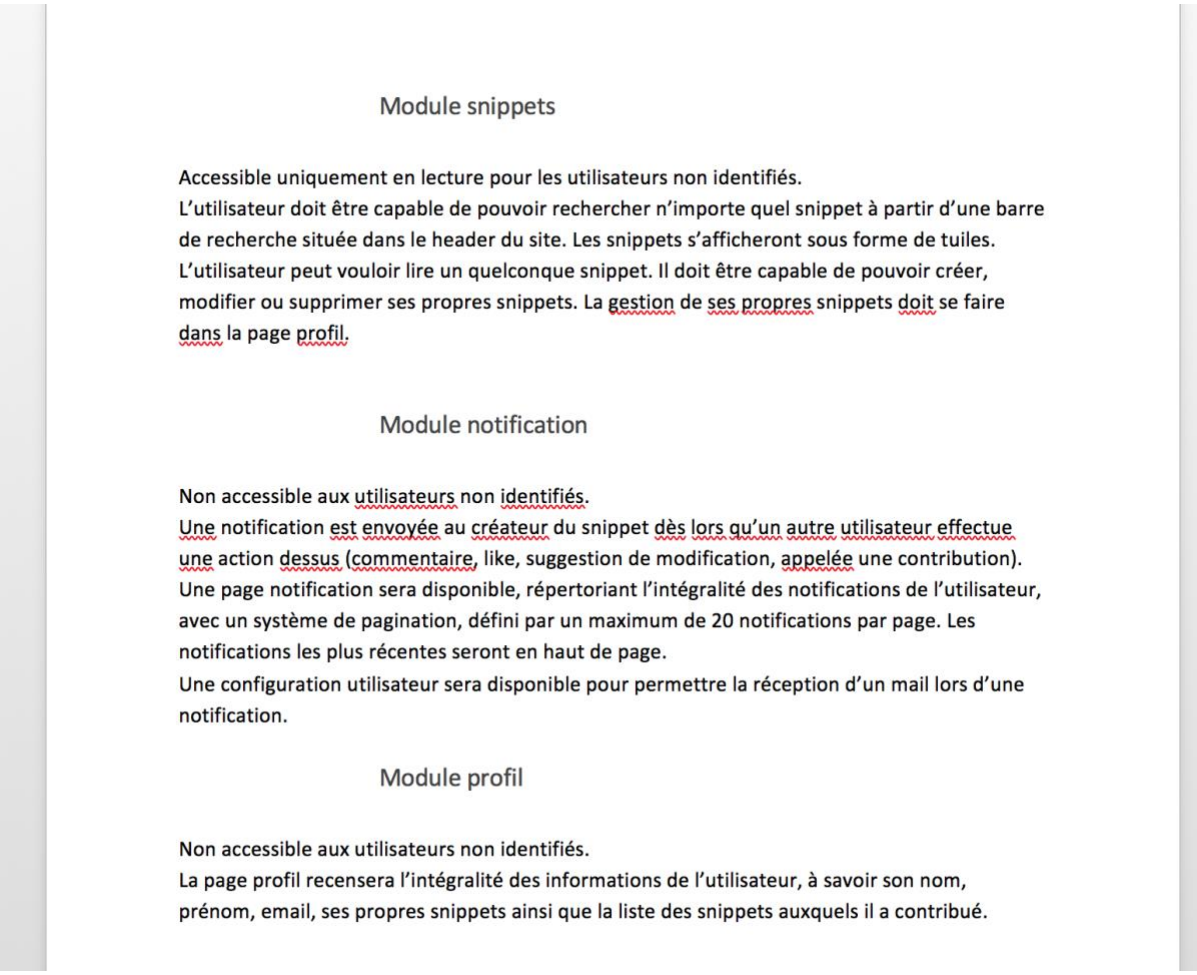
En ce qui concerne les concurrents, nous avons recensé les suivants :

- **GitHub :** permet de pouvoir publier des *gists* (même principe que les snippets). Peu organisés, difficile de rechercher un snippet précis dans un langage précis.
- **StackOverflow :** site de référence pour tout type de recherche pour un développeur, mais n'est pas conçu pour la recherche de snippets. Rechercher un snippets est donc souvent laborieux car il faut rechercher les bons termes par Google pour tomber sur un article du site référençant la réponse.
- **DevHints :** Le plus grand concurrent potentiel, mais leur recherche s'avère laborieuse en ne donnant pas toujours des résultats pertinent, l'ajout de snippets n'est pas simple (le point communautaire est mal élaboré), absence de toutes informations de qualité (les « j'aime », les commentaires).

Spécification, estimation, coûts

Cahier des charges

Durant la formation nous avons eu des cours de gestion de projet, et notamment savoir rédiger un cahier des charges. Ces notions nous ont permis de pouvoir rédiger un cahier des charges élaboré et pertinent, dans un but de mieux organiser nos idées en avance de phase.



Module snippets

Accessible uniquement en lecture pour les utilisateurs non identifiés.
L'utilisateur doit être capable de pouvoir rechercher n'importe quel snippet à partir d'une barre de recherche située dans le header du site. Les snippets s'afficheront sous forme de tuiles.
L'utilisateur peut vouloir lire un quelconque snippet. Il doit être capable de pouvoir créer, modifier ou supprimer ses propres snippets. La gestion de ses propres snippets doit se faire dans la page profil.

Module notification

Non accessible aux utilisateurs non identifiés.
Une notification est envoyée au créateur du snippet dès lors qu'un autre utilisateur effectue une action dessus (commentaire, like, suggestion de modification, appelée une contribution).
Une page notification sera disponible, répertoriant l'intégralité des notifications de l'utilisateur, avec un système de pagination, défini par un maximum de 20 notifications par page. Les notifications les plus récentes seront en haut de page.
Une configuration utilisateur sera disponible pour permettre la réception d'un mail lors d'une notification.

Module profil

Non accessible aux utilisateurs non identifiés.
La page profil recensera l'intégralité des informations de l'utilisateur, à savoir son nom, prénom, email, ses propres snippets ainsi que la liste des snippets auxquels il a contribué.

Aperçu du cahier des charges (voir Annexe 3)

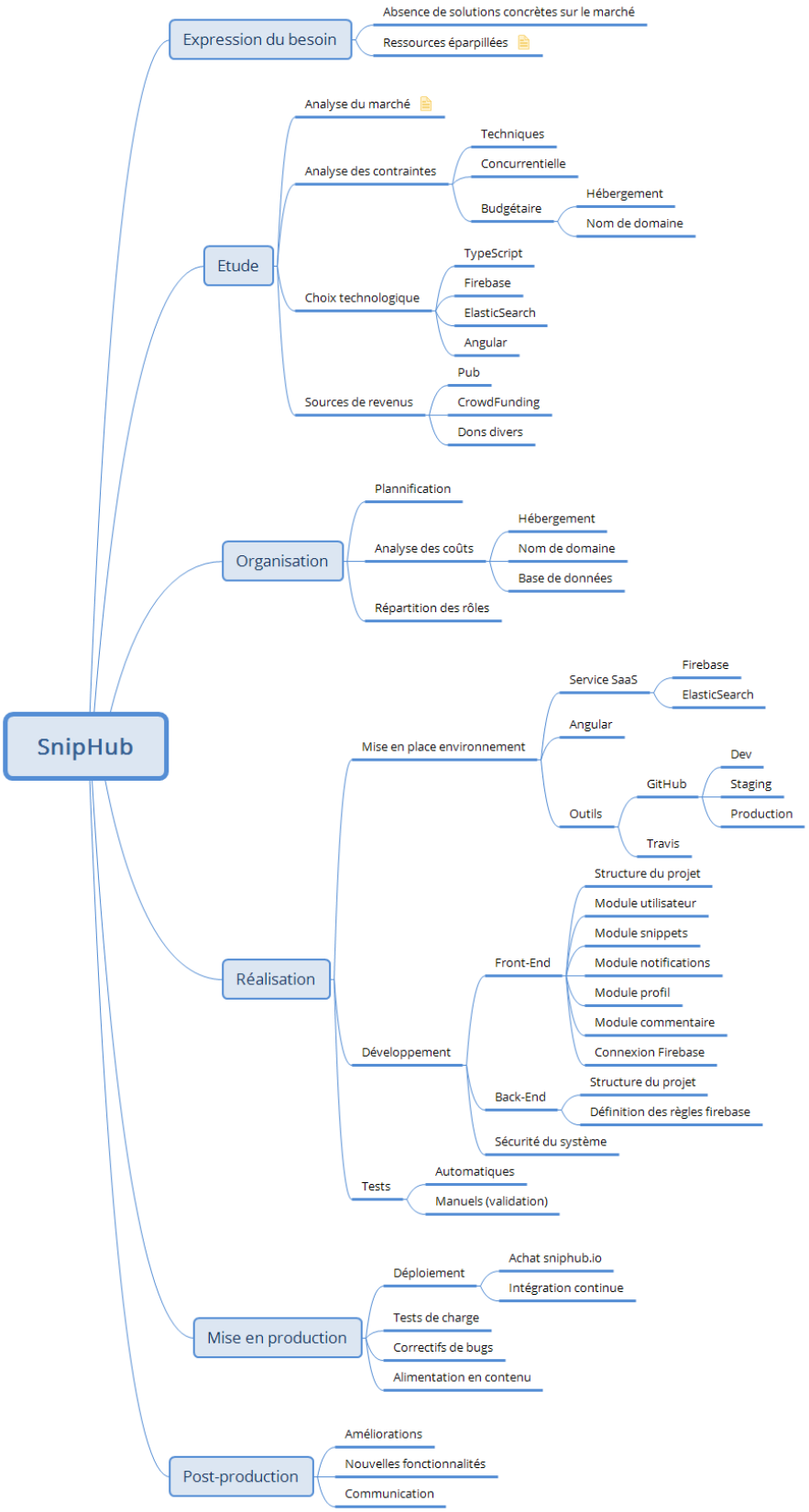
Work Breakdown Structure

Le Work Breakdown Structure (WBS) est un organigramme permettant d'ordonnancer les différentes tâches d'un projet, en les regroupant sous des jalons. Un jalon est une grande ligne, permettant de regrouper plusieurs tâches assignées à une même étape de la vie du projet.

Dans le cadre de la formation, nous avons pu élaborer le WBS de notre projet, permettant de voir plus clairement les différentes phases de ce dernier, mais aussi de pouvoir donner un premier appui pour le planning.

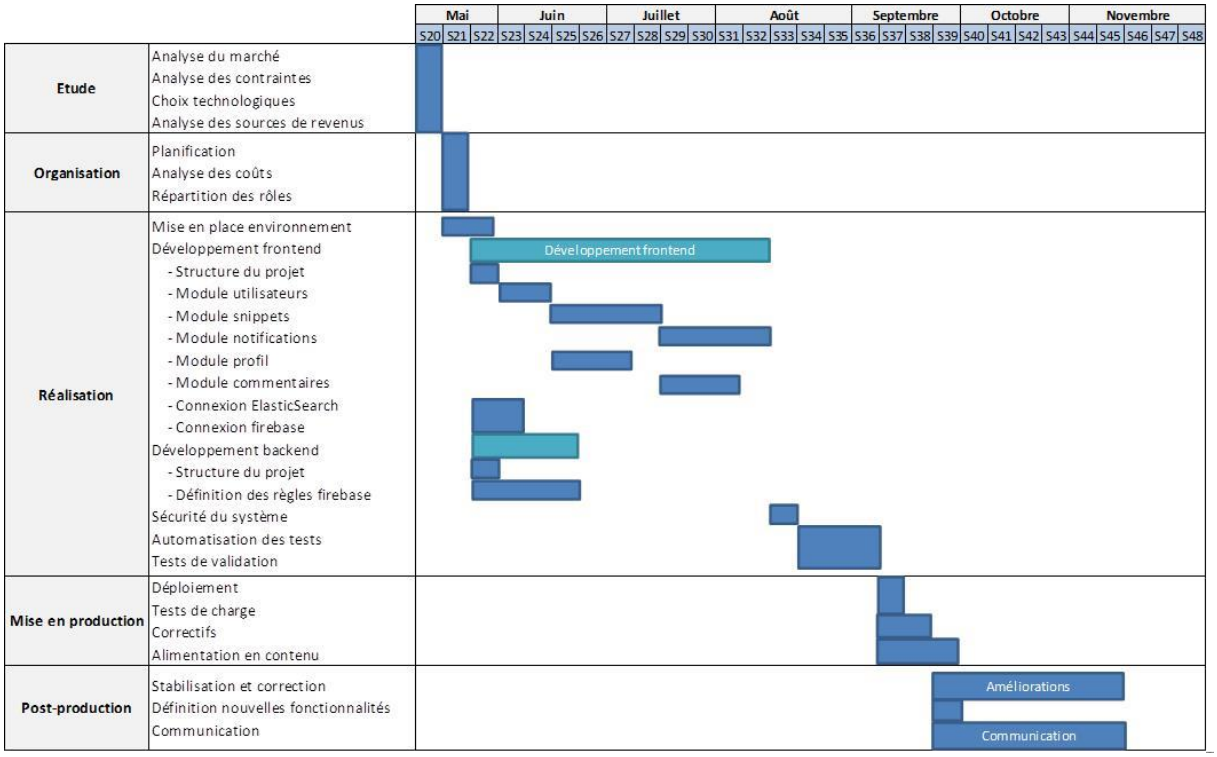
Le WBS du projet comporte six grandes phases :

- **Expression du besoin :** Prise de conscience à l'origine du projet
- **Étude :** Phase durant laquelle des recherches ont été effectuées, notamment sur la question des concurrents, l'analyse du marché et les différentes sources de revenus.
- **Organisation :** Mise en place de la gestion de projet, avec la planification, la répartition des rôles et les différentes analyses des coûts.
- **Réalisation :** Phase souvent complexe à estimer, puisqu'elle comprend la réalisation complète de l'application (mise en place de l'environnement, développements des fonctionnalités et développement des tests augmentant la maintenabilité et la qualité du code). Cette phase est souvent sous-estimée, ce qui a donc souvent pour conséquence un retard important sur la mise en production du projet.
- **Mise en production :** Phase critique du projet, puisque cette dernière rendra tout le travail effectué visible aux utilisateurs. Durant cette phase on retrouvera un déploiement suivi d'un test de charge pour s'assurer du volume d'utilisateur que peux supporter l'application, on trouvera également des correctifs de bugs (plus communément appelés hotfix). En dernière tâche de cette phase se trouve l'alimentation du site en contenu.
- **Post-production :** À la suite de certains retours utilisateurs, des améliorations et de nouvelles fonctionnalités pourront être apportées. Une communication importe est nécessaire pour faire connaître le projet.


SnipHub
Expression du besoin
Absence de solutions concrètes sur le marché
Ressources éparpillées
Etude
Analyse du marché
Analyse des contraintes
Techniques
Concurrentielle
Budgétaire
Hébergement
Nom de domaine
Choix technologique
TypeScript
Firebase
ElasticSearch
Angular
Sources de revenus
Pub
CrowdFunding
Dons divers
Organisation
Plannification
Analyse des coûts
Hébergement
Nom de domaine
Base de données
Répartition des rôles
Réalisation
Mise en place environnement
Service SaaS
Firebase
ElasticSearch
Angular
Outils
GitHub
Dev
Staging
Production
Travis
Développement
Front-End
Structure du projet
Module utilisateur
Module snippets
Module notifications
Module profil
Module commentaire
Connexion Firebase
Back-End
Structure du projet
Définition des règles firebase
Sécurité du système
Tests
Automatiques
Manuels (validation)
Mise en production
Déploiement
Achat sniphub.io
Intégration continue
Tests de charge
Correctifs de bugs
Alimentation en contenu
Post-production
Améliorations
Nouvelles fonctionnalités
Communication

Work Breakdown Structure de SnipHub

Planning



Planning prévisionnel de SnipHub

Réalisation et contraintes

Mise en place d'une intégration continue

Automatisation des tests

Protection des données personnelles

Mise en production

Maintenance et évolutions

Premier bilan

Conclusion

Annexes