



LIBERTY RIDER

DOSSIER DE VALIDATION

CHEF DE PROJETS INFORMATIQUE

Nom Prénom	SERGENT Julien
Nom(s) Prénom(s) du ou des tuteurs	D'Allens Martin
Acronyme de la certification IPI visée	CDPI
Niveau visé	RNCPI
Date de la soutenance	17/07/2018
Lieu de la soutenance	IPI Toulouse, 186 route de Grenade, 31700 Blagnac

Remerciements

Je tiens à remercier Liberty Rider, et plus particulièrement Martin d'Allens, pour m'avoir beaucoup appris, tant sur le plan humain que technique. Cette alternance m'a permis de découvrir la vie en startup ainsi que savoir prendre des responsabilités.

Martin d'Allens a su se montrer très pédagogue et patient, de plus, sa passion pour l'informatique m'a beaucoup apporté. J'ai pu développer une grande quantité de compétences dont je me sers dorénavant au quotidien. Je remercie aussi les autres membres de l'équipe Liberty Rider, ainsi que Emmanuel Petit, de m'avoir permis de réaliser mon alternance au sein de Liberty Rider ainsi que son bon déroulement.

Je remercie également l'IPI pour leur accueil durant ces deux années, et de leur programme de Mastère, m'ayant permis d'avoir une meilleure vision de l'entrepreneuriat et de la gestion de projet. Les formateurs sont compétents et passionnés, ce qui rend l'apprentissage plus agréable, de plus, les retours d'expériences qu'ils nous fournissent nous permettent d'avoir une vraie vision du monde professionnel.



Index des 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

<Refaire avec les nouvelles compétences>



Grille de compétences

<Mettre à jour>

	Projet		
Compétence	Nouvelle version de détection d'accident	Migration de la base de données	SnipHub
Gérer les processus et la qualité			?
Spécifications, Environnement technique, Réalisation et gestion			
Pratiquer des audits en respectant une méthode		✓ (RGPD)	
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	✓ (Architecture)	✓	✓
Concevoir et proposer des solutions innovantes de reconstruction des processus	✓	✓	✓
Autres compétences à décrire :			
Concevoir et appliquer un processus d'automatisation de tâches	✓	✓	✓
Utilisation d'un cadre DevOps pour améliorer la communication et le travail inter-équipes	✓ (Environnement technique)	✓ (Environnement technique)	
Savoir améliorer la qualité du projet	?	?	?
Gérer les ressources du projet	Réalisation et gestion	Préparatifs, Réalisation et gestion	
Prévoir des ressources humaines, recruter des collaborateurs	✓ (Equipe)	✓ (Equipe, Recrutement de compétences)	✓
Manager une équipe, évaluer les collaborateurs		✓ (Gestion de projet)	✓
Prévoir et trouver les moyens logistiques nécessaires au projet	✓ (Choix techniques)	✓ (Logistique)	✓
Maîtriser les aspects juridiques des contrats à passer			✓
Gérer le budget des projets		Estimation des charges et du budget	Spécification, suivi du budget
Élaborer, faire valider un budget			✓
Contrôler les tableaux de bord de suivi budgétaire			✓
Savoir rendre compte de l'état d'avancement du budget			✓
Communiquer	Travail R&D, Spécifications, Réalisation et gestion	Préparatifs, Spécifications	?
Élaborer un cahier des charges ou y répondre	✓	✓	✓
Organiser des réunions, capacité à négocier	✓	✓	
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	Réalisation et gestion	Réalisation et gestion	
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	Environnement technique	Environnement technique	?
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 — Équivalent à PDG

R&D : Recherche et Développement

B2B : Business to Business — Activités commerciales et marketing entre entreprises

B2C : Business to client — Activités commerciales et marketing entre entreprises et particuliers

CPO : Chief Product Owner — Accompagne, structure et accélère la croissance économique de l'entreprise

CMO : Chief Marketing Officer — Responsable de la direction marketing d'une entreprise

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 satisfassent au mieux la demande de leurs consommateurs, avec un investissement initial minimal.

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 rencontrés.

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.

POC : Proof Of Concept est une réalisation expérimentale concrète et préliminaire, courte ou incomplète, illustrant une certaine méthode ou idée afin d'en démontrer la faisabilité.

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

FRAMEWORK: Ensemble de composants logiciels permettant de guider la manière de développer un projet.

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 continu

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



Table des matières

1. Introduction	9
2. Liberty Rider	10
2.1. L'histoire	10
2.2. La startup	11
2.2.1. Effectif et organisation.....	11
2.2.2. Hiérarchie.....	12
2.3. L'application	13
2.4. Vision, mission et ambition	14
2.5. Clients et partenaires.....	14
2.6. Contexte et problématiques	15
2.6.1. Gestion de projet.....	15
2.6.2. Environnement technique	16
3. Projet : Nouvelle version de détection d'accidents	17
3.1. Contexte et objectifs.....	17
3.2. Travail de recherche en amont	18
3.3. Spécifications	18
3.3.1. Rédaction	18
3.3.2. Architecture	20
3.4. Environnement technique	21
3.5. Réalisation et gestion	22
3.5.1. Équipe.....	22
3.5.2. Développement	22
3.5.3. Revue de planning et des spécifications.....	23
3.5.4. Contraintes	23
3.6. Mise en production.....	24
3.7. Analyse des erreurs	25
4. Projet : Migration et nouvelles fonctionnalités	27



4.1. Contexte et objectifs.....	27
4.2. Préparations.....	28
4.2.1. Cahier des charges	28
4.2.2. Recrutement de compétences	29
4.2.3. Roadmap	29
4.3. Mise en place de l'environnement.....	30
4.3.1. Environnement technique	30
4.3.2. Structure du projet	31
4.3.3. Automatisation des tests	32
4.3.4. Intégration continue et déploiement	33
4.4. Réalisation et gestion	34
4.4.1. Équipe	34
4.4.2. Gestion de projet	34
4.4.3. Développement.....	40
4.4.4. Contraintes	45
4.5. Mise en production.....	46
4.6. Retours sur investissement.....	46
4.6.1. Augmentation des performances	46
4.6.2. Réduction des coûts	46
5. Transformation des processus.....	47
5.1. Nouvel environnement agile.....	47
5.1.1. Proposition et mise en place d'une gestion agile.....	47
5.1.2. Force de proposition pour des outils plus adaptés	48
5.2. Nouvel environnement technique.....	48
5.3. Nouvelle politique de confidentialité	49
6. Projet personnel : SnipHub	50
6.1. Problématique récurrente et naissance de l'idée	50
6.2. Étude du marché et analyse des concurrents.....	51
6.3. Spécification, estimation, coûts.....	52
6.3.1. Cahier des charges.....	52



6.3.2. Work Breakdown Structure.....	53
6.3.3. Planning	55
6.3.4. Analyse des coûts	56
6.4. Réalisation	57
6.4.1. Mise en place de l'environnement	57
6.4.2. Développement	58
6.4.3. Intégration continue et automatisation des tests.....	58
6.5. Aspects juridiques	59
6.6. Mise en production.....	60
6.7. Suivi du budget.....	60
6.8. Retour d'expérience	62
7. Conclusion.....	63
8. Annexes.....	64

1. 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.



2. Liberty Rider



 LIBERTY RIDER

2.1. L'histoire

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

Liberty Rider est donc né, et 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 certaines difficultés à trouver des investisseurs. Une autre problématique apparaît : définir la fonctionnalité primaire de l'application, selon une démarche de lean startup. Cette démarche est 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 Toulousain. C'est à partir de ce moment-là que tout prend forme, les premiers stagiaires et employés ont été recrutés, les premiers rôles ont été attribués.

Ce sont les prémisses de la jeune startup, comme on a la connaît actuellement.

2.2. La startup

2.2.1. Effectif et organisation

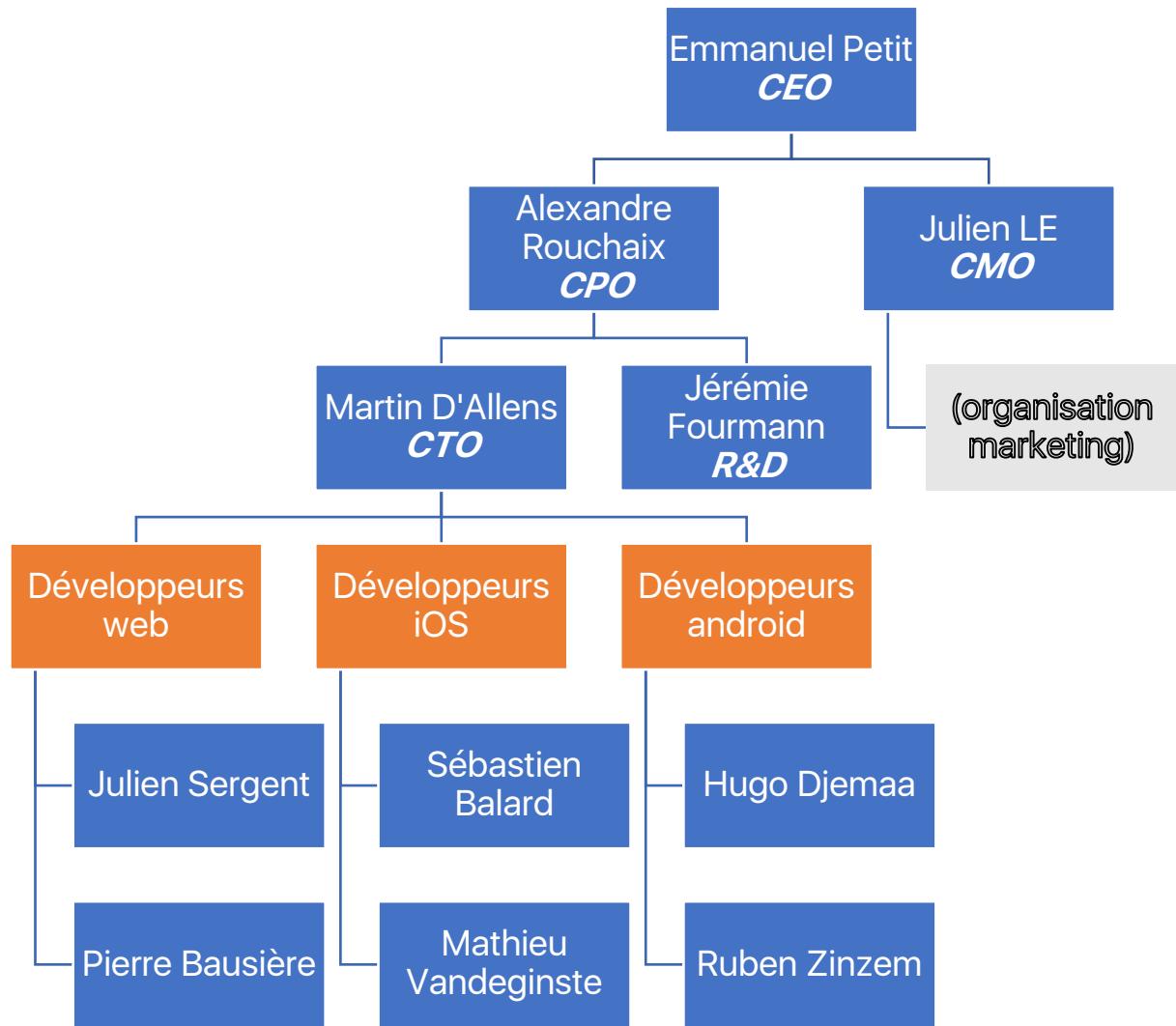
Liberty Rider est actuellement composé de dix-sept personnes ; dont le CEO, trois développeurs web, quatre développeurs mobiles, un CPO (Chief Product Owner), un responsable R&D (Recherche & Développement), quatre personnes chargées de la partie B2C (Business To Client), deux personnes chargées de la 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 : parler du CA, du nombre d'effectif, les coûts des salaires>

2.2.2. Hiérarchie





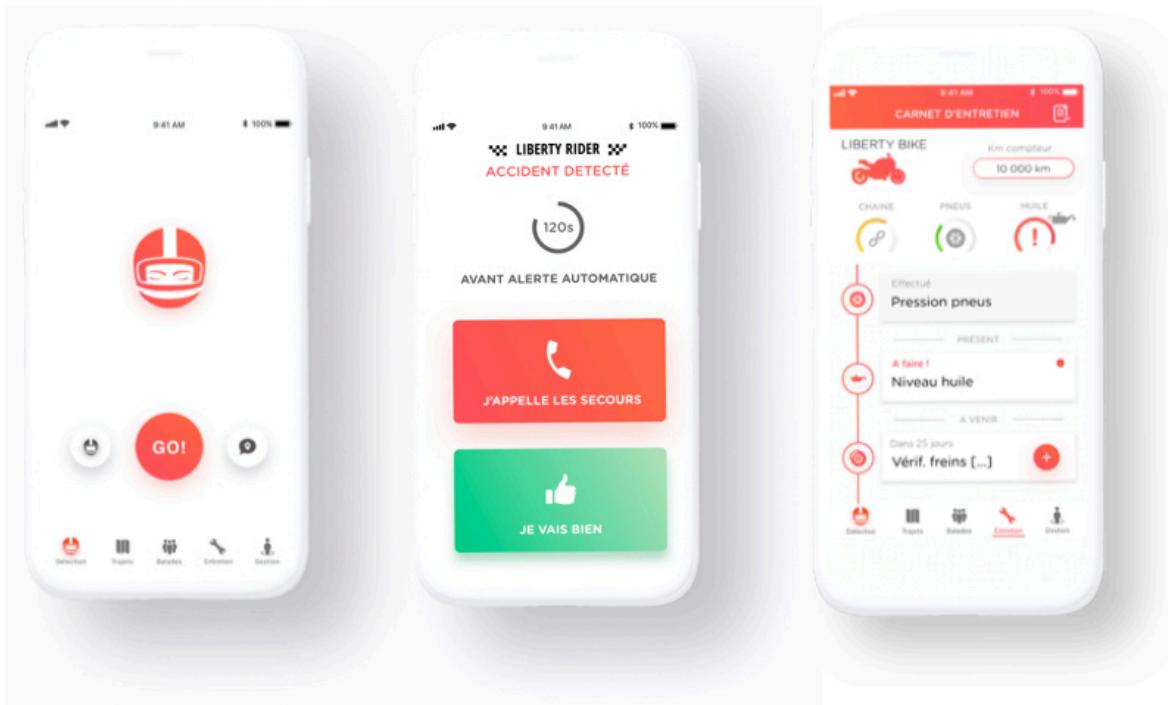
2.3. L'application

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 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. Par cette action, l'entreprise permet de porter assistance au motard dans les plus brefs délais.

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, comme être alerté quand on doit réaliser sa vidange, ou bien tout autre type de contrôle. Elle permet également d'organiser des balades avec plusieurs motards, de façon à 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.).

2.4. 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 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 cette dernière.

Aujourd'hui, l'application ne fonctionne qu'exclusivement en France pour des raisons de proximité avec les secours, et d'échanges avec les utilisateurs. Il est aussi envisagé, sur du moyen terme, une expansion en Europe, grâce à un nouveau partenariat avec IMA (Inter Mutuelles Assistance). IMA permettra à Liberty Rider de pouvoir agir dans tous les pays européens.

2.5. 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 accroître ses effectifs et assurer l'augmentation de charge des nouveaux utilisateurs. Ce nouvel écosystème s'accompagne d'une croissance 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.

2.6. Contexte et problématiques

2.6.1. Gestion de projet

L'organisation au sein d'une startup passe par différentes étapes. Les débuts sont plutôt désordonnés, mais suivis de différentes phases d'améliorations qui aboutissent à une meilleure mise en place des procédures.

A mes débuts dans la startup, j'ai commencé par le projet *Gamification*. Le but du projet est de fournir une boutique dans l'application, où le motard peut y dépenser ses flooz gagnés durant ses sessions. Ce projet consiste à rendre l'application plus ludique et favoriser la rétention des utilisateurs.

Liberty Rider ne possédait pas la capacité financière à employer des salariés pour chacune de ses missions. Certains employés se sont vus attribué un rôle dont les compétences requises n'étaient pas nécessairement les leurs. Par conséquent, un manque d'organisation du projet a entraîné un besoin récurrent de solliciter le chef de projet pour mieux connaître le cadre des fonctionnalités à développer. Cette désorganisation a engendré un retard sur les dates de livraison du projet, et parfois des non conformités aux spécifications demandées.

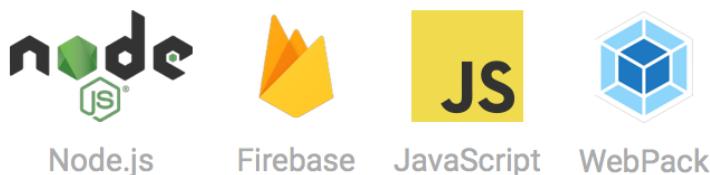
Les prises de décisions s'effectuées entre les équipes des développeurs, le plus souvent par oral. Certains comptes rendus étaient déposés sur Slack.

Aux débuts de mon alternance chez Liberty Rider, l'organisation était quelque peu empirique, mais très vite des améliorations ont été mises en place.

2.6.2. Environnement technique

Les solutions techniques adoptées chez Liberty Rider sont les suivantes :

WEB :



MOBILE :



DEVOPS :



ARCHITECTURE :



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

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

3. Projet : Nouvelle version de détection d'accidents

3.1. Contexte et objectifs

Liberty Rider est avant tout une application de détection d'accidents, 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 commencé chez Liberty Rider sur le projet de la nouvelle version de détection d'accidents. Ce projet a pour but d'améliorer les contraintes et problématiques présentes, à savoir de fausses, voire aucune, détections, alors qu'il y a accident. Raison supplémentaire, la startup était en passe de concrétiser un partenariat avec IMA, qui aurait pour charge d'intervenir sur le lieu de l'accident.

Les nouveaux enjeux de l'entreprise sont importants pour cette dernière, puisqu'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 d'en obtenir davantage par la suite.

Bien évidemment, d'autres objectifs étaient attendus à l'issu de ce projet :

- Une nouvelle architecture côté serveur, pour rendre hautement disponible l'API d'accidents
- Amorcer la migration et l'insertion d'un nouveau backend, afin de commencer à se détacher au fur et à mesure de Firebase

L'API est un des points clés de l'application et ne peut pas se permettre d'être indisponible pour une raison quelconque, sous peine de rendre l'application privée de sa fonction première.

Il est important de garder à l'esprit que Firebase coûtait 700€/mois, ce qui n'est pas négligeable quand on est une startup. Son manque de souplesse et de prise de contrôle provoquait des problèmes de performance sur tout le réseau de l'application. Certaines des requêtes émises par l'application (par exemple, lorsqu'un utilisateur visionne tous ses anciens trajets) entraînent un ralentissement de Firebase, dû au traitement de beaucoup données. Cette problématique avait pour conséquence de rendre l'application indisponible pendant dix minutes, ce qui est un problème majeur.



Liberty Rider étant situé dans les locaux de l'incubateur AtHome, au milieu de beaucoup d'autres startup, nous étions régulièrement dérangés dans milieu du travail. Afin de pouvoir mener à bien ce projet et de rester concentré, nous avons décidé de partir une semaine, dans une maison isolée à Pau, appelée « dev week ». Cette semaine avait pour but de dégrossir le projet, produire les spécifications et commencer à les implémenter pour en obtenir une première version testable.

3.2. Travail de recherche en amont

Comme expliqué dans le chapitre précédent, la détection d'accidents est le cœur de l'application, et ce n'est pas l'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 les vrais accidents.

Le problème est que de tels algorithmes requièrent un important travail préparatoire. Il était donc nécessaire de réaliser des recherches, préparer tous les aspects des nouveaux algorithmes, anticiper les contraintes, et rédiger les documents techniques pour en simplifier l'implémentation par 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 réalisées à l'aide du langage de programmation Python.

3.3. Spécifications

3.3.1. Rédaction

Avant de commencer le projet et afin de mieux le réaliser, nous avons décidé de rédiger des spécifications pour mieux prévoir les aspects de ce dernier. Notamment les éventuels points bloquants, le temps que nous allions y consacrer, et des documents à rédiger en anglais et français (anglais pour la partie technique, axés développeurs, et français pour la partie commerciale). Ces documents permettront de recenser la manière



dont seront menés les travaux, les temps attribués aux différentes tâches, et les différents comportements des nouveaux algorithmes.

Pour mieux réaliser ces spécifications, nous avons effectué plusieurs réunions. Chacune avait pour but les points suivants :

- Mieux prévoir toutes les frontières de la nouvelle version de détection d'accidents (concerne tous les éléments qui allaient être migrés, modifiés et ajoutés)
- Se mettre d'accord sur les aspects techniques, et comment nous allions procéder

J'ai animé des réunions de planifications et contribué à rédiger les spécifications du cahier des charges. Ce qui m'a permis de mettre en place une cohésion de groupe, de manière à éviter les conflits et afin d'optimiser le 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 futilles 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
 - ↳ string

Aperçu des spécifications de l'API du serveur en charge de gérer les accidents (voir Annexe 1)

3.3.2. Architecture

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

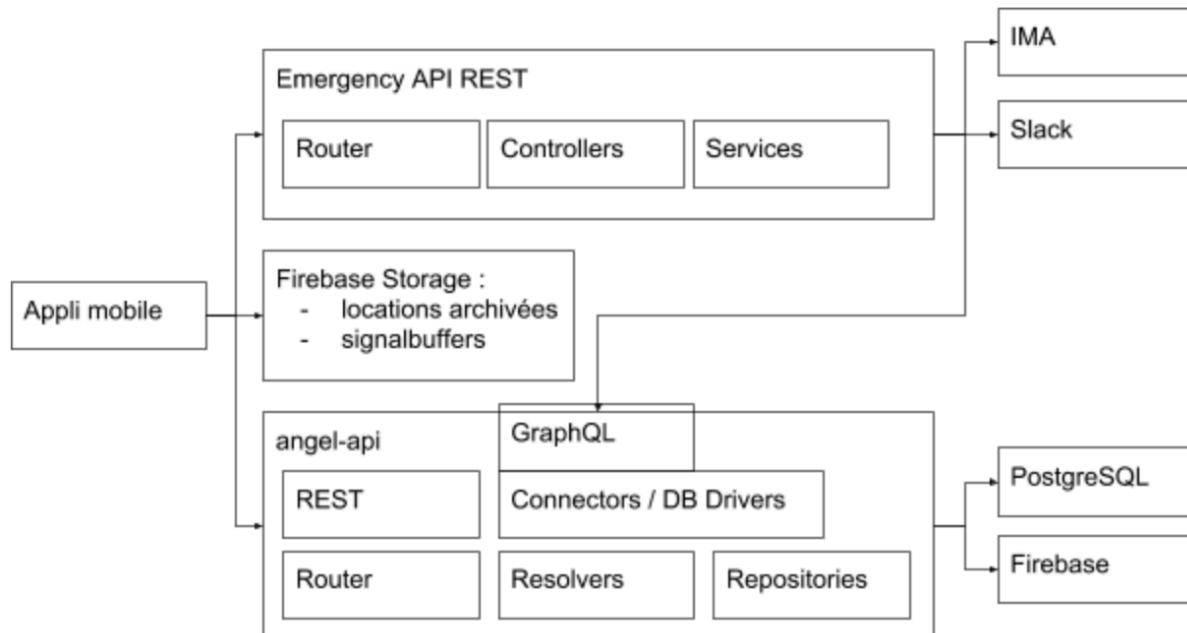


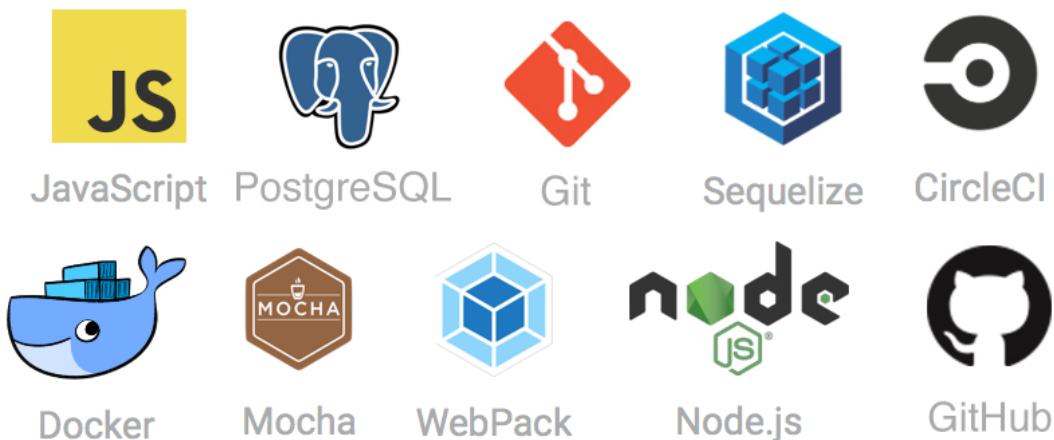
Schéma de l'architecture des différents services web de Liberty Rider

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

Dans le précédent schéma on représente les clients qui utilisent les services web par le 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 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 détail dans la présentation du projet **Migration et nouvelles fonctionnalités**.

3.4. 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 choisis de manière élaborée. 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).

GitHub a été utilisé comme support de gestion de version de code, pour sa popularité et sa simplicité d'utilisation.

3.5. Réalisation et gestion

3.5.1. 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 les nouveaux algorithmes de détection d'accidents.

Les équipes projet étaient donc les suivantes :

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

3.5.2. Développement

Les spécifications prêtes, le temps était venu de commencer à développer le projet. A la suite des premiers jours de développement, j'étais plus sur un terrain d'écoute et d'observation que d'autonomie. En effet il s'agissait d'un de mes premiers projets avec Liberty Rider, et je n'avais pas encore les connaissances métiers de leur système. Il m'était délicat de prendre des décisions seul face aux questions et demandes des autres équipes. Pour mieux appréhender un problème ou un doute rencontré, je m'adressais systématiquement à Martin. De cette manière j'ai pu rapidement évoluer sur l'écosystème de l'entreprise afin de pouvoir travailler en pleine autonomie.

Le développement du projet s'est bien déroulé. J'ai pu mettre en pratique mes connaissances techniques pour mettre en place l'architecture du projet qui allait accueillir les services web. J'ai pu également accroître 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. Je trouve qu'elle occupe une place très importante dans l'organisation du travail au sein de l'équipe, et donne un impact positif sur la réalisation du projet.

<parler des web services, des routes, config, REST, etc.>

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

3.5.3. 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 allions ajouter et sous quel format, afin d'en débattre et se mettre d'accord entre les différentes équipes. J'ai pu rédiger des spécifications en anglais, 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 pour d'améliorer la qualité du projet.

3.5.4. Contraintes

Cependant, la réalisation de ce projet ne s'est pas déroulée sans accroc. 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 initialement prévue, des stratégies de suivi et d'amélioration de qualité ont été mises en place, mais sans structure réelle, ni responsable de leur gestion.

Sans planning, ni de sprint ni de tâche, 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 quelques 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 des différentes phases (sprint planning, rétrospectives, sprint review). Dans le cadre de la dev week, il s'agissait d'avoir une version bêta en seulement une semaine.

D'autres contraintes sont venues perturber l'équilibre du projet, et notamment le manque de spécifications claires et constructives. Cela a mené à des dysfonctionnements au sein des équipes mobiles (équipe iOS et équipe Android), qui n'avançaient pas à vitesse égale. Il arrivait 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.

3.6. 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 fonctionne en local, mais quand on déploie le projet en production des dysfonctionnements sont constatés.



Pour les besoins du projet, nous avons dû prévoir deux serveurs sur AWS (Amazon Web Services). Un serveur pour gérer uniquement la chaîne de détection d'accidents, ce serveur est isolé afin d'éviter tout dysfonctionnement de la chaîne d'alerte si une erreur survient. Un autre serveur est nécessaire 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, accompagnées d'un équilibrEUR de charge permettant de répartir les charges serveurs. Ces éléments sont alloués dans le but d'assurer une disponibilité continue du serveur en charge de la chaîne d'alerte pour les accidents.

Ces machines virtuelles sont rassemblées dans des groupes nommés « cluster ». Chaque groupe correspond à un environnement qui leur est dédié, nous avons au total trois environnements différents : développement, validation et production. 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.

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

Une fois l'architecture réalisée, j'ai dû mettre en place une 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 escompté.

3.7. 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, vitesse 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.**

4. Projet : Migration et nouvelles fonctionnalités

4.1. Contexte et objectifs

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 décisions. Durant ces deux premières années, Liberty Rider a accumulé quelques dettes techniques, c'est à dire pas forcément les meilleurs choix techniques, et avec la croissance des effectifs il était important de corriger ces défauts pour repartir sur de meilleures bases. Il s'agit du premier objectif de ce projet.

À la suite des levées de fonds, des résultats sont attendus par les investisseurs. Il est 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 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

A l'issue de ce projet, Liberty Rider doit devenir autonome sur le plan financier, et arriver à produire un chiffre d'affaire conséquent. Cette condition a été fixée avec les investisseurs.

Une autre attente à l'issue de ce projet est un nouveau socle technique et marketing. Le socle technique a pour objectif de faciliter l'arrivée de futurs développeurs dans l'équipe. Quant au socle marketing, il a pour objectif d'augmenter la remontée de données des utilisateurs pour le pôle marketing, mais aussi de fournir des outils communs pour traiter les données, entre développeurs et commerciaux.



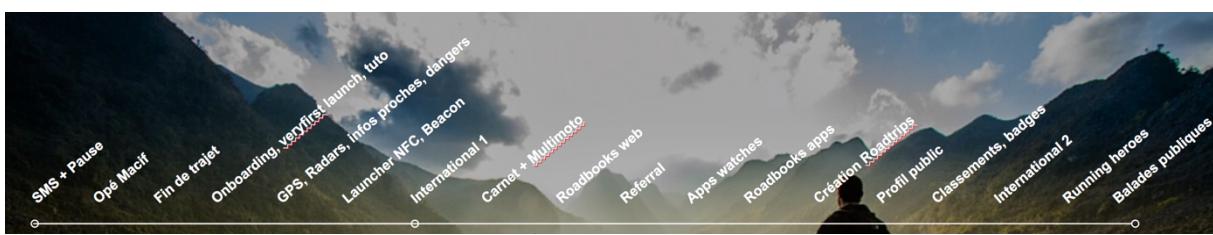
4.2.2. Recrutement de compétences

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).

4.2.3. 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 accroître la communauté de Liberty Rider pour gagner en notoriété.

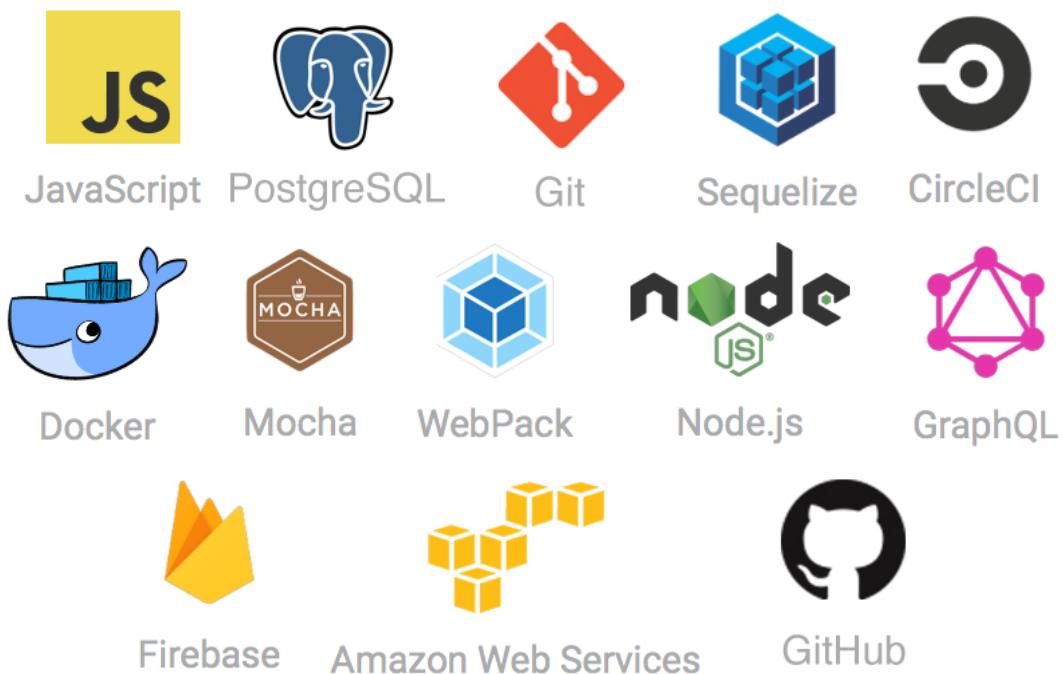


Roadmap de Liberty Rider

4.3. Mise en place de l'environnement

4.3.1. 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.

4.3.2. Structure du projet

```

    ▲ angel-api
      ▷ api
      ▷ system-tests
  
```

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 chaîne du programme de bout en bout, dans notre cas, tester qu'un service web réagisse de la manière attendue.

```

    ▲ angel-api
      ▲ api
        ▷ bin
        ▷ config
        ▷ database
        ▷ firebase
        ▷ graphql
        ▷ lib
        ▷ node_modules
        ▷ public
        ▷ repositories
        ▷ rest
        ▷ services
        ▷ test
        ▷ utils
  
```

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.

4.3.3. 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.

4.3.4. Intégration continue et déploiement

L'automatisation des tests s'inscrit dans le cadre de l'intégration continue. En effet, l'intégration continue est un ensemble de pratiques visant à vérifier que le nouveau code n'engendre pas de régression. Cette vérification est effectuée grâce aux tests.

Pour la mise en place de l'automatisation des tests avec CircleCI, nous avons dû configurer une intégration continue avec CircleCI. Cette intégration continue permettait de ne pas engendrer de régression, mais aussi effectuer plusieurs tâches, à savoir :

- Déployer le nouveau code sur le serveur
- Réaliser une compilation du schéma GraphQL pour les développeurs mobiles

Cet ensemble de pratiques nous permet de ne pas s'occuper, à chaque modification du code, du déploiement pour le mettre en production, ce qui est un gain de temps considérable.

Dans le cadre de la mise en place de l'intégration continue, j'ai été en charge de réaliser le fichier de configuration pour CircleCI, servant à définir les tâches à exécuter, ainsi que l'ordre dans lequel elles doivent l'être (la configuration se fait avec le langage **YAML**).



4.4. Réalisation et gestion

4.4.1. Équipe

Ce projet est décisif pour la survie de Liberty Rider. L'issue de ce dernier doit montrer des résultats pour que la startup puisse faire de nouveau une levée de fond et espérer vivre avec propre chiffre d'affaires.

Au vu de l'importance du projet, du délai qu'il est nécessaire pour le réaliser ainsi que du nombre de tâches à effectuer, la grande majorité des équipes ont dû être mobilisées.

Les équipes projet sont les suivantes :

- **Web** — Martin, Pierre et moi-même
- **Mobile** — Mathieu, Sébastien, Ruben et Hugo
- **Product Owner (dont CPO)** — Alfred et Alexandre

4.4.2. Gestion de projet

4.4.2.1. Méthode Kanban et utilisation d'Asana

À la suite de l'arrivée du CPO, la gestion de projet s'est vue drastiquement modifiée. Nous sommes entrés dans une nouvelle méthode de travail avec l'utilisation d'une méthodologie agile. Comme Liberty Rider est éditrice de son propre logiciel, la méthode Kanban se révèle être le meilleur candidat pour gérer le flux continu des tâches.

Pour la mise en place de la méthode Kanban, nous avons gardé l'outil Asana, qui était déjà utilisé. Nous avons cependant modifié le flux de travail pour mieux correspondre au flux de travail kanban.

Le nouveau flux de travail nous a permis de pouvoir découper le travail à réaliser en plusieurs sprint. Chaque sprint est composé de plusieurs user story. Une user story (récit utilisateur, en français) permet de décrire, avec des mots de tous les jours, une fonctionnalité à développer. Ces user story sont ensuite découpées en plusieurs tâches. Ces tâches sont les actions finales à réaliser pour compléter l'user story. Une fois toutes les user story terminées, on peut considérer un sprint comme terminé. Cependant, pour mieux cadrer les délais de réalisation, un sprint est encadré par une durée. Nous avons décidé de définir la durée d'un sprint à deux semaines. De cette manière, le CPO peut découper tous les travaux du projet en sprint, et en déduire le temps nécessaire pour tout réaliser.



Ayant déjà pu effectuer des projets sous une méthodologie agile au sein de Capgemini, j'ai été force de proposition pour la mise en place d'un tel flux de travail. Je n'avais pas réellement d'affinité à utiliser Asana, que je trouve peu intuitif pour une gestion agile, contrairement à l'outil Jira, mais je me suis rapidement adapté à cet outil et ai pu faire des propositions d'améliorations.

Notre flux de travail kanban est le suivant :

1. La tâche est dans son statut initial « A faire »
2. Lorsque quelqu'un commence à travailler sur une tâche, son statut passe à « En cours »
3. Une fois la tâche terminée, on passe son statut à « Revue de code » (la tâche doit être validée par les autres développeurs de la même équipe)
4. Lorsque la tâche est validée par les développeurs, son statut devient « Validation » (elle doit être validés par le CPO)
5. Une fois validée par le CPO, son statut devient « Prochain déploiement » (elle sera déployée à la fin du sprint)
6. A la fin du sprint, toutes les tâches avec le statut « Prochain déploiement » sont déployer en production, leur statut devient « Production », la tâche est terminée

Task Description	Sprint	Developer	Status
backend : terminer la PR stoppedRideConnection et ne pas retourner hidden=true	Sprint 3	back	Prod.
backend : préparer la liste des rides sans pagination (en plus de la version paginée)	Sprint 5	back	Prod.
backend : query pour lister les rides favorites	Sprint 5	back	Prod.
backend : stopper les rides au moment du Get si elles ont plus de 24h	Sprint 5	back	Next...
backend: retirer le field "isWithPassenger" des mutations update (bouton plus accessible depuis l'UI)	Sprint 5	back	Prod.
backend : écrire et tester en staging un script d'importation des anciennes Rides	Sprint 5	back	Next...
backend : résoudre 504 quand on fait endedRides sur libertybidon	Sprint 5	back	Valid.
backend : remplir la colonne sharingToken	Sprint 5	back	Next...
backend : rendre non optionnels tous les attributs (ou a minima clientTime) de startEventTime: Client	Sprint 5	back	Block...
backend : rendre non optionnels les attributs isFavorite, distance et duration dans Ride	Sprint 5	back	Block...
backend : déterminer précisément la cause des 503	Sprint 5	back	Valid.
backend : selfFavoriteRide doit renvoyer User		back	Next...
android : remplacer le get Firebase par un get GraphQL	Sprint 5	android	Valid.
android: cacher la liste	Sprint 5	android	Valid.
android: get ride by id	Sprint 5	android	Valid.
android : s'assurer qu'on a plus aucun setSynced() pour Firebase	Sprint 5	android	Next...
android : s'assurer qu'on touche plus Firebase (lecture/écriture)		android	Valid.
ios : remplacer le getRidesDis par la query GraphQL correspondante sans pagination	Sprint 5	iOS	Block...
ios : désactiver le getRide Firebase	Sprint 5	iOS	Block...
ios : dégager le cache existant pour utiliser le cache apollo	Sprint 5	iOS	Block...
ios : vérifier que le simplified locations est bien encore enregistré dans Firebase si il manque	Sprint 5	iOS	Block...
ios : annuler le job de renommage auto de ride quand un user essaie de renommer à la mano un ride (Sprint 5	iOS	Block...
ios : supprimer la fonctionnalité d'export, penser au wording sur la popup premium	Sprint 5	iOS	Block...
ios : utiliser AlamofireImage comme cache des images preview des rides	Sprint 5	iOS	Block...
ios : fix l'inversion des numéros de version		iOS	Prod...
back : fix problèmes de mémoire		back	Todo

Aperçu de l'outil Asana, présentant des user stories et des tâches d'un sprint en cours du projet

4.4.2.2. Périmètre et planification du sprint

Un projet, réalisé avec une méthode agile, est composé de plusieurs sprint. On peut imaginer chaque sprint par une itération, ou chaque itération va apporter de la valeur ajoutée au projet. Par exemple, un sprint permet d'ajouter une nouvelle fonctionnalité, avec tous les aspects que cela implique.

Avant de démarrer un sprint, il faut le préparer. Pour préparer un sprint, l'équipe qui va collaborer, doit se réunir et se mettre d'accord sur un planning. C'est cette première réunion qui définit le début d'un sprint. Lors de cette réunion, l'équipe va dépiler le « backlog », qui doit être trier et organiser au préalable par le PO (Product Owner) du projet. Le backlog est composé d'user story. L'équipe va devoir choisir un certain nombre d'user story, les estimer en « points », afin de pouvoir « remplir » au maximum le sprint, de manière à concorder avec leur vélocité. La vélocité d'une équipe se calcule au fur et à mesure des sprints, et permet de savoir la capacité d'une équipe à produire un certain nombre de tâches dans un sprint. La vélocité est exprimée en points, ce qui concorde avec les points d'estimation d'une user story.

Pour ce projet, l'équipe qui doit collaborer sur ce projet est composée de plusieurs sous équipes : web et mobile, ainsi que le CPO.

Pour préparer un sprint, nous devons donc faire une réunion préparatoire permettant d'en définir le périmètre, c'est à dire quelle en sera la valeur ajoutée. L'ordre des valeurs ajoutées est ordonné par le responsable produit, le CPO.

J'assiste donc à ces réunions préparatoires, en tant que membre de l'équipe, et plus précisément, de la sous équipe web. Mon rôle consiste à participer à examiner et sélectionner des user story, puis d'en estimer la durée (ou plus exactement la difficulté) afin d'en déduire si oui ou non on l'intègre dans le sprint. Je dois aussi également débattre avec les autres membres quand nos avis divergent sur l'estimation d'une durée. Les débats sont constructifs, ils m'apportent souvent une grande quantité d'informations dont je manquais pour comprendre l'intégralité de l'user story, et parfois même de la fonctionnalité. Grâce à ces débats j'ai pu, une fois de plus, élargir mon champ de connaissances sur l'écosystème de Liberty Rider et en tirer aisance pour la suite.

Il arrive régulièrement qu'une user story ou une tâche dans le backlog soit mal décrit. Il convient donc à l'équipe de devoir affiner cette dernière et savoir si elle doit être sélectionnée pour le sprint à venir.

Une fois le sprint organisé et planifié, l'équipe peut donc démarrer la réalisation des tâches contenues dans ce dernier, avant la fin de la date de fin définie.

4.4.2.3. Daily meeting

Lors du déroulement d'un sprint, l'équipe se réunie une fois par jour, le matin. Ces réunions sont de durée courte (quinze minutes maximum). Le but de cette réunion est que chaque membre de l'équipe expose oralement sur quelle tâche il travaille et s'il rencontre des points bloquants. De cette manière, si le membre rencontre un point bloquant, en faire part aux autres membres permet d'anticiper les éventuels retards engendrés sur des tâches dont le démarrage dépend de la tâche bloquée. Il arrive régulièrement qu'un membre de l'équipe puisse aider le membre bloqué et par conséquent éviter tout retard. Un autre avantage de cette réunion est de permettre d'avoir une vision générale d'où en est chaque membre et en déduire si le sprint est en retard ou non. Si le sprint s'avère être en retard, des actions peuvent être prises pour mieux anticiper le prochain.

En tant que membre de l'équipe du projet, j'assiste quotidiennement à ces réunions. Par ce biais, j'ai pu accroître et pratiquer mes connaissances sur les pratiques des méthodes agiles.

4.4.2.4. Rétrospective de sprint

Dans le cadre de la méthodologie agile, nous effectuons à chaque fin de sprint une rétrospective de ce dernier. L'objectif en est simple, fêter les victoires et les points positifs et chercher ensemble les solutions à apporter sur les éléments empêchant l'équipe de tenir ses promesses et de travailler dans un environnement sain et bienveillant. De plus, elle permet calculer la vélocité de l'équipe réalisée sur le sprint.

Lors de la fin de chaque sprint, le déploiement des travaux réalisés est effectué, permettant d'ajouter au projet l'incrément développé. À la suite de ce déploiement, le product owner peut commencer son analyse des KPI (cette partie est développée dans le chapitre **Suivi des Key Performance Indicator**).

Voici la vélocité moyenne de notre équipe après plusieurs sprints :

- **26 points** pour la partie web
- **33 points** pour la partie iOS
- **36 points** pour la partie Android

Notre vélocité totale de l'équipe est donc de **95 points**. Ce qui est une vélocité correcte pour la taille de l'équipe.

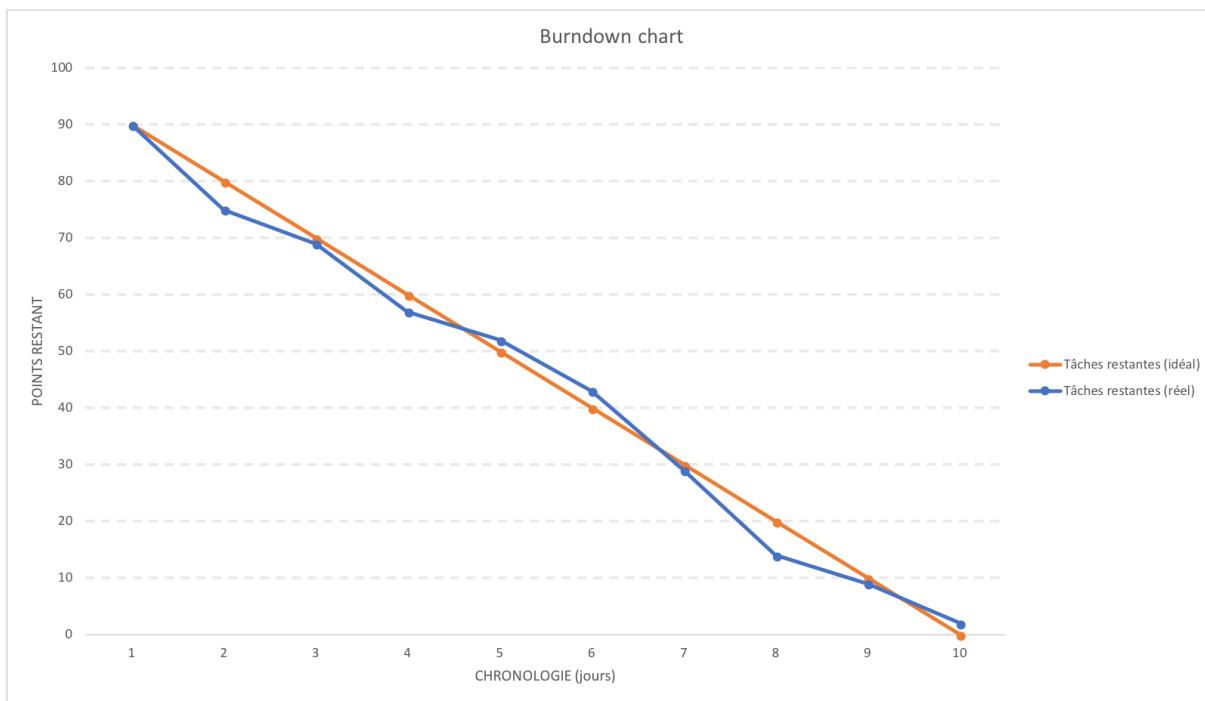
La rétrospective de sprint permet également à chaque membre de l'équipe de s'exprimer librement sur le déroulement du sprint. Cette étape de la rétrospective s'appelle **DAKI**



(Drop, Add, Keep, Improve). Nous utilisons la méthode DAKI, avec des post-it, pour exprimer des idées ou bien des améliorations pour améliorer le déroulement des sprints suivants.

Ce n'est pas tout, en effet la rétrospective de sprint est un événement important pour le product owner, puisque c'est à ce moment-là qu'il peut réaliser des **burndown chart** et **burnup chart**. Les burnup charts et burndown charts sont des graphiques permettant de suivre la progression du développement d'un produit en agile, au niveau d'un sprint ou d'un projet entier. Le burndown chart, souvent utilisé au niveau du sprint, permet de mieux visualiser le reste à faire sur une courte période. Le burn-up chart permet de mieux représenter les évolutions de périmètre du sprint ou du projet

Pour les besoins de la formation, j'ai eu la charge de réaliser le burn-down chart de ce projet pour le sixième sprint.





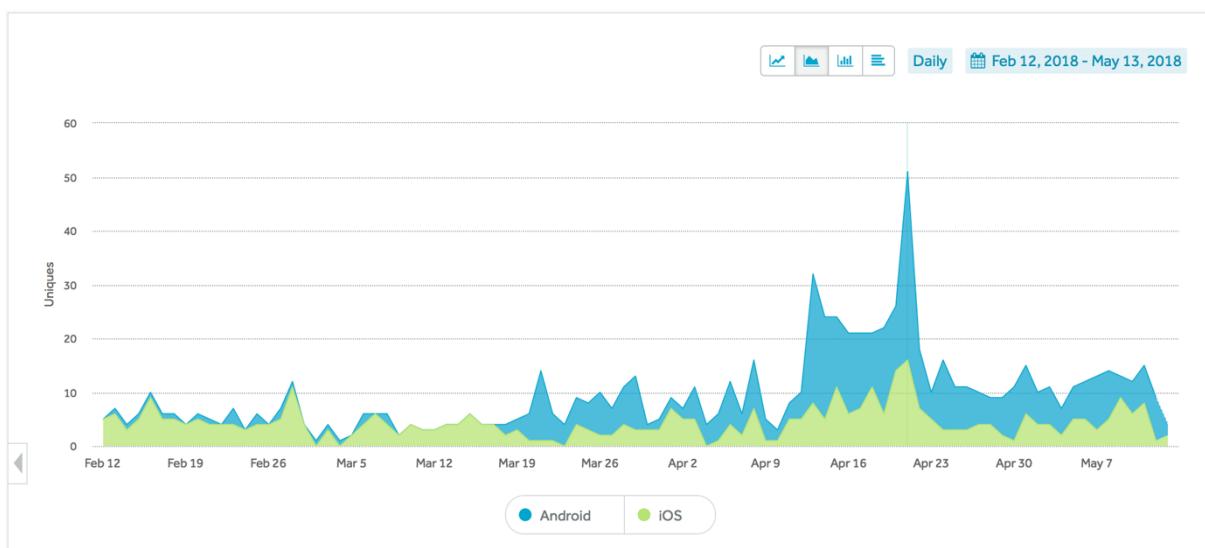
4.4.2.5. Suivi des Key Performance Indicator

Certaines fonctionnalités apportées par ce projet ont pour but d'améliorer le confort utilisateur lors de l'utilisation de l'application, ou bien d'augmenter le nombre de fonctionnalités premium pour favoriser les utilisateurs à passer premium afin d'accroître le chiffre d'affaires de Liberty Rider.

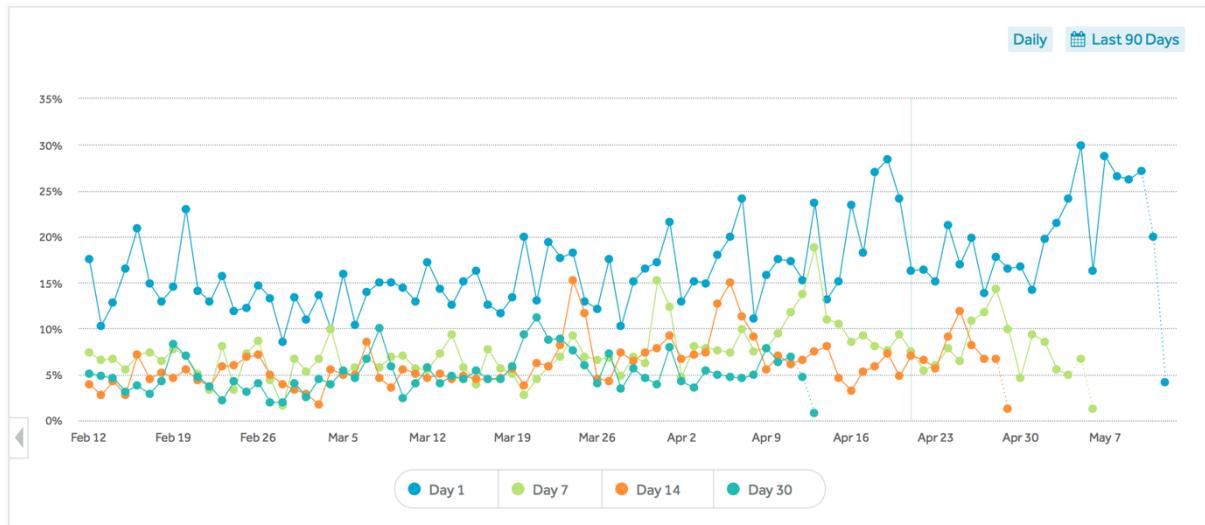
Le CPO est chargé de savoir si les fonctionnalités apportées répondent correctement aux besoins exprimés lors de leur mise en place. Pour ce faire, il doit analyser des indicateurs fournis par des outils, et voir si les résultats attendus sont bien respectés. Ces indicateurs sont nommés KPI (Key Performance Indicator).

J'ai pu assister au suivi des KPI, réalisé par le CPO, et connaître si en effet nos travaux durant les sprints ont permis de répondre aux besoins exprimés. Les résultats ne sont pas toujours positifs. Il arrive que parfois les KPI montrent une bonne réactivité des utilisateurs avec la nouvelle fonctionnalité, permettant d'accroître la rétention de ces derniers, mais il arrive que parfois aucun changement n'apparaît. Dans ce dernier cas, on en déduit que soit les besoins exprimés étaient trop ambitieux, ou bien en décalage avec les besoins utilisateurs. Il en résulte un temps de travail perdu, ainsi que de l'argent consommé. Il est donc vital de savoir s'approcher au plus près de ses utilisateurs pour en connaître au mieux leurs besoins.

Voici quelques exemples de graphiques issus de l'outil **Amplitude**, permettant de créer des tableaux de bord et y insérer tout type de graphique représentant une métrique :



Graphique montrant la courbe des achats de compte premium entre le 12 février 2018 et le 13 mai 2018


LIBERTY RIDER

Graphique montrant la rétention des utilisateurs sur les 90 derniers jours

4.4.3. Développement

4.4.3.1. Développement modulaire

Ce projet a deux objectifs, dont l'un d'eux est de migrer le système actuel vers un nouveau système, plus robuste, évolutif et maintenable. Migrer un système vers un nouveau comporte de multiplex complexité : Il faut avoir une connaissance globale du système à migrer et les erreurs qu'il comporte pour les corriger, garder une passerelle entre les deux mondes pour garder de la rétrocompatibilité avec les clients, ainsi que bien d'autres problématiques.

Pour mieux aborder ce développement, nous avons décidé de le prévoir de manière modulaire. Le cahier des charges, présenté dans la partie **Cahier des Charges**, montre que tout a été découpé par module. Par ce fait, les différents modules (utilisateur, trajet, contact, etc.) peuvent être développés de manière presque indépendante (certains modules fonctionnent ensemble). Le développement du projet en est grandement simplifié.

Pour la partie technique, un module est composé de plusieurs fichiers :

- **Le repository** — contient la logique du module ainsi que son interface avec la base de données
- Ses tests unitaires, intégrations et systèmes
- Ses configurations
- Son schéma GraphQL



Une fois un modulé intégralement développé, nous devons le migrer, c'est à dire changer sa source de vérité (l'endroit d'où provient la donnée). Pour changer sa source de vérité, un script de migration est prévu afin d'importer les anciennes données vers la nouvelle structure. Cependant, lors de l'écriture des données, et pour des questions de sécurité, le module continue d'écrire les nouvelles données dans l'ancien et le nouveau monde.

Cette gestion de la source de vérité est contrôlée par la configuration du module. Grâce à cette dernière, nous pouvons décider si un module doit lire/écrire ses données dans Firebase ou PostgreSQL (qui est le nouveau moteur de stockage de données).

```
1 const config = {};
2
3 config.readFirebase = true;
4 config.readRidesFromFirebase = process.env.USER_RIDE_SOURCE === 'firebase';
5 config.readUsersFromFirebase = process.env.USER_RIDE_SOURCE === 'firebase';
6 config.readContactsFromFirebase = process.env.CONTRACT_SOURCE !== 'postgres';
7 config.readPreviewFromFirebase = false;
8 config.writeFirebase = true;
```

Partie d'un fichier de configuration montrant les différentes sources de vérité pour quelques modules

Dans la configuration ci-dessus, on peut observer différents comportements, dont ceux spécifiques à chaque module (lignes 4 à 7).

Durant ce projet j'ai eu l'opportunité de pouvoir travailler sur une grande majorité des modules de l'application, par exemple : le module utilisateur, trajet, contact, événement, préférence, véhicule, etc. Cette expérience m'a beaucoup apporté quant au fait de savoir prendre du recul et voir un système dans sa globalité plutôt qu'un fragment. De plus, je n'avais jamais eu l'occasion de migrer un système complexe, et dont les choix devaient être réfléchis afin de ne pas perturber les utilisateurs déjà présents et qui utilisent l'application.

Un des aspects délicats et complexes de ce projet est la rétrocompatibilité. Plusieurs mondes cohabitent : les anciennes versions de l'application sur les mobiles et les différences de développement entre iOS et Android. Un des buts de ce projet est de permettre la coexistence de toutes ces différences, sans que le déploiement d'un nouveau module ne vienne perturber l'équilibre d'un des mondes (le client). Pour pallier cette complexité, le choix d'utiliser **GraphQL** est tombé sous le sens. Ce dernier permet de faire évoluer une API, tout en gardant compatibles les anciennes versions d'application qui n'ont pas évolué.

4.4.3.2. Utilisation de GraphQL

GraphQL est un langage de requêtes, développé par Facebook, proposant une alternative aux REST (Representational State Transfer) API. Il propose au client de formuler la structure de données dans la requête, alors cette même structure est retournée par le serveur.

Un des avantages de GraphQL est de permettant la rétrocompatibilité avec toutes les versions d'API utilisées par les clients. De cette manière, aucun dysfonctionnement ne sera perçu par les clients ne mettant plus à jour leur application. Cette problématique était l'une des majeures pour ce projet, puisque nous devions toujours nous assurer, durant la migration, que tout le monde puisse toujours continuer à avoir accès à l'application.

Le schéma GraphQL est défini selon les besoins des utilisateurs et non la base de données. Cette bonne pratique est importante car GraphQL doit avant tout convenir aux vues de l'application cliente et en faciliter la récupération de données.

Chaque module contient son propre schéma. Ayant eu la responsabilité de travailler sur la majorité des modules du projet, j'ai donc pu travailler sur les schémas GraphQL de ces derniers. Grâce à la philosophie de GraphQL, j'ai dû apprendre à collaborer avec les équipes mobiles pour nous mettre d'accord sur le schéma GraphQL qui leur convient le mieux selon leurs besoins. De plus, cela m'a également permis d'élargir mes compétences techniques.

```

1 type Query {
2   # The user on behalf of whom the query is made.
3   currentUser: User
4
5   # Lookup a ride by its sessionId.
6   rideBySessionId(sessionId: String!): Ride
7
8   # Try to find an user by his email (facebook, google, custom)
9   findUidByEmail(email: String!): String
10
11  # Lookup a ride by its sharingToken, a shorter identifier than the ID.
12  rideBySharingToken(sharingToken: String!): Ride
13 }
```

Exemple de schéma GraphQL

4.4.3.3. Augmentation de la robustesse du code par les tests

Le développement d'une application ne se limite pas à son code métier, d'autant plus si l'application est imposante. Une application imposante peut rapidement devenir complexe à maintenir et faire évoluer, surtout que bien souvent plusieurs développeurs doivent collaborer dessus. Il devient vite laborieux et chronophage de savoir si une modification du code engendre une quelconque régression. Afin d'éviter ces complications, il est nécessaire de rendre l'application plus robuste. Pour ce faire, les développeurs ont mis en place des tests. On peut retrouver plusieurs types de tests :

- Unitaires — Test une partie précise du logiciel
- Fonctionnel — Test plusieurs briques du logiciel, qui fonctionnent ensemble
- Système — Test de bout en bout une fonctionnalité du logiciel

Pour ce projet, nous avons mis en place ces trois types de test. Rendre l'API robuste et fiable était un besoin essentiel pour l'évolution du projet. Nous pouvons, de cette manière, garantir que la migration d'un module ne causera aucun dommage collatéral et correspondra aux besoins. De plus, **BDD (Behavior-Driven Development)** est une méthode agile permettant d'écrire les tests selon les user story définies par le product owner. Le processus BDD met en avant le langage naturel et les interactions dans le processus de développement logiciel.

Ayant eu déjà des expériences en termes d'écriture de tests unitaires, j'ai eu la charge de proposer la manière de les rédiger. Cela m'a permis de pouvoir apporter de la qualité au projet en étant force de proposition. J'ai eu aussi la responsabilité de rédiger des tests fonctionnels et systèmes pour chaque module développé. Cela m'a permis de renforcer certaines connaissances et d'en apprendre de nouvelles en matière de tests.

Les tests du projet sont exécutés par l'outil d'intégration continue, CircleCI. De manière plus technique, CircleCI lance une commande **NPM (Node Package Manager)**, utilisant le framework de test **Mocha**. La librairie utilisée pour rédiger les tests est **Chai**.



Mocha



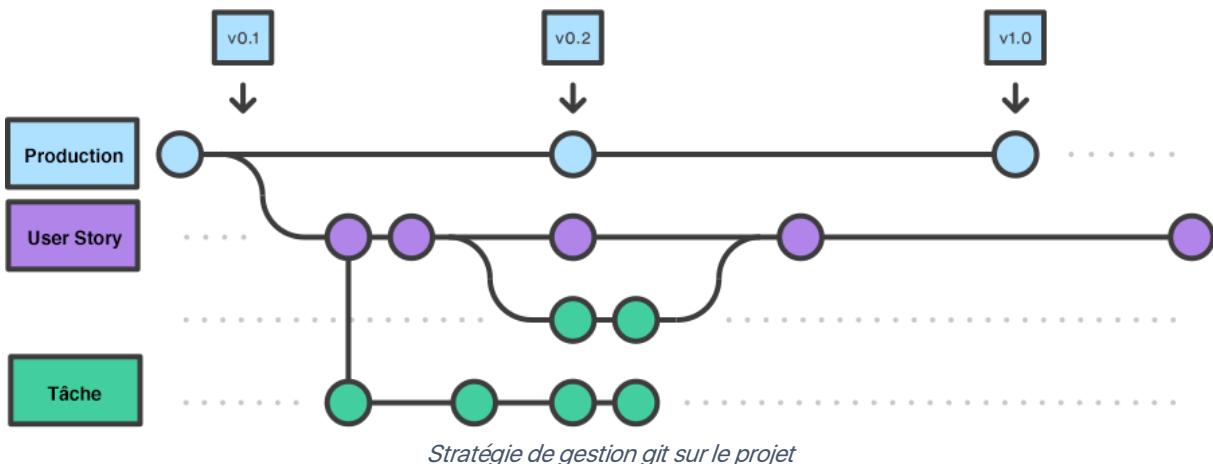
Chai

4.4.3.4. Flux de travail Git et GitHub

Pour la mise en place de l'intégration continue, et afin de collaborer en équipe, nous utilisons le logiciel de gestion de versions **Git**. Le choix de Git a été fait selon la popularité de ce dernier, de sa praticité ainsi que son exploit technique qu'il a permis contrairement à ses prédecesseurs.

Nous avons mis en place une stratégie de gestion de branchage Git basée sur la méthodologie agile. De cette manière, nous pouvons créer une branche portant le nom d'une user story, et chaque sous-branche de cette branche seront les tâches à réaliser. Grâce à cette stratégie, il est simple de pouvoir suivre et traquer les fonctionnalités qui ont été terminées qui seront mises en production à la fin du sprint. De plus, il est aussi facile de pouvoir choisir, pour le product owner, de demander aux développeurs de retirer une fonctionnalité précise pour le déploiement du sprint.

Afin d'améliorer cette stratégie, j'ai mis en place une manière plus robuste de faire converger les branches entre elles : le **rebasage**. Le rebasage est une méthode connue du monde communautaire de l'informatique, puisqu'il est majoritairement utilisé pour collaborer sur des logiciels open source. Elle permet de conserver un historique linéaire de la branche **production**, ce qui facilite le retour en arrière si un problème est survenu ou pour retirer une fonctionnalité.



Cependant, Git seul ne permet pas de pouvoir versionner son code. Il faut un outil (des serveurs) permettant de centraliser. Nous avons décidé d'utiliser **GitHub** pour cela. GitHub nous permet de renforcer la collaboration avec une stratégie de **revue**. Lorsqu'un développeur désirer fusionner la branche de sa tâche dans la branche de l'user story, il peut notifier les autres développeurs du projet. Ils pourront lui faire des retours et proposer des améliorations ou bien corriger des bugs si besoins. De cette manière, nous réduisons considérablement le nombre de bugs envoyés en production.

4.4.4. Contraintes

4.4.4.1. Adoption de la méthode kanban

Ce projet est le premier projet, pour la startup, réalisé avec la méthode agile kanban. Cette méthodologie a été apporté par l'arrivée du nouveau CPO. Cependant, comme la majorité des effectifs présents n'étaient pas formés à cette méthodologie, les débuts n'ont pas pu être aussi efficace qu'attendu. J'ai pu effectuer les constatations suivantes :

- Une phase de formation de l'équipe à la méthode agile était nécessaire
- Des non respects de la méthodologie ont été révélés pour cause le manque d'expérience sur cette dernière
- Quelques pertes de temps sur le projet pour apprendre la gestion de projet à l'équipe

Ces constatations sont issues de plusieurs semaines de sprint, en prenant du recul. Ces constatations ne sont pas négatives puisque j'ai pu en faire part lors des rétrospectives de sprint, et permettre aux membres de l'équipe de palier à ces contraintes.

4.4.4.2. Délicatesse de la migration

Comme le cadre du projet est essentiellement autour de la migration du système, nous avons rapidement rencontré quelques contraintes. Il devenait parfois délicat de prendre des décisions. Comme il s'agit d'une migration du système, cela implique d'arriver à migrer sans perturber un comportement existant. En outre, il est vital qu'aucun client ne soit victime d'un bug l'empêchant d'utiliser son application. Il est donc devenu délicat de savoir faire cohabiter une grande quantité de versions d'application. Nous avons dû procéder à un « force update », visant à forcer les utilisateurs à mettre à jour leur application afin de pouvoir continuer les développements.

4.4.4.3. Mise en place et utilisation des nouveaux outils

La dernière contrainte identifiée concerne les nouveaux outils. Avec l'arrivée de nouveaux membres dans l'équipe, chacun a pu apporter ses idées et ses outils. Cependant, tout le monde ne connaît pas nécessairement les outils proposés et adoptés.

Certaines phases du projet ont nécessité de former l'intégralité de l'équipe aux outils (par exemple l'outil **Amplitude**), entraînant quelques retards sur les délais prévus. Bien entendu, ces formations étaient vitales, il était impossible de les éviter, mais le temps consommé par les formations est négligeable contrairement à la durée du projet, il est donc possible de pouvoir négliger cette contrainte.

4.5. Mise en production

La mise en production est réalisée à chaque fin de sprint. Comme le but d'un sprint est d'ajouter un incrément stable pour le produit, nous déployons en production à la fin de ce dernier. L'avantage de cette stratégie est de pouvoir déployer quelques fonctionnalités, et par conséquent réduire le nombre de bugs possibles en production.

Pour la partie technique, la mise en production est constituée d'une chaîne complète. Le premier maillon est GitHub. Lorsque la branche **staging** est fusionnée sur la branche **production**, un **webhook** va déclencher le processus de CircleCI. A son tour, CircleCI pourra exécuter les commandes qui lui sont demandées : Construire le projet, exécuter les tests, puis déployer une nouvelle révision de tâche du projet construit sur Amazon Web Services. La nouvelle tâche exécutera un conteneur Docker, qui contient l'application. Une fois cette chaîne terminée, ce qui dure cinq minutes environ, la nouvelle version de l'application, contenant l'incrément du sprint, est disponible en production.

4.6. Retours sur investissement

4.6.1. Augmentation des performances

À la suite de plusieurs mois de développement, et d'un travail intense d'une équipe complète, les premiers retours sur investissement sont apparus. Firebase est un élément important à réduire de l'écosystème de Liberty Rider, et grâce à ce projet, son effacement partiel a permis d'améliorer considérablement les performances de l'application. Comme expliqué dans la partie **Contexte et objectifs** de ce projet, les derniers temps étaient parfois difficiles lorsque Firebase était complètement submergé par les requêtes client. Grâce à la mise en place de la nouvelle API, les outils d'analyse de performances montrent d'excellents résultats quant aux temps de réponse pour les clients.

4.6.2. Réduction des coûts

Un autre objectif du projet est la réduction des charges. Avant la réalisation de ce dernier, Firebase coûtait près de 700€ par mois, soit 8,400€ par an. À cela s'ajoutait les latences de Firebase, plusieurs fois par jour. Avec l'arrivée de la nouvelle API, les frais de Firebase ont été drastiquement diminués. Les frais économisés peuvent être réutiliser pour d'autres besoins : Marketing, serveurs, et d'autres outils facilitant le travail des développeurs ou bien la gestion de projet.

5. Transformation des processus

5.1. Nouvel environnement agile

5.1.1. Proposition et mise en place d'une gestion agile

A mon arrivée chez Liberty Rider, la gestion de projet n'était pas guidée par une méthode agile. Des outils et des processus avaient fait leur apparition, mais sans application réelle, ou bien mal réalisée.

Au vu du contexte de ma formation, ainsi qu'un peu d'expérience, j'ai proposé à Martin, qui s'occupait de superviser tous les projets à ce moment-là, la mise en place d'une méthodologie agile. Je n'étais pas nécessairement capable de mettre en place l'intégralité de la méthodologie, ainsi qu'en assurer le bon déroulement. Cependant, j'ai tout de même proposé la pratique de certaines briques de la méthodologie : les daily meeting, ainsi que des sprints.

Les membres de l'équipe communiquaient de manière non formelle, le daily meeting me semblait être une bonne pratique permettant d'informer chaque membre sur le travail réalisé par les autres. Cette pratique avait pour objectif de sensibiliser chacun à avoir une vision plus globale de ce qui se passe.

La deuxième proposition que j'ai faite concernait la mise en place de sprints. Comme Liberty Rider est un éditeur de son propre logiciel, aucun délai n'était réellement mis en place, ni suivi. Certaines dates butoir étaient fixées, mais il arrivait qu'elles n'étaient pas respectées, sans qu'aucune action ne soit prise dans ce type de circonstance. L'objectif de cette solution était de commencer à définir des périmètres de tâches à réaliser dans un délai fixé, et, par conséquent, prendre des décisions en cas de débordement des délais. De cette manière, les développements auraient été plus cadrés, et mieux définis dans le temps, permettant une meilleure vision à court terme de l'évolution du produit.

Comme je venais d'arriver dans l'entreprise, et n'ayant pas d'expériences solides dans la gestion de projet avec une méthode agile, mes propositions n'ont pas été appliquées. Cependant, ces solutions ont finalement été mises en place à l'arrivée des nouveaux membres dans l'équipe, et notamment le CPO, à l'origine de ces changements.

5.1.2. Force de proposition pour des outils plus adaptés

Des outils pertinents ont été développés pour mieux répondre aux besoins d'une gestion de projet en méthodologie agile. J'ai eu l'opportunité de pouvoir travailler avec un de ces outils, lors de précédentes expériences : **Jira**.

Jira est un système de suivi de bugs, un système de gestion des incidents, et un système de gestion de projets développé par Atlassian. Il est majoritairement adopté et apprécié par la communauté agile.

Dans une démarche de mise en place d'une gestion de projet agile, j'ai proposé l'utilisation de l'outil Jira pour remplacer Asana. Asana est un outil complet, mais au sein de notre équipe, une majorité s'en plaint du fait de sa mauvaise expérience utilisateur. Malheureusement, l'utilisation d'Asana s'est renforcée avec l'arrivée d'une gestion de projet agile, rendant plus délicat et complexe la migration vers Jira. Le débat reste encore d'actualité, mais aucune proposition de migration correcte n'a encore été proposée.

5.2. Nouvel environnement technique

La réalisation des projets **Migration et nouvelles fonctionnalités** et **Nouvelle version de détection d'accidents** est un changement considérable dans la qualité et la solidité du code. Avec l'arrivée de ces projets, des technologies modernes et robustes, ainsi que des pratiques, ont été intégrées : Docker, GraphQL, React, Angular, tests unitaires, tests d'intégration, tests système, intégration continue, etc. Pour ma part j'ai apporté l'utilisation de React et Angular, et les pratiques de tests unitaires.

Grâce à ce nouvel environnement, nous pouvons avancer de manière plus sereine sur le développement, puisque les tests vont nous assurer qu'aucune régression n'est apparue, et les applications web, utilisant des technologies modernes, sont optimisées et fournissent une meilleure expérience à l'utilisateur.

On dénote également que la maintenabilité et l'évolutivité des différents projets ont été accrues. Par exemple, l'utilisation de GraphQL permet de pouvoir maintenir simplement plusieurs versions d'une API. Par ce biais, aucune application mobile anciennes sera toujours capable de communiquer avec l'API et recevoir ses informations.

De plus, avec la nouvelle architecture des projets, les perspectives de collaboration des développeurs sont nettement améliorées. Ceci est rendu possible grâce à des développements modulaires ainsi que l'utilisation d'outils connus de la communauté des développeurs, rendant plus familier les nouveaux développeurs avec l'environnement.

5.3. Nouvelle politique de confidentialité

Le **RGPD** (Règle Général sur la Protection des Données), est un règlement qui a été voté le 14 avril 2016, ayant pour objectif de renforcer la protection des données personnelles pour les individus au sein de l'Union Européenne. Les dispositions du règlement sont applicables au vingt-huit états membres de l'Union Européenne à compter du 25 mai 2018.

La majorité des entreprises recueillant des données personnelles ont mis à jour récemment leur politique de confidentialité à ce sujet. Chez Liberty Rider, nous avons nous aussi effectué des réunions, auxquelles j'ai assisté, pour connaître les actions à réaliser afin de respecter la norme RGPD.

A l'issue de ces réunions, j'ai eu la responsabilité de devoir vérifier le respect des nouvelles normes quant aux types de données récoltées auprès de nos utilisateurs. Après un audit des données récoltées, voici celles les plus sensibles et sujettes à devoir être changées ou rendues anonymes :

- Données médicales (groupe sanguin, anciennes opérations, etc.)
- Données d'identité (nom, prénom, âge, sexe, etc.)
- Trajets parcourus avec l'application
- Numéros des contacts renseignés (pour les SMS d'alerte)

Le RGPD prévoit aussi aux utilisateurs de pouvoir effacer de manière certaine toutes ses données récoltées par l'application, et aussi de pouvoir révoquer, à tout moment, son accord avec les conditions d'utilisation (acceptées lors de son inscription).

A la suite de l'audit, il était difficile de ne plus stocker certaines informations qui sont nécessaires à IMA (les données médicales par exemple). Nous devons donc organiser à nouveau des réunions, avec l'avocate de l'entreprise, pour savoir quels seraient les impacts sur le retrait de telles données et quels sont les limites légales applicables pour conserver le maximum d'informations déjà présentes.

6. Projet personnel : SnipHub



6.1. 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, celui de devoir chercher de manière continue des bouts de code que j'ai déjà utilisés ou codés plusieurs fois auparavant.

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 trouver ma réponse. Il apparaissait de manière évidente qu'il y avait un manque d'organisation à ce sujet, une idée de projet devenait possible.

Souvent, les bouts de code réutilisables, appelés snippets, sont épars 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 part du projet à un ami développeur, qui lui aussi avait effectué le même constat.

Ensemble, nous prenons la décision de se mettre en collaboration pour ce projet.

6.2. 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 ou StackOverflow. Il arrive parfois qu'ils perdent leur temps à chercher plusieurs fois le même snippet d'un projet à l'autre, on constate donc un réel besoin à ce sujet, ce qui confirme notre idée de projet.

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 dont 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), mais ils sont 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 snippet 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 pertinents. L'ajout de snippets n'est pas simple (l'aspect communautaire est mal élaboré), absence d'informations de qualité (les « j'aime », les commentaires, etc.).

6.3. Spécification, estimation, coûts

6.3.1. 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 le but de mieux organiser nos idées en avance de phase.

4.2.2. 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.

4.2.3. 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.

4.2.4. 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)

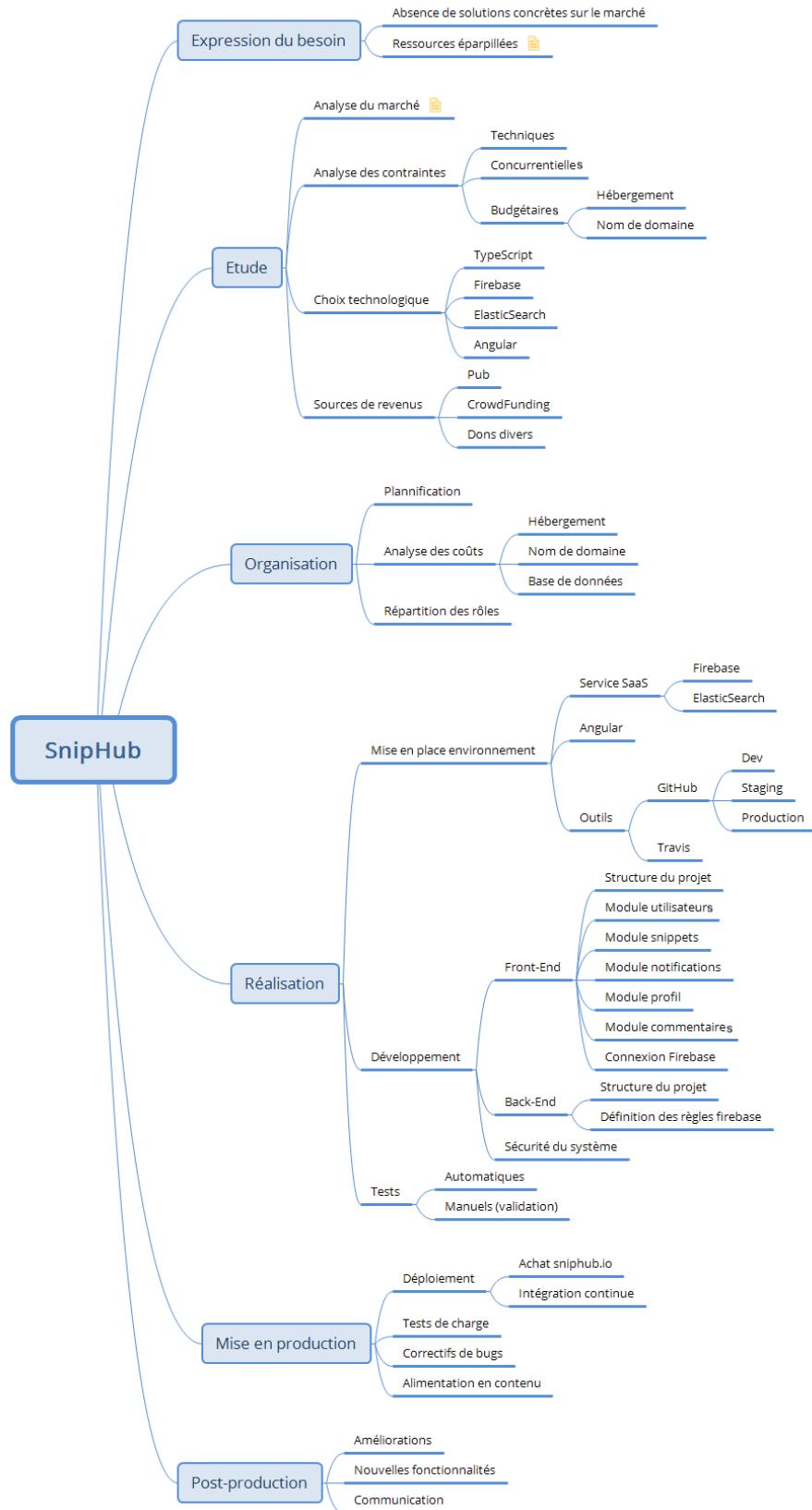
6.3.2. Work Breakdown Structure

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, ce qui nous a permis 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 :

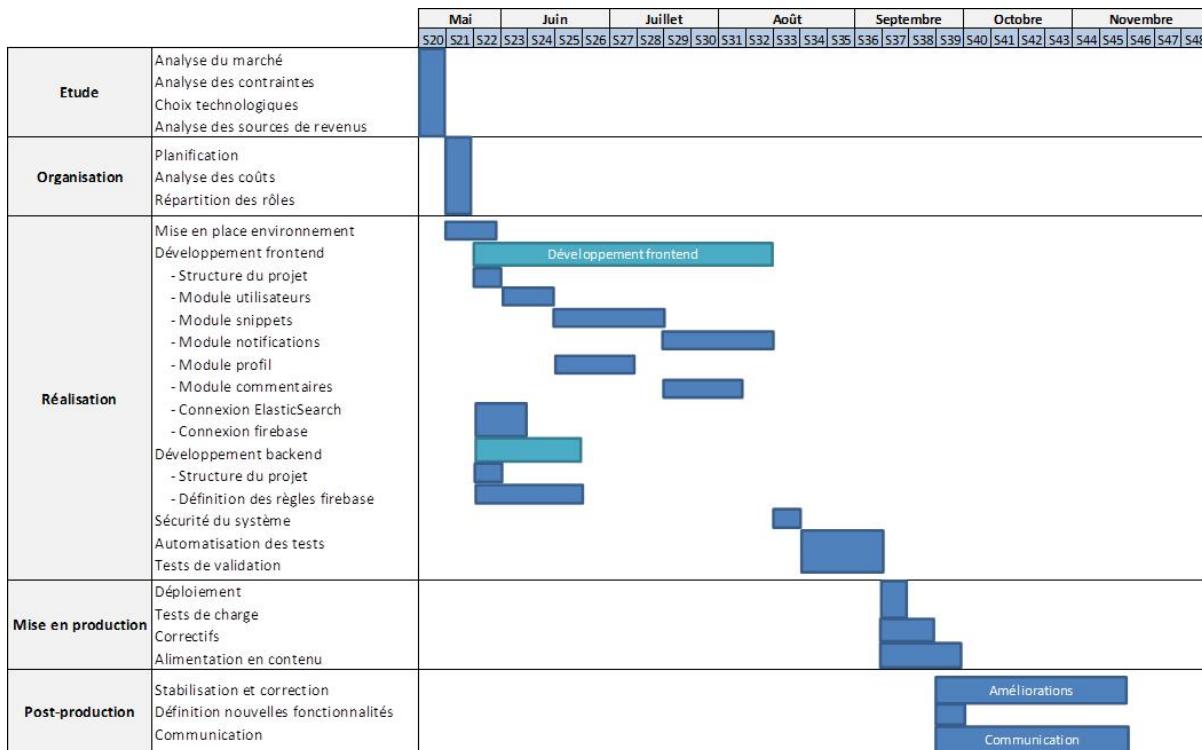
- **Expression du besoin** : Prise de conscience à l'origine du projet
- **Étude** : Phase durant laquelle des recherches ont été effectuées, notamment sur la question des concurrents, l'analyse du marché et les différentes sources de revenus.
- **Organisation** : Mise en place de la gestion de projet 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, à savoir : la mise en place de l'environnement, le développement des fonctionnalités et des tests, augmentant la maintenabilité et la qualité du code. Cette phase est souvent sous-estimée, ce qui a 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 effectuera un déploiement de l'application, suivi d'un test de charge pour s'assurer du volume d'utilisateurs supporté, ainsi des correctifs de bug (plus communément appelés hotfix), et pour finir, l'alimentation du site en contenu.
- **Post-production** : À la suite des retours utilisateurs, des améliorations et de nouvelles fonctionnalités pourront être apportées. Une communication importante est nécessaire pour faire connaître le projet.



Work Breakdown Structure de SnipHub


LIBERTY RIDER

6.3.3. Planning



Le planning prévisionnel de SnipHub a été réalisé avec l'outil **Project**, de **Microsoft**.

Le planning a été construits selon nos estimations, basées sur nos différentes expériences. Chaque estimation de temps sur une tâche technique incluse le développement et les tests de cette dernière, ainsi qu'une certaine marge d'erreur d'estimation.

6.3.4. Analyse des coûts

Ressources humaines	Coût / heure en €
Quentin Pomarel	15.00
Julien Sergeant	15.00

Charges estimées	
Rubrique	Coûts en €
Firebase	240.00
ElasticSearch	50.00
Hébergement	0.00
Nom de domaine	60.00
Mise en place de l'environnement	780.00
Gestion de projet	300.00
Développement frontend	2,535.00
Structure du projet	60.00
Module utilisateurs	600.00
Module snippets	750.00
Module profil	225.00
Module commentaires	225.00
Module notifications	450.00
Connexion ElasticSearch	150.00
Connexion firebase	75.00
Développement backend	360.00
Structure du projet	60.00
Définition des règles firebase	300.00
Total	4,325.00

Simulation d'une estimation de budget

Le budget présenté ci-dessus est une simulation sur les coûts prévisionnels de ce projet.

Cette estimation de budget est simulation, en partie. En effet, nous n'avions pas réellement fondé l'entreprise SnipHub. Il n'était donc pas nécessaire d'avoir des ressources humaines qui coûtent (salaires, etc.), puisque le projet a été réalisé sur notre temps libre.

Cependant, certaines charges restantes immuables : Firebase, ElasticSearch, l'hébergement et le nom de domaine. Ces charges sont emmenées par les tiers, mettant à disposition ces services, elles sont donc nécessaires même si le projet est personnel et en dehors de tout contexte professionnel.

6.4. Réalisation

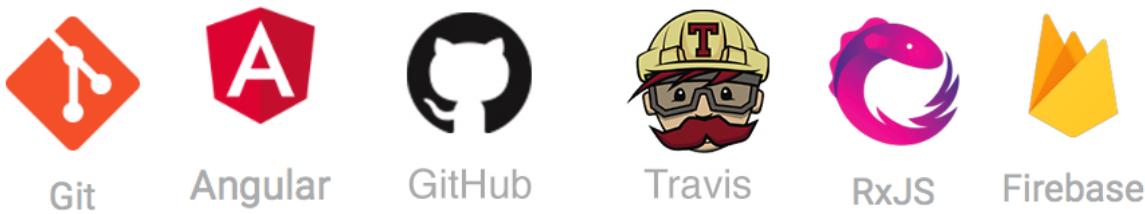
6.4.1. Mise en place de l'environnement

Nous avons décidé de réaliser ce projet en tant que POC (Proof Of Concept), puisque dans un cadre de lean startup, la première version de l'application devait contenir les fonctionnalités minimales pour voir si le projet intéresse des utilisateurs.

Un environnement simple a été choisi, pour plusieurs raisons :

- Simplicité de mise en place et de maintenabilité
- Facilité de changement d'architecture
- Faible coûts

L'environnement choisi est le suivant :



Firebase est une technologie « backend less », c'est à dire qu'il contient déjà une solution permettant de ne réaliser que la partie cliente sans devoir développer aussi une partie backend, ce qui est un gain de temps considérable pour le projet.

Travis a été choisi pour sa simplicité d'intégration avec GitHub, ainsi que sa gratuité. C'est un outil majoritairement utilisé par la communauté de l'open source (ensemble de personnes qui développent et/ou utilisent des logiciels libres de droits).

Angular est un framework frontend, développé par Google, permettant de développer rapidement des applications. Il est donc un excellent candidat pour réaliser ce POC.

RxJS est une librairie de gestion de tâches asynchrones, via la notion d'observables. Le choix de cette librairie est expliqué par le fait d'avoir Firebase, qui permet de faire du temps réel. Cette notion de temps réel, impliquant des tâches asynchrones, sera parfaitement gérée par RxJS.

6.4.2. Développement

La phase de développement s'est déroulée en trois grandes étapes :

- Ouverture du projet sur Firebase, permettant l'accès aux fonctionnalités de ce dernier (connexion utilisateur, base de données, etc.)
- Ouverture d'un projet ElasticSearch sur la plateforme Bonzaï (permettant la recherche de snippets)
- Réalisation de l'application web

Une dernière étape, non mentionnée ci-dessus, est la mise en production, qui sera expliquée plus en détails sans le chapitre **Mise en Production**.

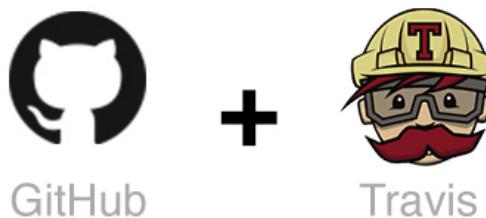
Le développement de l'application s'est déroulé durant nos temps libre, à savoir les soirs de semaine ainsi que les week-ends. Aucun problème majeur n'est apparu durant la phase de développement, les outils sélectionnés en amont ont permis une réalisation sans surprises ni difficultés.

6.4.3. Intégration continue et automatisation des tests

Dans notre philosophie, il était important de ne déployer qu'une version minimale du produit pour en tester l'efficacité sur la communauté des développeurs. Cependant, notre philosophie était aussi de livrer un produit de qualité, et pour renforcer cela, une intégration continue a été mise en place pour assurer l'exécution des tests du projet avant chaque déploiement. Nous désirons avant tout pouvoir assurer qu'aucune régression n'était engendrée par tout déploiement.

L'outil Travis a été choisi pour ce rôle. Son intégration avec GitHub est simple et rapide. En quelques minutes seulement, un début de cadre DevOps était déjà en place sur le projet.

De nos jours, une grande majorité des nouveaux projets possèdent une intégration continue, c'en est devenu une bonne pratique. Nous voulions nous inscrire dans cette démarche, qui est, selon nous, une manière d'apporter de la qualité à un produit.



6.5. Aspects juridiques

Du fait de l'utilisation d'un fournisseur de service (Firebase), et l'achat d'un nom de domaine (via OVH), nous nous devons de connaître les conditions d'utilisation de chacun. Voici ci-joint quelques exemples de « contrats » que nous acceptons lors de l'utilisation des fournisseurs :

Hébergement avec Firebase (Google)

- Le contenu diffusé est sous la responsabilité du client, des revues par le fournisseur peuvent être effectuées afin de s'assurer que rien d'illégal ne soit stocké
- Le fournisseur se réserve le droit de supprimer l'accès et les données si des informations illégales sont présentes
- Le fournisseur ne possède pas la propriété intellectuelle du contenu hébergé
- Le fournisseur se réserve le droit de supprimer l'accès si la propriété intellectuelle n'est pas détenue par le client sur le contenu (copie)

Nom de domaine (OVH)

- Le fournisseur ne sera pas tenu responsable de l'utilisation faite du nom de domaine
- Le client est tenu de fournir des informations exactes au bureau d'enregistrement aux vues de la publication dans des répertoires tel que le WHOIS

Concernant notre application, du fait que nous stockons des données privées des utilisateurs, ainsi que le contenu publié par ces derniers, nous avons des obligations à postériori. Il est donc nécessaire de mettre à disposition des conditions d'utilisation sur SnipHub, couvrant divers points, par exemple :

- L'équipe SnipHub se réserve le droit de supprimer tout commentaire raciste, homophobe, ou profanatoire, portant atteinte à la liberté personnelle
- L'équipe SnipHub se réserve le droit de supprimer l'accès à tout utilisateur faisant usage abusif des fonctionnalités (publication intempestive de commentaires à but non constructif, publication de snippet dénué de sens ou contenant tout propos diffamatoire)
- L'équipe SnipHub s'engage à ne pas divulguer les données personnelles des utilisateurs, sauf cas éventuel d'une attaque visant à dérober ces informations (respect de la CNIL)

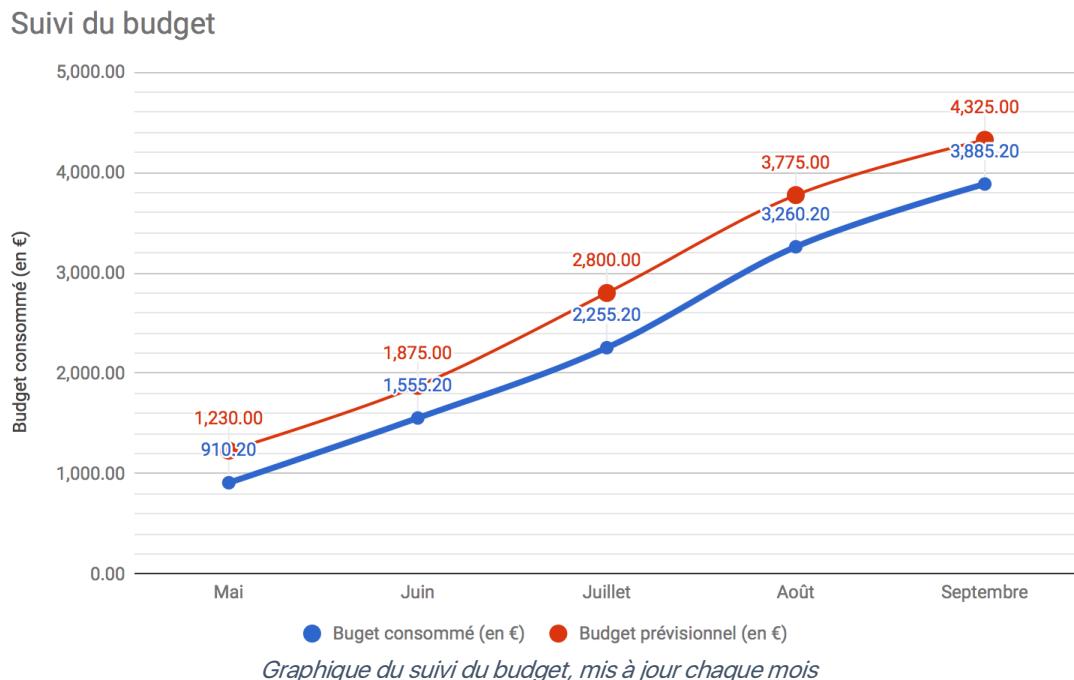
6.6. Mise en production

Grâce à la sélection d'outils modernes et simples, nous savions que la mise en production serait une phase non critique.

Le déploiement du projet se déroule en trois étapes :

- Construction de l'application Angular en mode production – compilation de tous les fichiers de l'application en quelques fichiers statiques, par exemple le fichier `bundle.js` va contenir tout le code JavaScript de l'application. On peut retrouver le même cas pour les styles avec le fichier compilé `styles.css`. On parle de SPA (Single Page Application).
- Déploiement des fichiers statiques compilés sur firebase hosting ainsi que les règles de sécurité de la base de données (firebase hosting est un service firebase permettant de distribuer des fichiers statiques). Par défaut le fichier `index.html` est servi au client en tant que page principale du site.
- Préparation du service ElasticSearch sur Bonsai

6.7. Suivi du budget





LIBERTY RIDER

Ressources humaines	Coût / heure en €
Quentin Pomarel	15.00
Julien Sergent	15.00

Charges réelles	
Rubrique	Coûts en €
Firebase	0.00
ElasticSearch	0.00
Hébergement	0.00
Nom de domaine	30.20
Mise en place de l'environnement	780.00
Gestion de projet	300.00
Développement frontend	2,415.00
Structure du projet	60.00
Module utilisateurs	450.00
Module snippets	750.00
Module profil	150.00
Module commentaires	255.00
Module notifications	525.00
Connexion ElasticSearch	150.00
Connexion firebase	75.00
Développement backend	360.00
Structure du projet	60.00
Définition des règles firebase	300.00
Total	3,885.20

Tableau du budget consommé

Même si la gestion du budget pour ce projet n'était que fictive, une simulation, nous avons tenu à tenir à jour le suivi du budget de manière à savoir si nos prévisions étaient correctes, et par conséquent, notre capacité à savoir estimer et suivre un budget, et savoir prendre des décisions en cas de débordement ou imprévus.

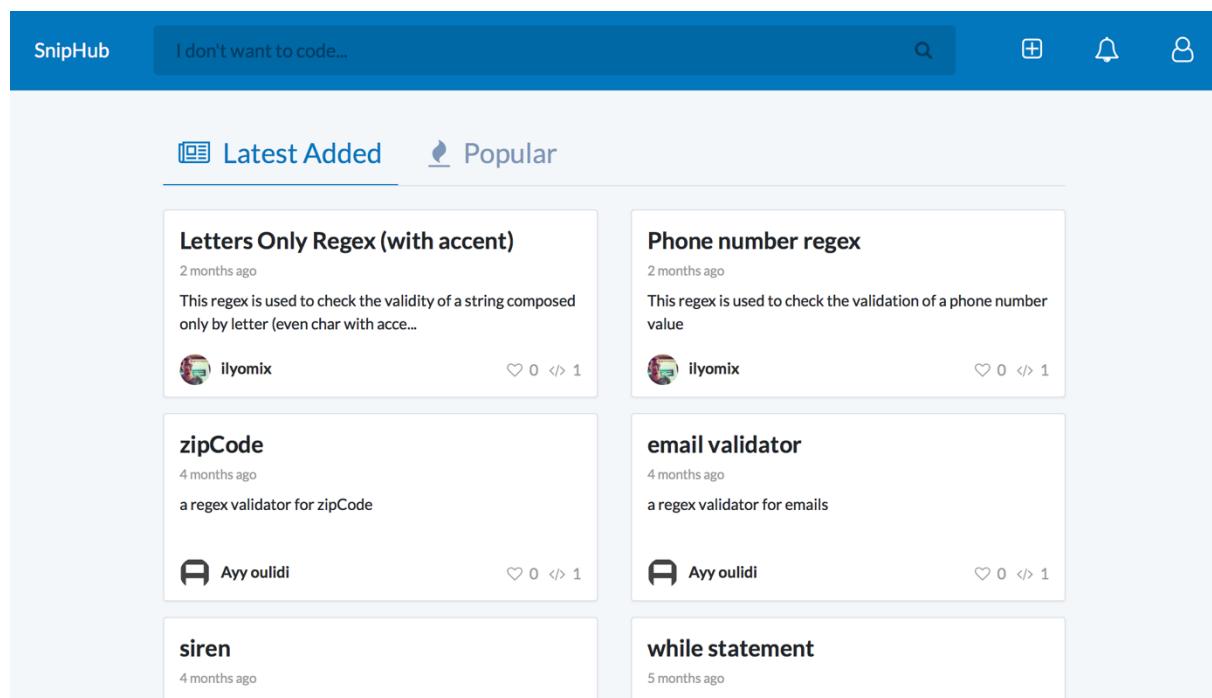
Nos prévisions de budget se sont avérées correctes, puisqu'en effet le dernier tableau montre que le budget total consommé théorique serait de 3,885€ pour un total prévu de 4,325€. Cet écart s'explique par la surestimation de certaines tâches qui n'ont pas nécessité autant de charge de travail, ainsi que certains coûts prévus qui se sont avérés réduits à 0€ (Firebase et ElasticSearch).

6.8. Retour d'expérience

Le bilan du projet SnipHub est très positif. En effet, la réalisation de ce projet a permis de mettre en pratique un large éventail de compétences non pratiquées en entreprise. SnipHub m'a également permis d'avoir un œil plus général sur tous les aspects d'un projet, et non seulement la technique :

- Savoir étudier un marché, les solutions existantes et les concurrents
- Prévoir des charges de travail
- Prévoir un budget
- Savoir s'adapter en cas de non-conformité aux délais ou au budget
- Connaître tous les aspects juridiques obligatoires
- Réaliser une version minimale de l'application pour savoir si les utilisateurs ciblés montrent un intérêt au projet

SnipHub m'a permis de pouvoir réaliser un projet, avec tous les tenants et les aboutissants, du début jusqu'à la fin. J'ai pu améliorer mes compétences techniques, mais aussi apprendre de nouveaux aspects de la gestion de projet à petite échelle. Une fois la première version de l'application déployer, il nous est maintenant possible de pouvoir entrer dans le monde de la startup et faire vivre de manière plus concrète SnipHub.



The screenshot shows the SnipHub application interface. At the top, there is a blue header bar with the text "SnipHub" on the left, a search bar containing "I don't want to code...", and several icons on the right: a magnifying glass, a plus sign, a bell, and a user profile icon. Below the header, there are two tabs: "Latest Added" (which is selected) and "Popular". The main content area displays a grid of six snippet cards:

- Letters Only Regex (with accent)** - posted 2 months ago by ilyomix. Description: This regex is used to check the validity of a string composed only by letter (even char with acc...).
- Phone number regex** - posted 2 months ago by ilyomix. Description: This regex is used to check the validation of a phone number value.
- zipCode** - posted 4 months ago by Ayy oulidi. Description: a regex validator for zipCode.
- email validator** - posted 4 months ago by Ayy oulidi. Description: a regex validator for emails.
- siren** - posted 4 months ago.
- while statement** - posted 5 months ago.

Capture d'écran de l'application web SnipHub

7. Conclusion



8. Annexes

Annexe 1 : Spécifications API de détection d'accidents



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
- firstName: string
- lastName: string
- email: string
- phoneNumber: string
- isDuo: boolean // if user is in duo
- isTest: boolean // if accident is a test
- sessionId: string // can also be sent as rideld
- shockId: string au format UUID
- appVersion: string
- appPlatform: string
- clientTime : double // ms
- clientUploadTime : double // ms
- lastLocations: Location[] // Les **10** dernières locations, au moins **1** location. L'ordre est important. Les locations les plus **récentes** doivent être à la **fin**. Le timestamp de location doit être arrondi à la minute.
 - latitude: decimal
 - longitude: decimal
 - accuracy: decimal
 - timestamp: int // ms
- brandName: string ("Kawasaki")
- modelName: string ("ER6")

Response status code: 204

Pas de data en output

Traitements déclenchés :

- Enregistrer serverUploadTime
- Si erreur imprévue, renvoyer 500
- Vérifier l'idToken
- Si erreur id token: renvoyer 401
- Vérifier intégrité des données (bon format, type, pas vide)
- Si erreur de données: renvoyer 400
- Build XML pour IMA
- Check XML pour IMA
- Si erreur : renvoyer 400
- Envoyer le XML à IMA (config staging seulement) CRITICAL
- Envoyer un email sur alarms@liberty-rider.com (pour déclencher la chaîne Zapier), (config staging et production seulement), comme dans le worker. CRITICAL
- Contacter healthchecks.io si test=true (config différentes par environnement)
- Renvoyer 204 (succès) au client
- Ensuite, poster les infos sur l'interface algorithme (on fait ça après avoir retourné 204 car la télémétrie n'est pas critique), cf [API - Interface algorithme](#)

Architecture Interne

Liste des services :

- IMA service : build, check, send

- Email service : send
- angel-api service : send
- http service : send response

Tests unitaire :

- Test de chaque service
- Test des controllers

Tests système :

- Teste l'intégralité de la chaîne sous forme de scénarios. Envoi des requêtes http, permet d'émuler le comportement d'un client (mobile, web, etc.).

Environments Variations

Paramètre REST isHealthcheck : Le body de la requête emergency contient le paramètre isHealthcheck=true lorsqu'on veut faire un **test automatique** de bout en bout. Envoyé uniquement par angel-scheduler (toutes les 5 min). Comme l'envoi est très fréquent les données ne sont pas stockées. Donc ça se comporte pas à 100% comme une vraie alerte.

Paramètre REST isTest : Le body de la requête emergency contient le paramètre isTest=true lorsqu'on veut faire un **test manuel** de la chaîne de bout en bout. La seule différence de traitement avec une vraie alerte c'est qu'on transmet le paramètre isTest. C'est donc plus réaliste que isHealthcheck. Le seul moyen de faire tourner ce test est d'utiliser un APK dédié.

Interdit d'envoyer à la fois isTest=1 et isHealthcheck=1.



	NODE_ENV=test (circleci et TU) les services externes mockés en TU			NODE_ENV=development (développeur)		
	isTest=0 isHealth=0	isTest=1 isHealth=0	isTest=0 isHealth=1	isTest=0 isHealth=0	isTest=1 isHealth=0	isTest=0 isHealth=1
IMA						
Email						
healthchecks.io						✓
angel-api				✓	✓	

	NODE_ENV=staging info "staging" ajoutée ?			NODE_ENV=production		
	isTest=0 isHealth=0	isTest=1 isHealth=0	isTest=0 isHealth=1	isTest=0 isHealth=0	isTest=1 isHealth=0	isTest=0 isHealth=1
IMA	✓	✓	✓	2018	2018	2018
Email	✓	✓		✓	✓	
healthchecks.io			✓			✓
angel-api	✓	✓		✓	✓	

IMA : envoi de l'alerte à IMA via une API REST en XML. Ils ont un paramètre "isTest" qui correspond au notre, et un paramètre "supervision" qui correspond à notre isHealthcheck.

NOTE: Actuellement IMA est activé seulement en staging !

NOTE TESTS IMA DEBUTANT 28/03 :

- Création d'une application Android flavor IMA TEST
- Envoi d'un paramètre spécial : isIMATest
- Notifier l'environnement de production IMA des alertes uniquement si l'accident est flaggé isIMATest = true
 - Asana : <https://app.asana.com/0/606735458576959/600688449298686>
 - PR : <https://github.com/liberty-rider/emergency-api/pull/42>

Email : envoi d'un email à un Google group dédié alarms@liberty-rider.com, auquel les fondateurs et Zapier sont inscrits. Zapier déclenche l'envoi d'un message Slack sur **#alertes** et l'appel du téléphone d'alerte. C'est le système qu'on utilisait avant IMA, et on le garde en parallèle.

En staging, l'email arrive sur debug+staging+alarms@liberty-rider.com et ne déclenche pas zapier.

healthchecks.io : envoi à <https://healthchecks.io/checks/>, un service qui nous alerte sur Slack dans le canal **#monitoring** lorsque le healthcheck est en retard par rapport à l'intervalle de 5 minutes qui est prévu.

angel-api : envoi des détails de l'alerte à notre API pour être stocké dans notre base de donnée, et affiché sur la carte de la session.

Annexe 3 : Cahier des charges de SnipHub

Cahier des charges Réalisation d'une application web

{ }
/ ** /

Nom de l'entreprise
Nom du projet

SnipHub
SnipHub



Glossaire	3
L'entreprise	4
Le projet	5
Contexte et objectifs	5
Cibles	5
Contenus	5
Livrables attendus	5
Exigences	6
Rôles et responsabilités	6
Fonctionnelles	6
Module utilisateur	6
Inscription	6
Connexion	7
Module snippets	8
Module notification	8
Module profil	8
Module commentaire	9
Calendaires	9
Budgétaire	9
Prestations attendues	10
Développement	10
Hébergement	10
Référencement	10
Nom de domaine	10
Mises à jour	10
Matrice de conformité	11
Module et fonctionnalités	11
Contraintes	12

1. Glossaire

Snippet : bout de code réutilisable

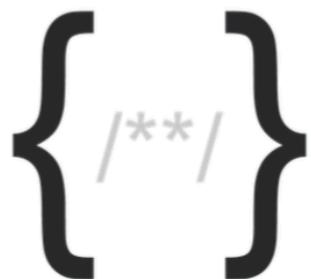
Header : Partie haute du site web, contenant souvent des liens et des onglets

Like : Action de pouvoir aimer quelque chose, dans le cadre du projet on peut aimer un snippet

2. L'entreprise

Nous sommes une jeune startup, SnipHub, spécialisé dans le domaine du Web, ayant pour objectif de devenir un des leader de la communauté des développeurs.

L'entreprise est composé actuellement des deux fondateurs, à savoir Quentin Pomarel et Julien Sergent. Chacun étant issue du monde des développeurs.



logo SnipHub

3. Le projet

3.1. Contexte et objectifs

Le projet SnipHub est une plateforme communautaire, pour développeurs, axée sur le partage et la collaboration de snippets, afin de faciliter la recherche de snippets pour un développeur ou bien qu'il ne réinvente pas la roue tout le temps. Le projet est né suite à un besoin ressenti par ses fondateurs, lors de développement de produits, il y a un manque important de ressources en ligne répertoriant des snippets.

3.2. Cibles

Le projet SnipHub a pour but de répertorier des bouts de code réutilisables, le projet s'adresse donc à toute la communauté des développeurs.

3.3. Contenus

Le site contiendra majoritairement des snippets, ajoutés par les utilisateurs du site.

3.4. Livrables attendus

Les livrables attendus sont les suivants :

- Modules
- Hébergement et configuration
- Plateforme d'administration

4. Exigences

4.1. Rôles et responsabilités

Rôle	Description
Utilisateur non authentifié	Capable de pouvoir s'inscrire et lire des snippets et leur contenu (commentaires, likes).
Utilisateur authentifié	Capable de pouvoir se connecter, de pouvoir créer, modifier, supprimer ses propres snippets, commentaires et likes. Il doit pouvoir aussi modifier ses informations utilisateur.
Administration	Capable de pouvoir créer, modifier, supprimer tout type de contenu du site.

4.2. Fonctionnelles

4.2.1. Module utilisateur

4.2.1.1. Inscription

L'utilisateur doit pouvoir s'inscrire avec les possibilités suivantes :

- Email / mot de passe
- GitHub
- Facebook
- Google

Un email de confirmation d'inscription doit être envoyé à l'utilisateur afin de valider son compte.

Dans le cadre de la connexion avec un tier comme GitHub, Facebook, Google, le compte utilisateur sera déjà rempli par son image de profil, son nom, son prénom, et son nom d'utilisateur (dans le cas de disponibilité de ces derniers).

Si une erreur survient lors de l'inscription, un message apparaît sur la page d'inscription et indique la nature de l'erreur.

4.2.1.2. Connexion

L'utilisateur doit pouvoir se connecter avec les possibilités suivantes :

- Email / mot de passe
- GitHub
- Facebook
- Google

Une fois connecté, l'utilisateur doit être redirigé vers sa page profil, dans le cas contraire, un message d'erreur apparaît sur la page de connexion, et indique la nature de l'erreur.

4.2.2. 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.

4.2.3. 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.

4.2.4. 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.

4.2.5. Module commentaire

Non accessible aux utilisateurs non identifiés.

Un utilisateur doit pouvoir commenter un snippet afin de pouvoir y laisser son avis, favorable ou non. Les commentaires d'un snippet seront affichés en bas de ce dernier, sous forme de pagination, avec pour maximum 20 commentaires par page.

4.3. Calendaires

Lancement du projet : Avril 2018 (T0)

Livraison planning : T0 + 2 semaines

Livraison spécifications : T0 + 1 mois

Livraison plan de tests : T0 + 4,5 mois

Livraison pour recette : T0 + 5 mois

Mise en production : T0 + 6 mois

Réunion de suivi de projet bi-mensuel.

4.4. Budgétaire

Nous disposons d'une marge budgétaire maximale de 60 000€ pour ce projet.

Le paiement sera effectué comme suit :

- 30% au lancement du projet
- 25% à livraison des spécifications
- 25% à livraison pour tests de recettes
- 20% à la mise en production

Une période de garantie de 6 mois est souhaitée, à l'issue de cette période un contrat de maintenance pourra être établis.

5. Prestations attendues

5.1. Développement

L'ensemble du développement fera partie intégrante de la prestation.

5.2. Hébergement

Nous désirons qu'une analyse concernant l'hébergement soit proposée dans la prestation, une fois le choix effectué, l'installation sera comprise dans la prestation.

5.3. Référencement

Le référencement fera parti intégrante de la prestation, l'objectif étant que le site soit rapidement référencer sur les moteurs de recherches principaux et de manière efficace.

5.4. Nom de domaine

L'achat du nom de domaine ne fera pas parti de la prestation, cependant la configuration et le déploiement de ce dernier seront inclus.

5.5. Mises à jour

Il est attendu une plateforme d'administration autonome, afin de pouvoir modérer tout type de contenu du site, les snippets, les commentaires, les likes et les comptes utilisateurs.

Il est aussi demandé un contrat de maintenance suite à la mise en production, afin de pouvoir effectuer des correctifs et des évolutions.



6. Matrice de conformité

6.1. Module et fonctionnalités

ID Module	Module	Priorité module	Id Fonctionnalité	Fonctionnalité
M01	Utilisateurs	1	M01-F01	Inscription
			M01-F02	Connexion
			M01-F03	Récupération des informations
M02	Snippets	1	M02-F01	Ajouter un snippet
			M02-F02	Editer un snippet
			M02-F03	Supprimer un snippet
			M02-F04	Visualiser un snippet
M03	Notifications	1	M03-F01	Recevoir une notification
			M03-F02	Lire une notification
			M03-F03	Mail d'information
M04	Profil	2	M04-F01	Editer les informations
			M04-F02	Visualiser les snippets créés
M05	Commentaire	2	M05-F01	Ajouter un commentaire
			M05-F02	Editer un commentaire
			M05-F03	Supprimer un commentaire

6.2. Contraintes

Id Contrainte	Contrainte
CO-01	Disponibilité 24/7
CO-02	Application "Single Page Application"
CO-03	Déploiement serveur linux

Annexe x : Curriculum Vitae



Julien Sergent
Développeur web
22 ans, né le 18 Septembre 1995

À propos
Ingénieur logiciel.
Artisan web.
Passionné d'intelligence artificielle.
Je dispose du permis et d'un véhicule,
avec une mobilité totale.

 MadDeveloper
 juliensergent.com

Contact
 65 rue Ernest Renan
31200 Toulouse
 sergent.julien@icloud.com
 06 45 53 24 90

Compétences
 Javascript
 Typescript
 Node.js
 PHP
 HTML
 CSS
 Angular
 Firebase



Expériences professionnelles

Développeur web fullstack

 Liberty Rider  Octobre 2017 - Aujourd'hui

Participation au développement de la nouvelle version de détection d'accident. La construction de la version nécessite la reconstruction de nouvelles API ainsi que l'adaptation de certains visuels.

Technologies : TypeScript, JavaScript, Node, GraphQL, Git, Firebase, Amazon Web Services, Docker, Angular, React, CircleCI, SASS

Développeur JavaScript

 Capgemini  Novembre 2016 - Février 2017

Développeur sur le projet MyCFM, un portail web pour Safran Power Units..

Ma mission était de maintenir et faire évoluer le portail selon les demandes du client.

Technologies : JavaScript, Git, SASS

Développeur React

 Capgemini  Mars - Septembre 2017

Développeur sur le projet Services by Airbus.

Ma mission était de travailler avec une équipe de consultants, de discuter et de proposer des solutions adaptées aux fonctionnalités demandées.

Technologies : React, Redux, Webpack, Node, Amazon Web Services, Git, Elastic Search, SASS

Développeur web fullstack

 Prep'App  Novembre 2014 - Août 2016

Création d'une startup ayant pour projet la réalisation d'une plateforme web et mobile permettant une gestion plus moderne des cours et révisions.

Technologies : PHP, Javascript, MySQL



Études

Mastere Chef de projet en ingénierie logicielle

 IPI  2016 - 2018

Licence développement informatique

 SUPINFO  2013 - 2016

Baccalauréat scientifique

 Lycée Clément Marot  2011 - 2013



Langues



Anglais



Français

