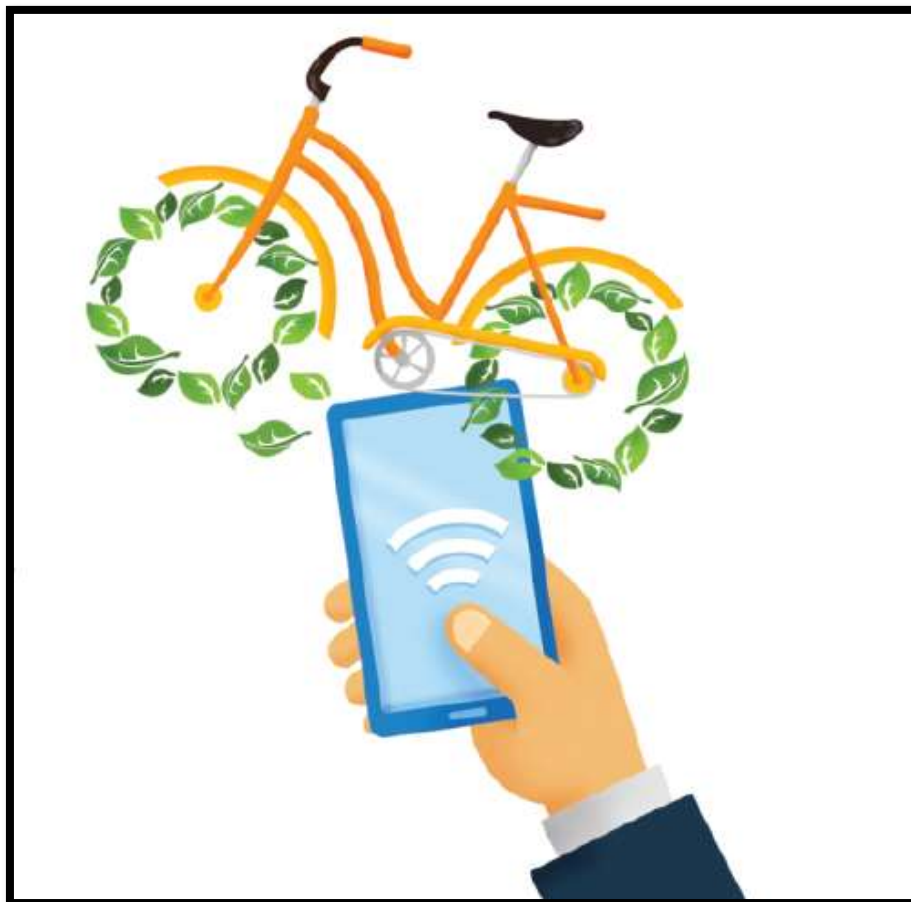


Travail de Bachelor 2019

Outil de maintenance de flotte de vélos en libre-service



Étudiant : Quentin Remion

Professeur : Yann Bocchi

Déposé le : 31 juillet 2019

Source de l'illustration de la page de titre

http://www.chinadaily.com.cn/opinion/2017-03/11/content_28517491.htm

Résumé

Inspirée d'une pensée plus écologiste et en accord avec notre temps, l'utilisation du vélo est revenue à la mode. De nombreuses villes se mettent aux énergies vertes et mettent à disposition des vélos en libre-service. C'est dans cette optique que le thème de ce projet a été initié. Le but étant de créer une application de vélos en libre-service.

Avant de développer ce système, il a été effectué un état de l'art des systèmes déjà existants ainsi que des cadenas connectés correspondant à nos besoins. Mais pour donner suite à cette analyse, il a été constaté qu'aucun cadenas ne correspondait à nos besoins et qu'il fallait partir dans une autre direction. C'est de là qu'a émergé l'idée de créer un outil de maintenance de vélos en libre-service, ce thème correspondant à un réel besoin du marché actuel.

Pour donner suite à cette décision, nous avons effectué une analyse des technologies correspondant à notre projet. Il a été déterminé que la création d'un site internet en Hypertext Preprocessor (PHP) et en utilisant le framework Laravel serait en adéquation avec nos besoins.

En utilisant ces technologies, nous avons pu concevoir un site internet qui englobe tous les besoins en la matière. Ce site permet à des utilisateurs de signaler un vélo qui a besoin d'une réparation directement sur le site ou en l'intégrant directement alors logiciel via une Application Programming Interface (API). Depuis ce site, il est également possible aux employés de voir tous les signalements des utilisateurs et de pouvoir effectuer une réparation.

Mots-clés : vélos en libre-service, système de maintenance, réparation de vélos.

Avant-propos

Ce travail a été effectué dans le cadre d'un travail de Bachelor de la Haute École spécialisée de Suisse occidentale (HES-SO). Durant la dernière année du cursus scolaire, chaque étudiant de la filière informatique de gestion doit effectuer un travail de Bachelor. Ce travail est donc effectué sur une plage de 3 mois qui se clôt par une défense orale ultérieure.

L'inspirateur du thème de ce travail de Bachelor est Gaël Ribordy. Sa passion pour le sport et plus particulièrement pour le vélo l'a poussé à lancer sa propre entreprise KargoBike. Cette entreprise fournit des services de livraison de colis par vélo. Mais son but principal est de « Créer un contexte permettant la mobilité durable en milieu urbain » (KargoBike, s.d.).

Au fil du temps, il a pu constater qu'il y avait un réel manque en Suisse, et principalement en Valais, de vélos en libre-service. Il a constaté qu'il existe un potentiel dans la région qui n'était pas encore exploité par les solutions actuelles de vélos en libre-service. C'est donc à partir de ce constat que Gaël Ribordy a proposé son idée à la Filière informatique de gestion avec l'aide de Yann Bocchi (professeur à la HES-SO) afin qu'elle soit proposée comme thème de Bachelor.

Le but de ce projet était dans un premier temps de créer une application de vélos en libre-service, mais suite à plusieurs constats, il a fallu réorienter le thème du travail de Bachelor pour créer un site internet de maintenance de vélos en libre-service. Ce thème correspond alors à une carence considérable dans le domaine et permet d'apporter des solutions innovantes aux entreprises existantes de vélos en libre-service.

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé durant la rédaction de mon travail de Bachelor. Un merci tous particuliers aux personnes suivantes :

M. Yann Bocchi, mon référent pour ce travail de Bachelor, pour le temps qu'il a consacré à m'aiguiller dans mon travail. Par son expérience en tant que référent, il m'a donné de nombreux et précieux conseils qui m'ont évité de perdre un temps considérable. Un grand merci à lui d'avoir assuré le suivi de mon travail jusqu'à son aboutissement.

M. Gaël Ribordy, l'inspirateur de ce projet, qui m'a permis, par sa connaissance et son expérience dans le domaine de la mobilité durable, de voir plus clair dans ce secteur d'avenir. Merci à lui pour tout le temps qu'il a mis à disposition à m'aider et m'éclairer sur le chemin à suivre.

Ma famille pour sa patience, pour son soutien et pour la relecture de mon travail.

Mes camarades de classe pour leur bonne humeur et pour leur collaboration durant toute la formation.

Table des matières

Table des figures.....	x
Table des tableaux.....	xiii
Glossaire	xiv
Liste des abréviations	xvi
Introduction.....	1
1. Méthodologie de travail.....	3
1.1. Gestion du travail.....	3
1.1.1. Travail par itération.....	4
1.1.2. Product Backlog.....	5
1.2. Méthodologie de l'état de l'art.....	5
2. Vélos en libre-service	6
2.1. Concept	6
2.2. Histoire.....	7
3. État de l'art.....	9
3.1. Systèmes existants.....	9
3.1.1. PubliBike.....	10
3.1.2. Mobike.....	11
3.1.3. Ofo	11
3.1.4. Valaisroule.....	12
3.1.5. Donkey Republic.....	12
3.1.6. AirBie	13
3.1.7. Tableau comparatif des systèmes existants	14
3.2. Cadenas.....	16
3.2.1. NOKĒ.....	16

3.2.2. Linka	16
3.2.3. OTO Hunter	18
3.2.4. Tableau comparatif des cadenas.....	19
4. Décisions au sujet des cadenas	20
4.1. Cadenas.....	20
4.1.1. OTO Hunter	20
4.1.2. Linka Leo.....	21
4.1.3. Original Linka.....	21
4.1.4. Conclusion du choix de cadenas	22
4.2. Décision finale.....	22
5. Analyse des technologies	24
5.1. Programmation d'applications mobiles.....	25
5.1.1. Analyse des systèmes d'exploitation pour mobiles	25
5.1.2. Décision au sujet des systèmes d'exploitation pour mobiles	27
5.2. Programmation Web.....	27
5.2.1. JavaScript - Node.js	28
5.2.2. Python – Django	29
5.2.3. PHP – Laravel.....	30
5.2.4. Comparaison des langages.....	33
5.2.5. Choix du langage de programmation.....	34
5.3. Base de données	35
5.4. Outils de développement	36
5.4.1. Xampp.....	36
5.4.2. Apache.....	36
5.4.3. Eloquent	37

5.4.4. Leaflet avec OpenStreetMap	37
5.4.5. Laravel API	37
5.4.6. SwiftMailer Laravel.....	38
5.4.7. Markdown	38
5.4.8. Bootstrap.....	38
5.4.9. Flash Messages.....	39
5.4.10. Laravel Excel	39
6. Développement.....	39
6.1. Mockup	39
6.1.1. Page d'accueil.....	40
6.1.2. Page de détail de signalement	41
6.1.3. Page de réparation	42
6.2. Guide technique.....	43
6.2.1. Version des technologies	43
6.2.2. Arborescence du projet.....	45
6.2.3. Configuration du fichier «.env».....	46
6.2.4. Schéma de la base de données	47
6.2.5. Création de la base de données	50
6.2.6. Modèles.....	52
6.2.7. Contrôleurs.....	53
6.2.8. Vues	56
6.2.9. Routage	57
6.2.10. Carte Leaflet	58
6.2.11. Envoi de courriel.....	60
6.2.12. Exporter en Excel.....	62

6.2.13. API	63
6.2.14. Test Unitaire	64
6.3. Guide d'utilisateur	65
6.3.1. Connexion.....	65
6.3.2. Accueil	66
6.3.3. Détails d'un signalement.....	67
6.3.4. Réparation	68
6.3.5. Signalement.....	69
6.3.6. Administration.....	70
7. Bilan	71
7.1. Difficultés rencontrées.....	71
7.2. Améliorations futures	71
7.2.1. Statistique du site internet.....	71
7.2.2. Facturation automatique	72
7.2.3. Optimisation du formulaire de signalisation.....	72
7.2.4. Ouverture vers un marché de vélos privés	72
Conclusion	73
Références	74
Références des figures.....	80
Annexes I : Mockups du premier système de BikeSharing	84
Annexes II : Product Backlog du premier système	87
Annexes III : Product Backlog du second système	89
Annexes IV : Livre de bord	90
8. Déclaration de l'auteur.....	91

Table des figures

Figure 1 : Agile vs Waterfall	3
Figure 2 : structure des itérations avec Scrum	4
Figure 3 : structure d'une User Story	5
Figure 4 : le mouvement Provo qui fait de la publicité pour les Vélos blancs à Amsterdam	7
Figure 5 : vélo jaune de La Rochelle de 1976	7
Figure 6 : carte de Suisse des systèmes de vélos en libre-service existants, qui ont existé et qui sont en projet.....	8
Figure 7 : carte du monde des systèmes de vélos en libre-service existants, qui ont existé et qui sont en projet.....	9
Figure 8 : logo PubliBike	10
Figure 9 : logo Mobike	11
Figure 10 : logo Ofo	11
Figure 11 : logo Valaisroule	12
Figure 12 : logo Donkey Republic	12
Figure 13 : logo AirBie.....	13
Figure 14 : cadenas pour vélo NOKÉ	16
Figure 15 : cadenas LINKA Leo.....	17
Figure 16 : cadenas OTO Hunter	18
Figure 17 : cadenas connecté Airbie	23
Figure 18 : schéma explicatif du fonctionnement du système de maintenance	24
Figure 20 : logo Android	25
Figure 19 : logo iOS.....	25
Figure 21 : statistique de janvier 2017 de ApplInstitute sur la part de marché des différents OS pour smartphones dans le monde	25
Figure 22 : statistique de StatCounter sur la part de marché des différents OS pour smartphones en Suisse	26
Figure 23 : logo Node.js	28
Figure 24 : logo JavaScript	28
Figure 25 : logo Python.....	29

Figure 26 : logo Django	29
Figure 27 : logo PHP.....	31
Figure 28 : logo Laravel.....	31
Figure 29 : logo MySQL.....	35
Figure 30 : logo XAMPP	36
Figure 31 : logo Apache	36
Figure 32 : logo Leaflet	37
Figure 33 : logo OpenStreetMap	37
Figure 34 : logo SwiftMailer.....	38
Figure 35 : logo Markdown.....	38
Figure 36 : logo Bootstrap	38
Figure 37 : Mockup de la page d'accueil	40
Figure 38 : Mockup de la page de détail d'un signalement	41
Figure 39 : Mockup de la page de réparation d'un vélo	42
Figure 40 : arborescence des fichiers Laravel.....	45
Figure 41 : schéma de la base de données.....	48
Figure 42 : fichier de migrations dans Laravel.....	50
Figure 43 : exemple de fichier de migrations pour la table Réparation.....	51
Figure 44 : exemple de fichier Seeder	52
Figure 45 : exemple du modèle Reparation	52
Figure 46 : méthode de connexion sur le site	53
Figure 47 : requête pour récupérer tous les signalements	53
Figure 48 : méthode pour la recherche de signalement	54
Figure 49 : requêtes de la page de détail d'un signalement	54
Figure 50 : requête d'insertion d'une réparation.....	55
Figure 51 : requête de filtre pour la page administrateur	56
Figure 52 : structure des fichiers vues.....	56
Figure 53 : routage du site internet.....	57
Figure 54 : structure du code de configuration d'une carte	58
Figure 55 : obtenir la géolocalisation d'un utilisateur.....	59
Figure 56 : exemple de courriel envoyé depuis Laravel	60

Figure 57 : configuration de courriel dans le fichier .env.....	60
Figure 58 : autorisation d'accéder à la messagerie Gmail via des applications externes...	61
Figure 59 : code de création d'un courriel dans un contrôleur.....	61
Figure 60 : contrôleur de courriel.....	61
Figure 61 : contenu du courriel créé en Markdown.....	62
Figure 62 : code pour exporter des données dans un fichier Excel	62
Figure 63 : exemple de contrôleur d'exportation	62
Figure 64 : route pour l'appel de l'API.....	63
Figure 65 : exemple d'utilisation de l'API.....	64
Figure 66 : tests unitaires pour la page administrateur	65
Figure 67: page d'accueil du site	66
Figure 68 : page de détails d'un signalement.....	67
Figure 69 : page de réparation	68
Figure 70 : formulaire de réparation	69
Figure 71 : page d'administration du site.....	70
Figure 72 : popup de réservation d'un vélo	84
Figure 73 : page d'accueil de l'application de vélos en libre-service	84
Figure 74 : début d'un trajet.....	85
Figure 75 : localisation du vélo réservé	85
Figure 76 : fin de trajet	86
Figure 77 : trajet en cours.....	86
Figure 78 : Product Backlog du premier système – Partie 1	87
Figure 79 : Product Backlog du premier système – Partie 2	88
Figure 80 : Product Backlog du second système (outil de maintenance)	89

Table des tableaux

Tableau 1 : comparaison des systèmes existants.....	14
Tableau 2 : comparaison des différents cadenas	19
Tableau 3 : choix du cadenas.....	22
Tableau 4 : comparatif des OS mobiles	26
Tableau 5 : comparaison des langages informatiques - Framework.....	33
Tableau 6 : comparaison pour le choix du langage de programmation	35
Tableau 7 : version des technologies présentes dans le projet	43
Tableau 8 : variable du fichier de configuration «.env».....	46
Tableau 9 : détails de la table Utilisateur	48
Tableau 10 : détails de la table Signalement.....	49
Tableau 11 : détails de la table Réparation	50

Glossaire

API :	l'interface de programmation d'application (en anglais : Application Programming Interface) est une solution qui permet un échange de services ou de données entre des applications/systèmes. Cela permet un accès simplifié à des données (Interface de programmation : API ou Application Programming Interface, 2019).
Backend :	le backend est la partie non visible du système informatique. C'est toute la logique qui fait marcher le système, il fait notamment le lien entre la base de données, le serveur et le site internet (Développement front-end et back-end : Quelles différences?, 2018).
Framework :	outil qui facilite la création de site internet en fournissant des composants et des fonctionnalités qui peuvent être directement réutilisés pour la création d'un site. Il permet un gain de temps et donc un gain d'argent pour le développement d'un site internet (Qu'est-ce que Django?, s.d.).
Frontend :	le frontend est tout ce qui concerne la partie visible par un utilisateur d'une interface. Donc, dans la même logique, un développeur frontend sera celui qui fait des maquettes de l'aspect du site internet et c'est celui qui créera tout le code pour la création visuelle de l'interface (Quelle est la différence entre développement back end, front end, et full stack ?, s.d.).
Mockup :	un Mockup est une maquette d'une interface utilisateur (Mockup, s.d.).
MoSCoW :	méthode pour prioriser les User Stories (Must, Should, Could, Would).

Open source :	« Un logiciel Open Source est un programme informatique dont le code source est distribué sous une licence permettant à quiconque de lire, modifier ou redistribuer ce logiciel. » (Dabischwebel, 2014)
Product Backlog :	le Product Backlog est un artéfact de Scrum sous forme de tableau permettant de prioriser les User Story.
Scrum :	Scrum est une méthode de gestion de projet faisant partie de Agile. Son but est d'améliorer la productivité d'une équipe.
Story Point :	le Story Point est une estimation de l'effort à fournir pour effectuer une tâche (Sciara & Cardoso, 2018).
User Story :	les User Stories sont des phrases qui décrivent une des fonctionnalités d'une application.

Liste des abréviations

HES-SO	Haute École spécialisée de Suisse occidentale
PV	Procès-Verbal
EPFL	École polytechnique fédérale de Lausanne
API	Application Programming Interface
QR Code	Quick Response Code
OS	Operating System
JS	JavaScript
MVC	Model View Controller
URL	Uniform Resource Locator
PHP	Hypertext Preprocessor
CSRF	Cross-Site Request Forgery
HTML	Hypertext Markup Language
REST	Representational State Transfer
JSON	JavaScript Object Notation
SQL	Structured Query Language
XAMPP	X (cross) Apache MariaDB Perl PHP
ORM	Object-Relational Mapping
HTTP	Hypertext Transfer Protocol
CSV	Comma-Separated Values
AJAX	Asynchronous JavaScript and XML
RF	Responsable de filière
s.d.	Sans date

Introduction

L'utilisation du vélo a fait depuis quelques années un bond exceptionnel ; en effet ce moyen de transport revient à la mode dans de nombreuses villes du monde. La société actuelle essaie de faire face à des réalités écologiques alarmantes et un engorgement des réseaux routiers. C'est dans cette optique-là que de nombreuses villes du monde essaient de promouvoir l'utilisation de vélos en mettant notamment à disposition de la population des pistes cyclables et des systèmes de vélos en libre-service.

Notre travail s'est déroulé en quatre phases bien distinctes. La première phase était celle de l'analyse du contexte actuel, donc la réalisation d'un état de l'art. Dans cette partie, nous avons analysé les systèmes existants pour voir l'état du marché actuel et nous avons également analysé les cadenas connectés qui existaient sur le marché. Trouver un cadenas était un des objectifs principaux du projet car son but était de connecter un cadenas à un système de vélos en libre-service.

La seconde phase consistait à prendre une décision sur l'avenir du projet. En effet, après l'analyse des cadenas et des systèmes existants, on est arrivé à la conclusion qu'aucun cadenas ne correspondait à nos besoins et que de nombreux systèmes existants pouvaient correspondre au besoin de ce projet. C'est donc dans cette phase que le thème du projet a changé lorsqu'on a constaté qu'il y avait une carence en matière d'outil de maintenance pour ces vélos.

La troisième phase était celle de l'analyse des technologies. En effet après avoir déterminé sur quel système partir, il faut décider de la technologie la plus efficace en fonction de nos besoins. Nous avons dans un premier temps déterminé s'il était plus judicieux de créer une application mobile ou un site internet ; c'est la seconde option qui a été choisie pour des raisons de temps et de compatibilité. Puis il a fallu donc déterminer quelles technologies choisir pour programmer un site internet ; cette analyse s'est principalement basée sur les besoins de notre projet.

La quatrième et dernière phase fut celle du développement. Cette phase a commencé par la création de différents Mockups et d'un schéma de la base de données. Puis il a fallu mettre

en place l'environnement de travail afin que celui-ci soit prêt pour commencer le développement du site.

Tout au long de notre travail, une question est revenue à de nombreuses reprises, chaque fois qu'une décision importante apparaissait au sujet du choix du système à devoir développer. Cette problématique était la suivante : quels sont les besoins actuels des systèmes de vélos en libre-service ?

1. Méthodologie de travail

1.1. Gestion du travail

Nous avons appliqué une méthodologie de travail très proche de la méthodologie Scrum, faisant partie d'Agile. Le fait étant que certains points, intéressants lors d'un travail d'équipe avec la méthodologie Scrum, n'ont aucun sens de se retrouver dans un travail individuel, tel que le « Daily Meeting » par exemple. On a donc mis en place un système avec itération : des séances régulières avec le client pour montrer l'avancement du travail.

Le choix de l'utilisation de Scrum s'est fait assez facilement. À la différence de la méthodologie Waterfall qui se veut plus rigide par l'obligation de finir une tâche avant de passer à une suivante, Scrum se veut beaucoup plus flexible en instaurant des itérations sous forme de petit projet et, à la fin de chacune de celles-ci, on réévalue les tâches pour la prochaine itération. Ceci permet de chaque fois pouvoir adapter le projet en fonction des besoins du client qui, dans la plupart des cas, évoluent ou changent au cours du temps.

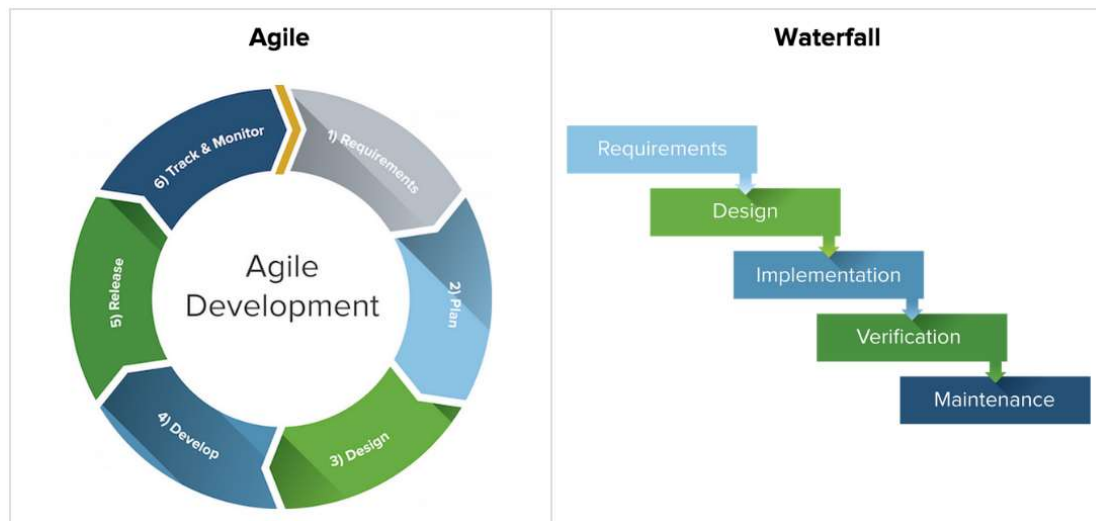


Figure 1 : Agile vs Waterfall

1.1.1. Travail par itération

Pour la conception du travail, nous avons choisi d'effectuer des itérations d'environ 10 jours. Ceci permet de laisser un certain temps pour développer des fonctionnalités entre chaque itération tout en ayant des rendez-vous réguliers pour présenter l'avancement du travail.

A chaque fin d'itération est prévue une séance avec Yann Bocchi et Gaël Ribordy afin de voir l'avancement du projet. Les tâches suivantes sont effectuées durant les séances :

- une rétrospective de l'itération afin de voir quels ont été les problèmes rencontrés, ce qui s'est bien passé et ce qui peut être amélioré,
- une présentation du travail effectué durant le sprint (itération),
- validation ou rejet des tâches qui étaient attribuées pour l'itération terminée,
- réévaluation du Product Backlog et de la priorisation des User Stories,
- attribution des tâches pour l'itération à venir.

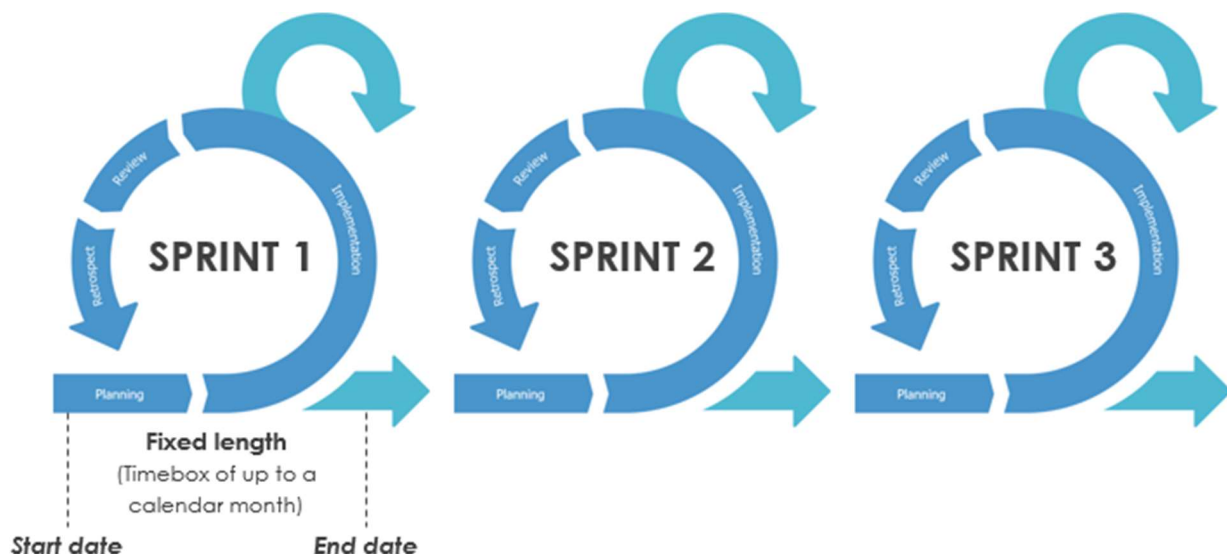


Figure 2 : structure des itérations avec Scrum

À la fin de chaque séance, un procès-verbal (PV) est envoyé à toutes les personnes concernées.

1.1.2. Product Backlog

La méthodologie Scrum implique l'utilisation d'un Product Backlog. Celui-ci aborde tous les besoins du projet sous forme d'User Stories. Il permet également d'indiquer des critères de reconnaissance pour chaque User Story, leur priorité et leur importance sous forme de MoSCoW (Must, Should, Could, Would). Ce sont ces deux derniers points et plus particulièrement le degré de priorité des User Stories qui vont déterminer dans quel ordre les tâches seront effectuées durant le projet.



Figure 3 : structure d'une User Story

A la fin de chaque itération, le Product Backlog est réévalué. Le client peut modifier les champs du tableau afin que ce dernier soit le plus représentatif de ses besoins. Cela permet une certaine souplesse avec Scrum que la méthodologie Waterfall n'aurait pas permise.

Le Product Backlog permet également de déterminer quelle User Story a été validée (ce qui veut dire que la tâche est terminée en respectant tous les critères d'acceptation) ou rejetée (pas entièrement terminée ou ne correspondant pas ou plus aux besoins et à la demande du client). Dans le cas où une User Story est rejetée, elle doit être indiquée comme telle et on doit créer une nouvelle User Story afin de pouvoir y travailler à nouveau.

1.2. Méthodologie de l'état de l'art

Les observations de l'état de l'art se concentrent sur les systèmes déjà existants de vélos en libre circulation et sur les différents cadenas connectés présents sur le marché.

Le client, inspirateur du projet, par sa connaissance du domaine, m'a orienté pour ma recherche. Il m'a notamment transmis toutes les recherches qu'il avait effectuées au préalable. Après la phase de recherche, les systèmes et les cadenas les plus pertinents ont été analysés plus précisément dans l'état de l'art.

2. Vélos en libre-service

2.1. Concept

Comme le dit le rapport de Transitec Ingénieurs-conseils SA :

Le principe consiste à emprunter un vélo pour une courte durée (inférieure à 1 heure en général), **de manière automatique, donc sans assistance humaine** (contrairement à la location) **et de pouvoir le rendre dans un lieu différent de celui de la prise** (à son lieu de destination, duquel on repartira plus tard avec un autre vélo ou avec autre moyen de transport). (Vélos en libre-service en Suisse: harmonisation des systèmes d'accès, 2009, p. 10)

Le but étant, à la différence des box de location de vélos, que les vélos soient disponibles 24 heures sur 24 et 7 jours sur 7 ; tout en ne contraignant pas l'utilisateur à devoir ramener un vélo au même endroit que lors de sa location. On le considère souvent comme un complément au service de transport public, parce qu'il permet d'accéder à des lieux que les transports publics ne couvrent pas. « Il accentue l'attractivité et l'effet "porte à porte" des transports publics en offrant une solution de déplacement pour le dernier kilomètre à parcourir quand on arrive à la gare. » (Pro Vélo Jura, 2013)

Un autre aspect, que cherche à promouvoir les vélos en libre-service, est celui du développement durable. Comme le dit Pro Vélo Jura, c'est « un moyen de transport non polluant, silencieux, convivial et qui est bon pour la santé » (Vélos en libre service (VLS) - Questionnaire 2013, 2013). De plus, c'est un moyen de transport idéal pour les villes, il permet de désengorger le trafic et d'offrir aux utilisateurs, dans certain cas, la possibilité de gagner du temps en évitant les bouchons créés par le trafic des véhicules.

2.2. Histoire



Figure 4 : le mouvement Provo qui fait de la publicité pour les Vélos blancs à Amsterdam

C'est en 1965 à Amsterdam aux Pays-Bas qu'est développé le premier concept de vélos en libre-service. Il a alors été lancé dans le but de diminuer le trafic dans le centre-ville d'Amsterdam. Le projet s'appelait « Vélo blanc » ; son but était de proposer à tout le monde de peindre son vélos en blanc et de le déposer dans la rue afin de le rendre disponible à tous.

Ce projet a rapidement été un échec : en effet, alors que les vélos étaient dépourvus de cadenas, tous les vélos ont été volés sur une courte période. Mais malgré cet échec, un nouveau concept était né et n'attendait qu'à être développé (de Préval, 2017).

C'est seulement en 1976, à La Rochelle en France, que l'idée de vélos en libre-service renaît. L'administration de la ville met à disposition 250 vélos jaunes nommé « Yélo ». Tous les vélos étaient proposés gratuitement et n'avaient aucun système d'anti-vol ; la ville avait bien souligné la responsabilité de chacun pour l'entretien des vélos. Cette solidarité a permis de faire vivre ce système à travers le temps et l'on peut encore voir ces fameux vélos jaunes dans la ville de La Rochelle ainsi que dans d'autres villes de France (Cadeau, 2018).



Figure 5 : vélo jaune de La Rochelle de 1976

Différents systèmes de vélos en libre-service se développent en Europe par la suite, comme à Copenhague au Danemark en 1995 et un peu plus tard au Royaume-Uni ainsi que dans plusieurs villes des Pays-Bas (de Préval, 2017). C'est en 1998 qu'a lieu une petite révolution avec le premier système informatique installé à Rennes en France ; ce système intègre l'identification de l'utilisateur (Intermodality SA, 2016). C'est seulement en 2000 que le concept s'étend en dehors de l'Europe, en s'installant à Singapour (de Préval, 2017).

En Suisse, il a fallu attendre 2009 pour voir apparaître le premier système de vélos en libre-service. C'est Vélopass – qui fut racheté plus tard par CarPostal Suisse pour devenir PubliBike (Viennet, 2012) – qui développa ce premier système à Lausanne sur le campus de l'École polytechnique fédérale de Lausanne (EPFL) (Intermobility SA, 2016). Par la suite, de nombreux systèmes différents se sont développés dans la plupart des villes de Suisse comme peut l'attester la carte ci-dessous (The Bike-sharing World Map, 2019).

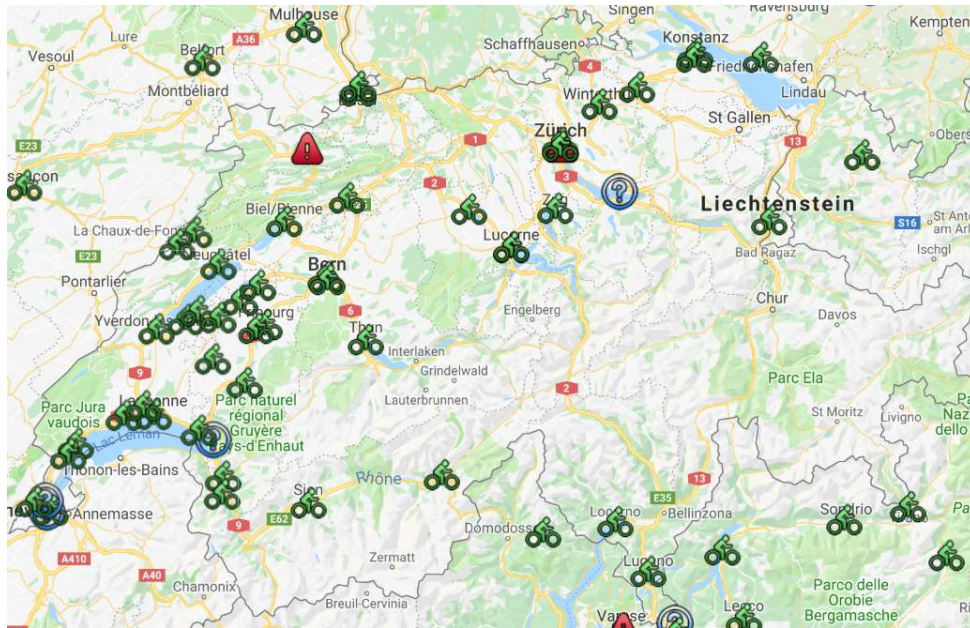


Figure 6 : carte de Suisse des systèmes de vélos en libre-service existants, qui ont existé et qui sont en projet

On peut voir sur la carte suivante que les vélos en libre-service se développent sur tous les continents ; c'est un système qui continue à s'étendre et à couvrir toujours une plus grande surface de la terre (The Bike-sharing World Map, 2019).



Figure 7 : carte du monde des systèmes de vélos en libre-service existants, qui ont existé et qui sont en projet

3. État de l'art

3.1. Systèmes existants

Dans un premier temps, nous analysons dans l'état de l'art les systèmes de vélos en libre-service déjà existants. Cette analyse est basée sur une recherche d'information sur internet, un test de l'application ainsi que, dans certains cas, sur un contact direct avec l'entreprise.

Cette étude nous permet d'avoir une vue globale du marché des vélos en libre-service en Suisse mais aussi dans le monde. De plus, il nous permet d'identifier les points positifs et négatifs de chaque système, ce qui nous permettrait dans le futur de développer un système qui correspond mieux aux besoins du marché et donc des utilisateurs.

3.1.1. PubliBike

PubliBike est un système public de partage de vélos basé en Suisse. Il propose ses services via un site internet et une application mobile. Il met à disposition des vélos et des vélos électriques dans des lieux de stationnement précis appelés « Stations PubliBike ».



Figure 8 : logo PubliBike

Sur le site internet et sur l'application, il est possible de visualiser la carte afin de voir où se situe chaque station PubliBike et ainsi de savoir en temps réel combien de vélos sont disponibles dans chaque station (PubliBike, 2019).

Une fois à la station PubliBike, l'utilisateur peut alors choisir son vélo. Dès qu'il l'a choisi, il peut déverrouiller le cadenas connecté à l'aide de l'application ou du *SwissPass* (carte regroupant plusieurs services de mobilité dont notamment les abonnements des CFF). Durant le trajet, il est possible de faire plusieurs arrêts et d'utiliser le cadenas pour sécuriser le vélo. Une fois le trajet fini, le vélo doit être ramené à une station PubliBike (PubliBike, 2019).

Le paiement du trajet s'effectue automatiquement à la fin du voyage. Il existe plusieurs tarifs : celui des vélos simples est de Fr. 3.– pour les trente premières minutes et Fr. 0.05 pour chaque minute supplémentaire ; tandis que pour les vélos électriques, c'est Fr. 4.50 pour la première demi-heure puis Fr. 0.10 pour chaque minute supplémentaire. Il existe également un système d'abonnement à Fr. 9.– par mois ou Fr. 60.– par année (PubliBike, 2019).

L'application intègre également un système pour signaler si un vélo est endommagé.

3.1.2. Mobike

C'est une entreprise chinoise qui créa cette application en 2015. C'est le plus grand service au monde de vélos partagés en libre-service. Ce qui est spécifique à cette application, c'est qu'il n'existe pas de station de dépôt de vélo. En effet, l'utilisateur peut déposer le vélo où il le souhaite mais en respectant certaines règles notamment celles de ne pas encombrer le trottoir ou gêner la circulation (Mobike, 2018).



Figure 9 : logo Mobike

Son utilisation est simple : tu réserves à l'avance ton vélo dans le lieu que tu désires et tu le verras alors s'afficher sur la carte de l'application afin de pouvoir le récupérer. Pour l'utiliser, il suffit de scanner le Quick Response Code (QR Code) présent sur le vélo afin de le déverrouiller et commencer son trajet. Les tarifs Mobike sont calculés sur le temps d'utilisation des vélos (Lee, 2016).

3.1.3. Ofo

Ofo est aussi une entreprise chinoise, créée en 2014. L'application se base, pour sa plus grande part, sur le même fonctionnement que Mobike (voir ci-dessus). Sa seule différence majeure est qu'Ofo a été imaginée par des étudiants pour des étudiants, ce qui implique qu'Ofo utilise des vélos dit « bas de gamme » et que les prix sont nettement moins élevés (plus ou moins la moitié des tarifs de Mobike) (Soula, 2017).



Figure 10 : logo Ofo

3.1.4. Valaisroule

Cette entreprise valaisanne propose, comme son nom le suggère, une location de vélos en Valais. Son but principal est de promouvoir les paysages valaisans tout en favorisant l'utilisation écologique du vélo. Valaisroule met à disposition des vélos dit « standards » en prêt, totalement gratuits pour une utilisation maximale de quatre heures. Sinon il est possible de louer d'autres vélos tel que des VTT, des vélos électriques, etc. (ValaisRoule, s.d.)



Figure 11 : logo Valaisroule

Pour se procurer un vélo, il suffit d'aller dans une station Valaisroule est de le louer directement sur place. Il est alors possible de le ramener dans n'importe quelle station du Valais (ValaisRoule, s.d.).

Ce système se différencie des autres systèmes analysés dans ce travail parce qu'il ne permet pas une location 24 heures/24, il est uniquement possible de venir prendre un vélo lors des heures d'ouverture des box de location. Son analyse est tout de même intéressante par son aspect de promouvoir gratuitement l'utilisation de vélos pour découvrir le Valais.

3.1.5. Donkey Republic

Cette entreprise, basée à Copenhague, propose des vélos en libre-service. Afin de louer un vélo, il suffit d'aller sur l'application et de voir quelle est la station la plus proche, puis on peut sélectionner le vélo (une location maximale de cinq vélos). Une fois la réservation effectuée, il faut se rendre sur place et déverrouiller le cadenas du vélo avec son téléphone. Il est possible d'utiliser le vélo durant plusieurs heures et même jours, et donc de verrouiller et déverrouiller



Figure 12 : logo Donkey Republic

le cadenas du vélo lors des différents arrêts. Une fois le trajet terminé, il faut ramener le vélo dans une station (Donkey Republic, 2019).

Les tarifs évoluent de manière décroissante (plus on utilise un vélo, plus le tarif horaire diminue). Il est aussi possible de prendre des abonnements qui permettent d’avoir un nombre d’heures gratuites durant chaque location (Donkey Republic, 2019).

Ce système est utilisé en Suisse à Genève, Neuchâtel et au Locle (Donkey Republic, s.d.).

3.1.6. AirBie

AirBie est une entreprise Suisse créée en 2018. Elle propose un système de vélos en libre-service. Son système est opérationnel dans la ville de Zoug mais reste tout de même en version test (Systèmes en Suisse, s.d.).



Figure 13 : logo AirBie

Actuellement elle propose une application classique avec la possibilité de visualiser où sont les vélos. Elle utilise également un système de cadenas connectés utilisant le Bluetooth pour le déverrouiller. Le service est actuellement entièrement financé par la ville de Zoug et les vélos sont disponibles gratuitement. Ils proposent également une plateforme web afin que les responsables de la flotte de vélos puissent gérer son utilisation (Airbie, 2019).

Ils sont actuellement en train de poursuivre le développement de leur application avec notamment l’ajout d’un système de paiement en ligne.

3.1.7. Tableau comparatif des systèmes existants

Tableau 1 : comparaison des systèmes existants

	PubliBike	Mobike	Ofo
Application	Oui	Oui	Oui
Site Web	Oui	Non	Non
Location de vélo privé	Non	Non	Non
Location de vélo public	Oui	Oui	Oui
Place de stationnement définie	Oui	Non	Non
Moyen de signalisation de vélos endommagé	Oui	Oui	Oui
Moyen pour déverrouiller le vélo	Proximité entre le téléphone et le vélo (max. 20 cm avec Bluetooth) ou coller le <i>SwissPass</i> au cadenas	Scanner un QR Code	Scanner un QR Code
Cadenas connecté	Intégré au vélo	Intégré au vélo	Intégré au vélo
Région	Suisse	Plusieurs pays du monde (la Suisse n'en fait pas partie)	Plusieurs pays du monde (la Suisse n'en fait pas partie)

	Donkey Republic	Valaisroule	Airbie	<notre app>
Application	Oui	Non	Oui	Oui
Site Web	Non	Non	Non	Non
Location de vélo privé	Non	Non	Oui	Oui
Location de vélo public	Oui	Oui	Oui	Oui
Place de stationnement définie	Oui	Oui	Non	Non
Moyen de signalisation de vélo endommagé	Oui	Oui	Oui	Oui
Moyen pour déverrouiller le vélo	Proximité entre le téléphone et le cadenas (Bluetooth)	Location sur place	Proximité entre le téléphone et le cadenas (Bluetooth)	À définir
Cadenas connecté	Intégré au vélo	Pas de cadenas	Intégré au vélo	À définir
Région	Plusieurs pays d'Europe dont la Suisse	Valais	Suisse	Valais

3.2. Cadenas

L'un des objectifs principaux de ce travail de Bachelor était de trouver un cadenas connecté qui correspondrait, idéalement si possible, à nos besoins pour créer une application de vélos en libre-service. Cette analyse se base uniquement sur des informations disponibles sur internet.

Cette étude nous a permis de tirer un premier bilan pour déterminer quel cadenas nous semble correspondre à nos besoins. De plus, elle nous a aidé à identifier quel cadenas il est intéressant de commander pour approfondir les recherches en le testant directement.

3.2.1. NOKĒ

Ce cadenas se veut facile d'utilisation : il suffit d'être à moins de trois mètres du cadenas avec son smartphone pour qu'il se déverrouille. Et il n'est même pas utile de sortir son téléphone, une simple pression sur le cadenas permet de le déverrouiller. De plus, si le téléphone est inutilisable, il est possible d'utiliser un code en morse pour ouvrir le cadenas (Sebastien, 2016).

NOKĒ permet également, grâce à son application, de facilement donner l'accès au cadenas à une autre personne. Il est aussi aisé de localiser le cadenas depuis l'application, ainsi que d'avoir accès à l'historique des ouvertures. L'application vous préviendra également si quelqu'un tente d'ouvrir le cadenas sans autorisation (Nokē, 2019).

De plus, NOKĒ permet une intégration dite « facile et rapide » d'une API pour des sites internet et des applications iOS et Android (Nokē, 2019).



Figure 14 : cadenas pour vélo NOKĒ

3.2.2. Linka

Cette marque de cadenas connectés fournit des systèmes fixes à poser directement sur la structure du vélo. Leur utilisation reste très simple : il suffit de s'approcher à moins de deux mètres avec son téléphone sur soi pour que le cadenas se déverrouille. Pour le verrouiller, il

suffit juste d'appuyer sur un bouton qui se trouve dessus tout en ayant le téléphone à proximité (Linka, 2019).



Figure 15 : cadenas LINKA Leo

Il est possible de créer sa propre application en utilisant l'API de Linka ; elle permet d'intégrer toutes les options de l'application Linka GO à l'intérieur d'une autre application. Elle est compatible avec Android et iOS (Linka, 2019).

Afin d'utiliser Linka, il est important d'avoir une selle droite pour s'assurer que les différents cadenas soient solidement fixés aux vélos. Avec une structure de siège courbé ou non standard, ce n'est peut-être pas le meilleur ajustement et donc il se pourrait que le cadenas ne soit pas compatible (Linka, 2019).

Il y a deux modèles de base :

- Original Linka : ce cadenas intègre toutes les fonctionnalités de base de Linka. Il a également un système anti-vol qui se met à sonner lorsqu'il détecte un éventuel problème et envoie un message d'alerte à son propriétaire s'il se trouve à moins de 120 mètres (Linka, 2019).
- Linka Leo : il intègre également toutes les fonctionnalités Linka. En plus de cela, ce cadenas utilise les données internet afin d'être géolocalisable à tout moment. Son système anti-vol diffuse également une alarme et envoie un message à son propriétaire sans restriction de distance à la différence de l'Original Linka (Linka, 2019).

3.2.3. OTO Hunter

OTO Hunter est un cadenas créé par FOX-TECH CO. Sa grande force est qu'il est facilement intégrable au vélo et est compatible avec la plupart des vélos. Son utilisation est proche de celle des autres cadenas : il suffit d'être à proximité avec son téléphone pour que le cadenas s'ouvre (Fox Tech, 2019).

Ce cadenas a de nombreuses fonctionnalités, telles qu'un moyen de géolocaliser l'emplacement du vélo, un anti-vol et enfin un panneau solaire qui lui permet de ne pas avoir de problème de batterie trop faible. De plus, il permet l'utilisation d'une API pour créer nos propres applications (Fox Tech, 2019).



Figure 16 : cadenas OTO Hunter

3.2.4. Tableau comparatif des cadenas

Tableau 2 : comparaison des différents cadenas

	NOKÉ	Original LINKA	LINKA Leo	OTO Hunter
Cadenas	Plusieurs types dont un spécial pour vélo avec chaîne (non fixe)	Cadenas fixe pour vélo	Cadenas fixe pour vélo pour tout type de roue (plus grand que <i>Original LINKA</i>)	Cadenas fixe pour tout type de vélo
Fonctionnement de déverrouillage	Tenir le smartphone à moins de 3 m. du cadenas (Bluetooth)	Être à proximité avec le téléphone et appuyer sur le bouton du cadenas (Bluetooth)	Être à proximité avec le téléphone et appuyer sur le bouton du cadenas (Bluetooth)	Être à proximité avec le téléphone (Bluetooth)
Web API	"Easy to use"	Oui	Oui	Oui
Intégration web	Oui	Oui	Oui	Aucune information
Intégration application	Oui	Oui	Oui	Oui
Prix de l'API	Aucune information	€ 2.– par cadenas	€ 2.– par cadenas	Aucune information
Autonomie annoncée de la batterie	1 année	16 mois	2,5 ans	Panneaux solaires
Prix du cadenas	Fr. 250.– durant les 5 premières années et Fr. 70.– après	€ 125.– et € 109.– dès 100 cadenas	€ 184.– et € 167.– dès 100 cadenas + les données internet (exemple : 1 MB de données par mois pour € 2.50 par cadenas)	Aucune information
Fonctionnalités particulières	Suivi géographique et historique des ouvertures. Plateformes complètes pour la gestion du cadenas	Intégration d'un système anti-vol (alarme et message d'alerte à moins de 120 m.)	Intégration d'un système anti-vol (alarme et message d'alerte), possibilité d'avoir la position GPS du vélo	Géolocalisation du vélo, anti-vol, utilisation de panneaux solaires
Problèmes	Livraison impossible avant 2-3 mois	Uniquement disponible pour certains types de vélo (selle droite et respect de certaines dimensions)	Uniquement disponible pour certains types de vélo (selle droite et respect de certaines dimensions)	Produit éventuellement retiré du marché

4. Décisions au sujet des cadenas

4.1. Cadenas

À la suite d'analyses détaillées de l'état de l'art sur les différents cadenas connectés existants sur le marché, tous les cadenas ont été évalués comme potentiellement utilisables pour notre projet.

Après avoir fait de nombreuses recherches sur NOKĒ, nous avons dû prendre contact avec eux pour avoir de plus amples informations au sujet de l'utilisation du cadenas. Nous avons alors appris de leur part que toute livraison de cadenas serait totalement impossible durant deux à trois mois ce qui compliquerait la mise en route du projet. C'est pourquoi, il a été décidé d'abandonner ce cadenas et de se concentrer sur les autres.

A la suite de l'analyse, excluant le choix de NOKĒ, il a été décidé de commander les cadenas suivants afin de pouvoir effectuer des tests :

- OTO Hunter
- Original Linka
- Linka Leo

Les tests et les analyses des cadenas commandés se sont concentrés sur une utilisation basique du cadenas avec l'application, la géolocalisation des cadenas, le système anti-vol, le management d'une flotte de cadenas et l'intégration dans l'API.

4.1.1. OTO Hunter

Ce cadenas était choisi notamment pour sa polyvalence, du fait qu'il peut facilement être accroché à n'importe quel type de vélo. Un autre point important c'est qu'il intègre un panneau solaire ce qui pourrait être intéressant à l'utilisation, parce qu'il permettrait de fonctionner sans se soucier de devoir recharger sa batterie constamment.

Le problème avec OTO Hunter, c'est qu'apparemment l'entreprise n'est plus existante et ne propose donc plus ce produit, il a donc été impossible de communiquer avec eux et d'avoir accès à un cadenas pour faire des tests.

4.1.2. Linka Leo

Linka Leo est le cadenas « premium » de Linka, il permet une sécurité accrue grâce à sa géolocalisation permanente, une grande autonomie de la batterie, un anti-vol qui permet d'être prévenu instantanément de tout mouvement du vélo, la possibilité d'ajouter une chaîne au cadenas pour mieux sécuriser le vélo, et la possibilité aussi d'augmenter la capacité de la batterie. Tous ces points ont fait de celui-ci, un cadenas que l'on voulait à tout prix tester. Son seul désavantage est son prix relativement élevé ainsi que le paiement mensuel d'un accès aux données mobiles.

Les tests ont, pour leur part, été décevants. Tout d'abord, pour une raison mystérieuse, il était impossible d'avoir accès depuis un téléphone Android à la géolocalisation du cadenas (test effectué sur un Huawei P20 et un Samsung Galaxy S8). De plus, tout contact avec le service client était quasiment impossible. Nous n'avons jamais reçu les accès à la plateforme Web qui aurait permis de pouvoir gérer une flotte de vélos ainsi que l'accès à l'API. Tous ces points nous ont donc convaincus qu'une utilisation du cadenas Linka Leo et d'une collaboration avec Linka ne correspondaient pas au besoin de notre projet.

4.1.3. Original Linka

Ce cadenas, tout comme le Linka Leo (voir le point précédent), nous paraissait intéressant et potentiellement en lien avec nos besoins. Original Linka se trouve à un prix beaucoup plus abordable que les autres cadenas.

A la différence du Linka Leo, ce cadenas ne possède pas de système de géolocalisation en temps réel. Mais ce point-là n'est pas réellement un problème pour notre projet, car pour avoir l'information de la localisation d'un vélo lors de son utilisation il suffit de récupérer la géolocalisation du téléphone de la personne qui l'utilise. Tout comme pour savoir la localisation d'un vélo verrouillé, il suffit d'utiliser la géolocalisation du téléphone de son dernier utilisateur lorsqu'il l'a déposé et verrouillé.

Les mêmes conclusions que pour le Linka Leo ont pu être constatées pour ce cadenas. Le manque d'accès aux plateformes de Linka déjà existantes ainsi qu'à l'API ont compliqué les tests. De plus, on a reçu un cadenas Original Linka déjà verrouillé et visiblement connecté à un autre appareil ; il s'est donc retrouvé inutilisable vu la qualité du service client de Linka.

C'est pourquoi, on a diagnostiqué aussi ce cadenas comme ne correspondant pas à nos besoins.

4.1.4. Conclusion du choix de cadenas

À la suite des différents tests effectués sur les cadenas sélectionnés après l'analyse de l'état de l'art, on peut affirmer qu'aucun cadenas présent sur le marché actuel ne correspondait aux besoins de notre projet (voir les détails dans le tableau ci-dessous).

Tableau 3 : choix du cadenas

	NOKĒ	Original Linka	LINKA Leo	OTO Hunter
Pourquoi l'avoir choisi ?		1) Prix abordable 2) Possibilité de créer une flotte de vélos 3) Anti-vol	1) Géolocalisation 2) Possibilité de créer une flotte de vélos 3) Anti-vol	1) Compatible avec tous les vélos 2) Panneau solaire
Problèmes rencontrés	Livraison impossible avant 2-3 mois	1) Mauvais service client 2) Cadenas reçu avec défaut 3) Pas reçu d'accès à l'API	1) Mauvais service client 2) Impossible d'avoir la géolocalisation sur Android 3) Pas reçu d'accès à l'API	Impossibilité de commander des cadenas
Correspond à nos besoins ?	NON	NON	NON	NON

4.2. Décision finale

À la suite de l'analyse approfondie des cadenas et le constat qu'aucun d'entre eux ne correspondait à nos besoins, il a fallu réorienter le travail de Bachelor vers d'autres possibilités. Les Mockups du premier système (voir Annexes I : Mockups du premier système de BikeSharing) ainsi que son Product Backlog (voir Annexes II : Product Backlog du premier système) sont disponibles dans les annexes.

L'analyse des systèmes existants dans l'état de l'art nous a permis de déterminer qu'il y avait de potentielles applications qui pourraient correspondre à nos besoins. C'est pourquoi,

on a approché Airbie. On a choisi cette entreprise notamment parce qu'elle est Suisse (l'aspect régional est une valeur importante de KargoBike) mais aussi par le fait qu'elle est nouvelle sur le marché et qu'elle a un potentiel certain dans le domaine des vélos en libre-service. Nous avons donc rapidement pris contact avec son créateur pour se faire une idée d'un partenariat possible avec KargoBike mais également pour la réalisation de ce travail de Bachelor.

Nous avons participé à diverses réunions avec M. Christian Raemy de Airbie afin de comprendre au mieux le logiciel qu'il avait créé. Il nous a notamment fourni l'accès à la dernière version de l'application, ainsi qu'à l'interface administrateur. De plus, il nous a prêté un cadenas connecté et un vélo électrique équipé d'un cadenas afin de pouvoir faire divers tests et surtout d'imaginer les possibilités d'un partenariat.

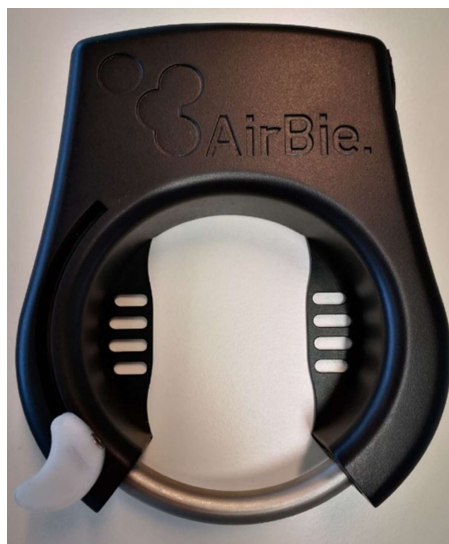


Figure 17 : cadenas connecté Airbie

À la suite de grandes réflexions, en commun accord avec KargoBike et pour éviter tout conflit avec Airbie, il a été décidé de ne plus se focaliser sur une application pour un utilisateur de vélos en libre-service, mais plutôt de porter notre intervention sur la partie maintenance des vélos pour l'administrateur et ses employés. On a pu identifier qu'une interface web de maintenance serait une plus-value pour une entreprise comme KargoBike et que c'était notamment un aspect important que beaucoup d'entreprises de vélos en libre-service n'avaient pas développé.

Ce système pourra donc aussi être proposé sous forme de service auprès d'autres entreprises qui mettent à disposition des vélos en libre-service. Donc notre rôle s'étendra à la réparation et l'entretien des vélos d'autres entreprises.

L'interface affichera tous les vélos signalés par les différents utilisateurs du service et permettra aux employés de maintenance de pouvoir aller facilement réparer un vélo. À la fin de la réparation, l'employé écrit un rapport sur la réparation et calcule le prix qu'a coûté cette réparation. Ce système se veut complet, de la réception d'un signalement de problème, à la réparation du vélo, jusqu'à l'envoi de la facture au client utilisant ce service.

En complément à cela, il y aurait la possibilité par le biais d'une API d'intégrer le système d'envoi de signalement de vélos endommagés directement dans les systèmes des entreprises clientes. Ce qui veut dire, qu'il sera possible de signaler un problème de façon externe à l'outil de maintenance.

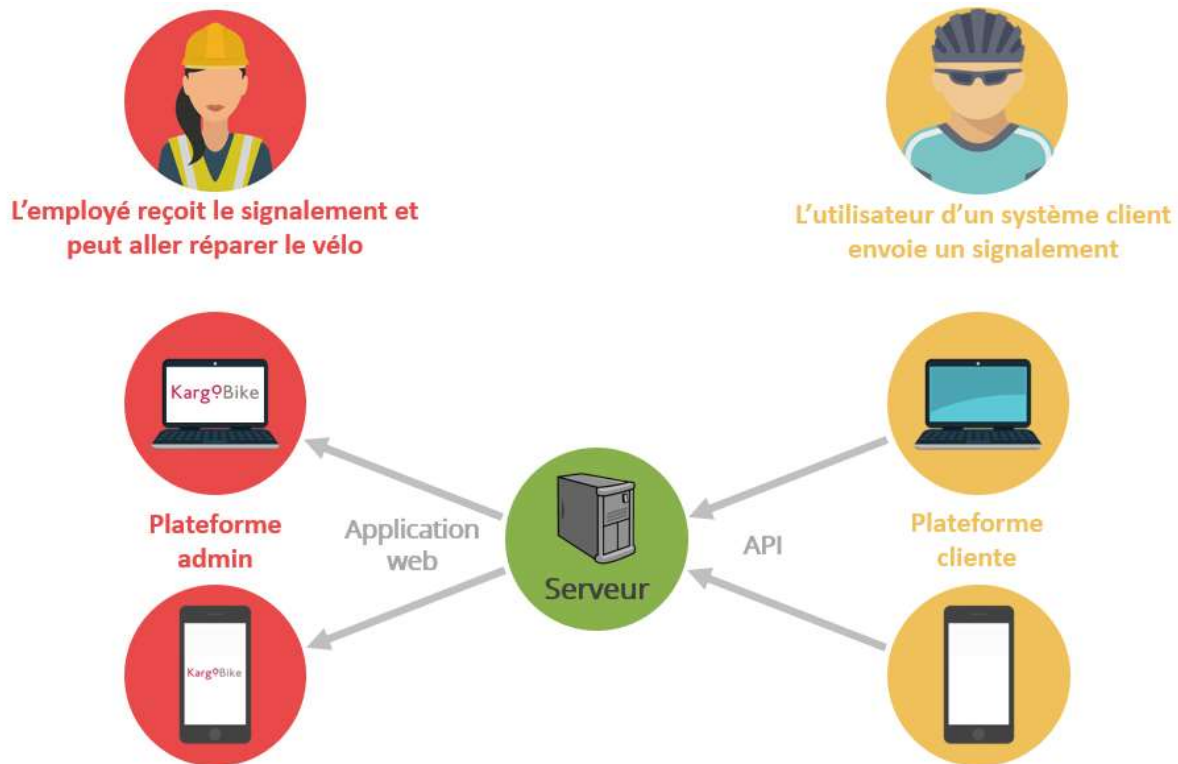


Figure 18 : schéma explicatif du fonctionnement du système de maintenance

5. Analyse des technologies

Cette analyse va se focaliser sur les technologies utilisées pour la conception du système. Dans un premier temps, on va étudier quel type de plateforme serait optimale pour nos besoins.

Les objectifs initiaux sont d'avoir une interface accessible sur toutes les plateformes, c'est pourquoi cette analyse va se baser dans un premier temps sur cet aspect.

5.1. Programmation d'applications mobiles

Pour l'analyse de la programmation d'applications mobiles, on s'oriente pour savoir quels seraient les Operating System (OS) pour mobiles les plus utilisés et donc arriver à une conclusion qui nous permet d'identifier quels seraient les OS essentiels à être développés dans le cas où l'on choisirait de programmer une application.

5.1.1. Analyse des systèmes d'exploitation pour mobiles

Dans un premier temps, il a fallu analyser qu'elles sont les OS mobiles les plus utilisés sur le marché. On a pu déduire facilement que iOS et Android étaient les deux OS les plus répandus en Suisse et dans le monde.



Figure 20 : logo iOS



Figure 19 : logo Android

Comme vu ci-dessous, on voit que Android domine largement le marché mondial suivi, beaucoup plus loin, de iOS (Murray, 2017). Il est prévu que cela persiste ou même augmente d'ici à 2021 (Auffray, 2018).

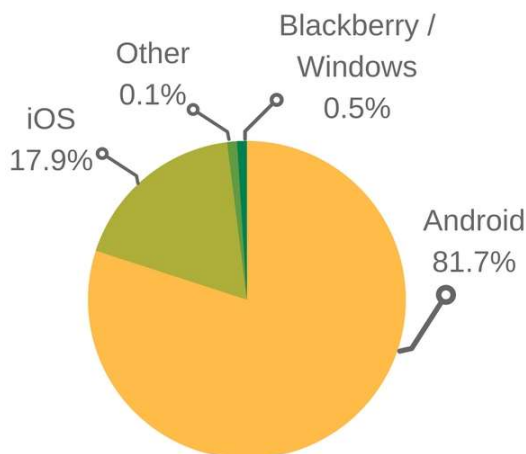


Figure 21 : statistique de janvier 2017 de Applinstitute sur la part de marché des différents OS pour smartphones dans le monde

Mais au niveau Suisse, par contre, c'est quasiment du 50/50 entre iPhone et Android (Mobile Operating System Market Share Switzerland, 2019).

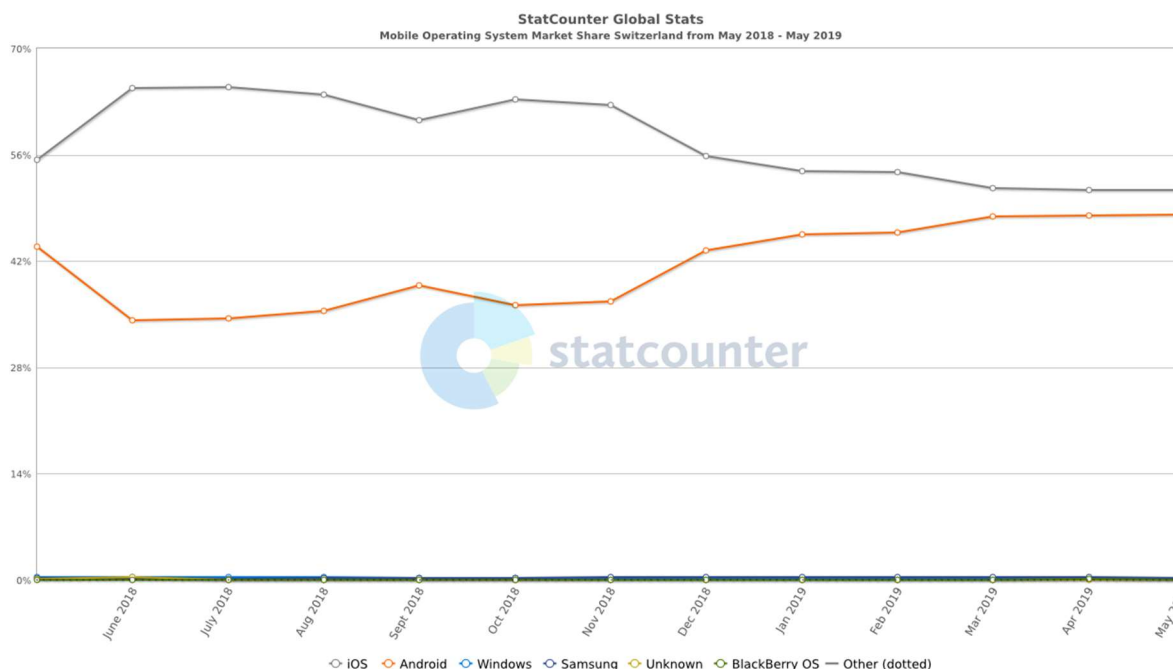


Figure 22 : statistique de StatCounter sur la part de marché des différents OS pour smartphones en Suisse

Comme on peut le voir sur le graphique présent ci-dessus, au niveau Suisse c'est iPhone qui est un rien devant Android.

A la suite de cette analyse, il a été déterminé qu'au niveau Suisse et dans un moindre cas au niveau mondial, il était important de fournir des applications autant pour iOS que pour Android. Donc, dans le cas où nous voulons avoir un système fonctionnel via une application mobile, il faut programmer aussi bien pour Android que pour iOS.

Tableau 4 : comparatif des OS mobiles

	Android	iOS
Parts du marché mondial	81.7% (2017)	17.9% (2017)
Projection de la part du marché mondial pour 2021	85.3%	14.6%
Parts du marché Suisse	48.13% (mai 2019)	51.36% (mai 2019)
Est-il essentiel de programmer pour cet OS ?	Oui	Oui

5.1.2. Décision au sujet des systèmes d'exploitation pour mobiles

À la suite de l'analyse des systèmes d'exploitation, on est arrivé à la conclusion que pour avoir un système multiplateforme correspondant aux besoins, il nous faudrait programmer autant pour Android que pour iOS.

C'est pourquoi vu le temps imparti pour effectuer ce travail de Bachelor et vu le temps que cela prendrait de créer une application Android et iOS, il a été décidé de s'orienter vers une autre possibilité. Il a donc été imaginé de créer un site internet, ce qui permettrait directement d'avoir un système multiplateforme. De plus, en créant des applications Android et iOS, on ne couvrirait même pas le 100% du marché mondial des smartphones, tandis qu'avec un site internet, cela devient possible.

Dans le cas où nous serions restés sur le système initial d'application de vélos en libre-service, les applications auraient été plus judicieuses. Notamment par le fait que la plupart des cadenas connectés présents sur le marché utilisent du Bluetooth pour les déverrouiller, ce qui est actuellement infaisable avec un site internet alors que c'est totalement intégrable dans une application.

5.2. Programmation Web

Pour donner suite à l'analyse des OS pour mobiles, il a été décidé d'analyser des langages de programmation orientés web. Cette analyse s'est effectuée sur 3 langages différents :

- JavaScript avec la plateforme Node.js
- Python par le biais du framework Django
- PHP avec le framework Laravel

Cette analyse met en valeur les avantages et les inconvénients de chacun des langages/frameworks analysés afin de pouvoir tirer une conclusion et choisir la technologie qui répondra le mieux aux besoins de notre projet.

5.2.1. JavaScript - Node.js

Node.js a permis une forme de renouveau pour JavaScript. En effet, auparavant, JavaScript était uniquement utilisé du côté client, c'est-à-dire du côté du navigateur seulement pour afficher les pages internet. Mais maintenant Node.js permet de ne pas utiliser JavaScript uniquement du côté client mais également du côté serveur, ce qui lui permet donc de générer les pages et de les afficher (Nebra, 2018).



Figure 23 : logo Node.js

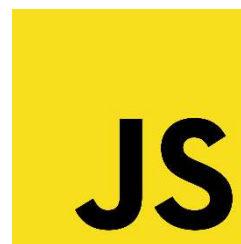


Figure 24 : logo JavaScript

L'un des avantages de Node.js, c'est qu'il est très puissant et rapide, ce qui est idéal pour la création d'un site de chat, d'un site pour télécharger des documents ou pour tout autre site qui a besoin d'une certaine rapidité d'exécution. Tout cela est permis par l'utilisation du moteur d'exécution V8 de Google Chrome. Ce moteur V8 permet d'analyser et d'exécuter du code JavaScript ultra rapidement (Nebra, 2018). En plus de cela, Node.js utilise une conception asynchrone pour gérer les requêtes. Ceci permet de traiter un volume important de requêtes en même temps et donc d'éviter une attente (Sfez, 2017).

Un autre aspect important c'est que Node.js permet d'utiliser un seul et même langage tant du côté Backend que du Frontend. Ce qui permet d'utiliser un langage de moins dans cet environnement en comparaison à d'autres outils tels que Laravel ou Django (Sfez, 2017). « C'est un gain de temps pour le développeur et une économie d'argent pour l'entreprise. » (Sfez, 2017)

Une autre force de Node.js est la taille de sa communauté, c'est l'une des communautés de développeurs la plus présente sur le web (Sfez, 2017). De plus, la configuration de Node.js se veut simple et accessible pour des personnes qui découvrent son environnement (Laurent, 2016).

Les principaux problèmes de Node.js sont au niveau de l'API et des modules. En effet, l'API est souvent mise à jour et les modifications effectuées sont souvent incompatibles avec les versions précédentes, ce qui complique considérablement la tâche des développeurs (Rarchaert, 2017). Concernant les modules, ceux-ci sont souvent peu fiables vu leur environnement open source. Ils sont souvent de mauvaise qualité ou très peu documentés. De plus, nombre d'entre eux n'ont jamais été supervisés (Rarchaert, 2017).

Un autre problème, c'est que Node.js n'est pas adapté pour des sites qui demandent beaucoup de ressources tel que des processeurs. Cela pourrait devenir un problème pour des sites tel que ceux proposant l'édition de vidéos (Chrzanowska, 2017).

5.2.2. Python – Django

Django est un framework open source qui permet la création de sites internet en utilisant le langage de programmation Python (Qu'est-ce que Django?, s.d.). Ce framework se base sur la technologie model-view-controller (MVC) qui facilite la création de la structure du site web et permet une séparation entre l'affichage et la logique (Kasimov, 2017).



Figure 26 : logo Django



Figure 25 : logo Python

Une des forces de Django, c'est Python. En effet ce langage se veut simple et accessible, il faut notamment 40% moins de lignes de code en utilisant Python plutôt que Java. La syntaxe de Python permet également plus facilement d'avoir un code propre, c'est-à-dire un code bien structuré qui fera gagner du temps au développeur. On peut facilement retrouver cette facilité et rapidité de codage dans Django. En effet, une fois qu'on a les connaissances de Django, il est alors très rapide de coder un site internet avec ce framework (Franck, 2015).

Un autre avantage, c'est la communauté d'utilisateurs, aussi bien pour Django que pour Python. En effet, la communauté Django est très active et fournit des tutoriels de qualités. C'est la même chose avec Python notamment vu que l'on retrouve ce langage un peu partout dans l'informatique (Franck, 2015).

Django utilise également un système de Template qui fournit des modèles qui peuvent être réutilisés sur plusieurs parties du site ; c'est un énorme avantage parce que cela évite notamment d'avoir à doubler le code et donc c'est un gain de temps (Franck, 2015).

Un dernier point important, c'est la partie administrateur que fournit Django ; elle est très utilisée car elle est facilement configurable et permet de modifier rapidement et facilement des données. En effet, depuis la partie administrateur, il est facile de créer, modifier ou supprimer les champs de la base de données (Franck, 2015).

Un désavantage de Django, c'est que sa prise en main qui n'est pas des plus évidentes lorsqu'on a très peu ou aucune connaissance du framework. Pour commencer à programmer, il faut donc dans un premier temps se renseigner avec précision sur son utilisation. Mais une fois les connaissances acquises, c'est facile et ultra rapide de l'utiliser (Hansen, 2017).

L'utilisation de Regex (expression régulière) pour spécifier le routage des Uniform Resource Locator (URL) n'est pas des plus optimales, en effet ce routage implique parfois des lignes de code très grandes avec une syntaxe plutôt compliquée ce qui ne facilite pas son utilisation (Sidorenko, s.d.).

Comme le dit Nuri Kasimov (À quoi sert le framework Django ?, 2017), un autre inconvénient c'est que « Django reste lourd et complique la migration à cause de son poids, il est mieux de ne pas utiliser la plateforme pour des petits projets » (À quoi sert le framework Django ?, 2017).

5.2.3. PHP – Laravel

Laravel est un framework relativement récent en comparaison des autres reconnus sur le marché ; il a été créé en 2011 par un certain Taylor Otwell. Au lieu de partir de rien, son créateur s'est basé sur un autre framework, Symfony, pour le créer. Il se base sur la structure MVC ce qui permet de structurer de façon logique le projet (Pourquoi utiliser le framework Laravel ?, 2018).

Son installation et sa configuration se veut simple. De plus, il est accessible facilement aux personnes qui ont des connaissances en programmation, mais il ne faut pas forcément avoir des connaissances en PHP ou en Laravel pour se débrouiller, bien qu'il soit tout de même conseillé d'en avoir quelques connaissances (Pourquoi utiliser le framework Laravel ?, 2018).



Figure 28 : logo Laravel



Figure 27 : logo PHP

Un des atouts majeurs réside dans la sécurité de Laravel : il « ne permet à aucune application de logiciel malveillant ni de menace à la sécurité d'entrer dans l'application Web » (Pourquoi utiliser le framework Laravel ?, 2018). De nos jours, la sécurité est au centre de tous les débats, c'est pourquoi il est important de pouvoir s'appuyer sur un Framework qui promet une haute sécurité (Pourquoi utiliser le framework Laravel ?, 2018).

De plus, Laravel intègre une fonctionnalité simple et efficace de test unitaire. Dans le développement d'un site internet, c'est une étape importante parce que ces tests permettent de pouvoir identifier rapidement si un bug ou un problème apparaît sur des fonctionnalités créées auparavant (Pourquoi utiliser le framework Laravel ?, 2018).

Laravel a la chance de s'appuyer aussi sur une grande communauté qui, de plus, ne cesse de croître ; de nombreux développeurs continuent de se tourner vers Laravel. D'autre part, sa documentation est très développée et bien conçue afin de pouvoir se renseigner au mieux (Pourquoi utiliser le framework Laravel ?, 2018). Cependant, il arrive parfois de tomber sur de la documentation qui n'est pas actualisée suite à l'apparition de nouvelles mises à jour Laravel bien que, dans certains cas, celles-ci modifient totalement l'usage et la perception de certaines fonctionnalités. De plus, il est parfois possible qu'il faille lire et relire beaucoup la documentation avant de pouvoir effectuer une tâche dans Laravel, car elle peut parfois être assez compliquée et demander certaines connaissances avant de se lancer dans le code (Oamkumar, 2018).

De nombreuses bibliothèques sont également préinstallées dans Laravel telle que celle d'authentification qui permet de rendre cette dernière totalement sécurisée avec la protection cross-site request forgery (CSRF), mais également la bibliothèque SwiftMailer qui permet un envoi de courriel simplifié (Laravel – Pourquoi choisir ce framework pour développer votre MVP ?, 2018). En plus de cela, Laravel fournit un système de ressources API Representational State Transfer (REST) qui est simple et facilement intégrable à un projet. Cette API REST utilise des fichiers JavaScript Object Notation (JSON) pour transmettre les données (Hicham, 2017).

Laravel intègre un système de routage simple et bien structuré. Il utilise des Templates Blade qui permettent d'intégrer à la vue (au visuel) facilement du code PHP au milieu du code Hypertext Markup Language (HTML). Cela permet notamment de ne pas perdre en rapidité d'affichage les pages web (Laravel – Pourquoi choisir ce framework pour développer votre MVP ?, 2018).

5.2.4. Comparaison des langages

Tableau 5 : comparaison des langages informatiques - Framework

	JavaScript - Node.js	Python – Django	PHP - Laravel
Avantages	<ul style="list-style-type: none"> • Rapidité d'exécution grâce au moteur V8 • Un seul langage pour la partie Backend et Frontend • Gestion de requêtes asynchrones • Grande communauté de développeurs • Configuration relativement facile 	<ul style="list-style-type: none"> • Structure MVC • Python est simple et accessible • Rapidité de codage • Communauté active et tutoriels de bonnes qualités • Système de Template pour la partie administrateur 	<ul style="list-style-type: none"> • Structure MVC • Python est simple et accessible • Configuration et installation simples • Juste besoin de connaissances de programmation basiques • Haute sécurité • Test unitaire • Grande communauté • Documentation détaillée • Nombreuses bibliothèques préinstallées • Système API REST intégré
Inconvénients	<ul style="list-style-type: none"> • API instable • Certains modules open source ne sont pas fiables • Pas adapté à des opérations lourdes 	<ul style="list-style-type: none"> • Difficile d'accès sans connaissance de base de Django • L'utilisation de Regex • Framework lourd • Déconseillé pour des petits projets 	<ul style="list-style-type: none"> • Documentation parfois pas mise à jour en retard • Demande parfois beaucoup de temps pour lire la documentation avant de coder

5.2.5. Choix du langage de programmation

Le choix du langage de programmation s'est basé sur ce qu'apportait chacun comme avantages et désavantages en lien avec notre projet.

JavaScript avec Node.js est intéressant notamment pour sa grande rapidité et le fait qu'avec l'utilisation de JavaScript, il n'y a plus qu'un seul langage pour le Backend et le Frontend. Mais sa rapidité n'est pas forcément essentielle dans notre cas parce qu'on n'a pas forcément besoin d'intégrer un système de chat ou un autre élément qui demande une certaine rapidité. De plus Node.js a un handicap majeur, celui d'être mis à jour souvent, et le fait que ses mises à jour ne sont pas forcément compatibles avec les versions précédentes. Cela pourrait être handicapant à l'avenir dans le cas où l'on voudrait passer à une version plus récente. Mais Node.js peut se vanter d'avoir la plus grande communauté ; il est facile de trouver du contenu tel que de la documentation ou des tutoriels traitant de celui-ci sur internet.

Django apporte pour sa part une excellente structure en MVC, mais en revanche il n'est pas facile à prendre en main ; il faut un certain temps pour comprendre comment fonctionne ce dernier. D'un autre côté, l'utilisation de Python permet de gagner du temps, en effet écrire en Python est plus efficace et permet de coder sur moins de lignes. Mais comme le dit Nuri Kasimov (À quoi sert le framework Django ?, 2017), Django est un software très lourd et est donc peu adapté à des petits projets, ce qui est le cas du nôtre.

Laravel, quant à lui, est un framework aussi bien structuré mais il rajoute un système de routage très efficace et facile d'utilisation. Sa prise en main est aussi simple en ayant des connaissances en informatique. Il intègre de nombreuses librairies préinstallées dont notamment une pour la messagerie qui s'avère être utile pour notre projet puisqu'on aura besoin d'envoyer des courriels de confirmation. De plus, il intègre un système de création d'API, ce qui est également un des objectifs de ce projet. Quant à sa documentation, elle est très vaste et complète mais peu parfois s'avérer en retard dans sa mise à jour.

Après l'analyse de tous ces points, il a été décidé d'utiliser PHP et Laravel notamment parce que le framework intègre des outils tels que les courriels et la création d'API directement. Ces deux aspects sont très importants pour le développement de notre site internet, tandis que les deux autres technologies doivent faire appel à d'autres modules pour avoir accès à cela,

modules qui ne sont pas forcément aussi simples d'utilisation que ce ne l'est dans Laravel. De plus, il faut prendre en compte que des connaissances sur Laravel ont été acquises lors de précédents projets ce qui influe sur le temps de développement ; c'est un point non négligeable pour le temps accordé au présent projet.

Tableau 6 : comparaison pour le choix du langage de programmation

	JavaScript – Node.js	Python - Django	PHP - Laravel
Avantages pour notre projet	<ul style="list-style-type: none"> • Rapidité • Même langage Backend et Frontend • Grande communauté 	<ul style="list-style-type: none"> • Structure MVC • Python est efficace et simple 	<ul style="list-style-type: none"> • Structure MVC • Routage • Prise en main facile • Vaste et bonne documentation
Désavantages pour notre projet	<ul style="list-style-type: none"> • Incompatibilités des versions successives 	<ul style="list-style-type: none"> • Complicé pour débiter • Lourd et pas adapté aux petits projets 	<ul style="list-style-type: none"> • Documentation parfois mal mise à jour
Système d'envoi de courriels	Module à installer	Oui	Oui
Système de création d'API	Oui	Module à installer	Oui
Connaissance de base personnelle	Aucune	Basique	Bonne
Technologie choisie	NON	NON	OUI

5.3. Base de données

Pour notre projet, nous avons décidé de partir sur une base de données MySQL. C'est un « **Système de Gestion de Bases de Données Relationnelles** » (Gribaumont, 2019) qui utilise le langage Structured Query Language (SQL), ce qui veut dire que c'est un logiciel qui permet de gérer un grand nombre de données et de les stocker dans une base de données (Gribaumont, 2019).



Figure 29 : logo MySQL

Le principal avantage d'utiliser MySQL est qu'il est facile à utiliser et à intégrer à Laravel. Il intègre également de nombreuses fonctionnalités SQL tout comme, nous l'avons vu plus haut, il autorise un grand type de données contrairement à d'autres logiciels. Sa rapidité, sa puissance et sa sécurité sont également des avantages intéressants pour notre projet (Wang Y. , 2017).

5.4. Outils de développement

Dans ce chapitre, nous allons lister et détailler tous les outils et technologies de développement qui sont présents dans notre projet. Ce sont autant des outils utilisés pour coder en Backend qu'en Frontend.

5.4.1. Xampp

X (cross) Apache MariaDB Perl PHP (Xampp) est un ensemble de logiciels qui permet de mettre en place un environnement web PHP localement sur un système. Ce logiciel permet d'interpréter le langage PHP, d'utiliser le serveur Apache (voir point suivant) et d'utiliser MySQL pour la base de données. Tous ces éléments permettent de créer un serveur web local et sont donc très utiles pour utiliser notre environnement PHP (Hory, 2015).



Figure 30 : logo XAMPP

5.4.2. Apache

Apache est un serveur Web. C'est un « intermédiaire entre le serveur et les machines des clients. Il extrait le contenu du serveur sur chaque requête d'utilisateur et le transmet au web » (Hajir, 2018). C'est l'outil qui permet la communication entre le serveur et le client.



Figure 31 : logo Apache

Dans notre cas, lorsqu'on effectue une requête pour accéder à une page dans un navigateur, Apache va aller chercher sur le serveur les fichiers correspondants (fichier PHP, HTML, etc.) et va les afficher dans notre navigateur du côté client sous forme d'une page HTML (Hajir, 2018).

5.4.3. Eloquent

Eloquent est un Object-Relational Mapping (ORM) proposé par Laravel. Ce système permet la création de modèles (moyen pour représenter le contenu d'une base de données) et une utilisation simplifiée de l'accès aux contenus de la base de données. En effet, celle-ci permet notamment de faire des requêtes simplifiées et donc de ne pas devoir utiliser des requêtes SQL qui sont parfois longues et compliquées à mettre en forme.

5.4.4. Leaflet avec OpenStreetMap

Leaflet est une bibliothèque JavaScript qui permet d'afficher des cartes dynamiques. Elle utilise la technologie cartographique de OpenStreetMap. Pour y accéder, il faut faire appel à l'API Leaflet dans une page HTML. Elle permet notamment de se déplacer sur une carte et d'afficher des pointeurs pour indiquer l'emplacement d'un lieu. Pour notre projet, il est judicieux d'utiliser cette bibliothèque parce qu'elle est simple d'utilisation ; elle correspond à nos besoins et est totalement gratuite à la différence de Google Maps.



Figure 33 : logo OpenStreetMap



Figure 32 : logo Leaflet

5.4.5. Laravel API

Laravel fournit une API pour les développeurs, c'est-à-dire qu'il est possible de créer son API directement dans le projet Laravel. L'API utilise la technologie Hypertext Transfer Protocol (HTTP) pour communiquer entre le serveur et les sites internet. Elle utilise donc les requêtes HTTP qui sont GET (récupérer des données pour les afficher), PUT (modifier des données), POST (insérer des données) et DELETE (supprimer des données). Pour transmettre des données, l'API les transforme sous le format JSON (Castelo, s.d.).

5.4.6. SwiftMailer Laravel

Laravel intègre directement dans son framework une technologie d'envoi de courriels. Il utilise la technologie de SwiftMailer, tout en simplifiant son utilisation. Il est possible de configurer l'envoi de courriels depuis un serveur de messagerie standard ou depuis un webservice comme Gmail. La configuration est très simple et permet de créer un courriel personnalisé. L'envoi de courriels étant un des objectifs du projet, SwiftMailer est sans conteste un excellent système qui correspond à nos besoins.



Figure 34 : logo SwiftMailer

5.4.7. Markdown

Markdown est un langage informatique qui permet d'écrire en texte ce qui sera affiché dans des pages en HTML. Il est alors plus facile d'écrire de l'HTML avec cela ; c'est une sorte de raccourci d'HTML. L'utilité de ce langage pour notre projet se trouve pour l'envoi de cour-



Figure 35 : logo Markdown

riels, en effet, ce langage permet tout autant de représenter sous un format texte normal que sous un format HTML. Ceci est très intéressant pour les courriels mais également si nous voulions plus tard rajouter l'envoi de notification par SMS, il suffirait alors de réutiliser le même texte (Nebra, Rédigez en Markdown !, 2017).

5.4.8. Bootstrap

Bootstrap est un framework CSS qui permet de faire de la mise en pages de site web ; c'est purement du frontend. Il permet de disposer des éléments facilement sur une page et de fournir des composants déjà tout configurés esthétiquement en CSS mais également en JavaScript. Cette technologie permet de gagner du temps tout en ayant un visuel clair et agréable. Pour



Figure 36 : logo Bootstrap

l'utiliser c'est vraiment simple : il suffit d'appeler les classes (attributs qui permettent la personnalisation avec CSS) toutes configurées par Bootstrap directement dans le code html. Le

fait d'utiliser Bootstrap a un impact certain dans la rapidité de la création de notre projet, ce qui en fait un gros avantage.

5.4.9. Flash Messages

laracasts/flash est une bibliothèque externe de création de messages flashs. Ce sont des messages qui peuvent être affichés lors d'un événement ; ce peuvent être des messages de type alerte, information, succès, etc. C'est une bibliothèque créée par un autre développeur, donc externe à Laravel. Cette bibliothèque est intéressante pour notre projet notamment pour afficher des messages quand un utilisateur doit se connecter avant de pouvoir accéder à une page ou pour afficher un message de succès lorsqu'un formulaire a pu être envoyé sans problème (Easy Flash Messages for Your Laravel App, 2018).

5.4.10. Laravel Excel

Laravel Excel est une bibliothèque créée par Maatwebsite. Son but est de pouvoir exporter des données depuis Laravel dans un fichier au format Excel ou Comma-Separated Values (CSV). A l'aide de cette bibliothèque, il est alors très simple d'exporter des données. Cette technologie a sa place dans notre projet pour combler le besoin de pouvoir exporter certaines données de notre site (Maatwebsite, s.d.).

6. Développement

6.1. Mockup

Dans la phase de développement, l'un des tout premiers points à effectuer est la création de Mockups, en effet ceci permet de visualiser les idées. C'est une phase essentielle car c'est depuis les Mockups que le client et le développeur vont pouvoir se mettre d'accord sur la façon dont sera représenté le site internet. C'est très important d'effectuer des Mockups car c'est un moyen de voir si le développeur a compris tous les besoins de son client.

Ces Mockups sont là pour représenter le visuel d'une interface. Dans notre cas, plusieurs Mockups ont été effectués, mais seuls les plus importants sont développés plus loin.

6.1.1. Page d'accueil

La page d'accueil est la page principale du site internet, c'est sur cette page que l'employé aura une vue globale de tous les vélos qui ont été signalés.

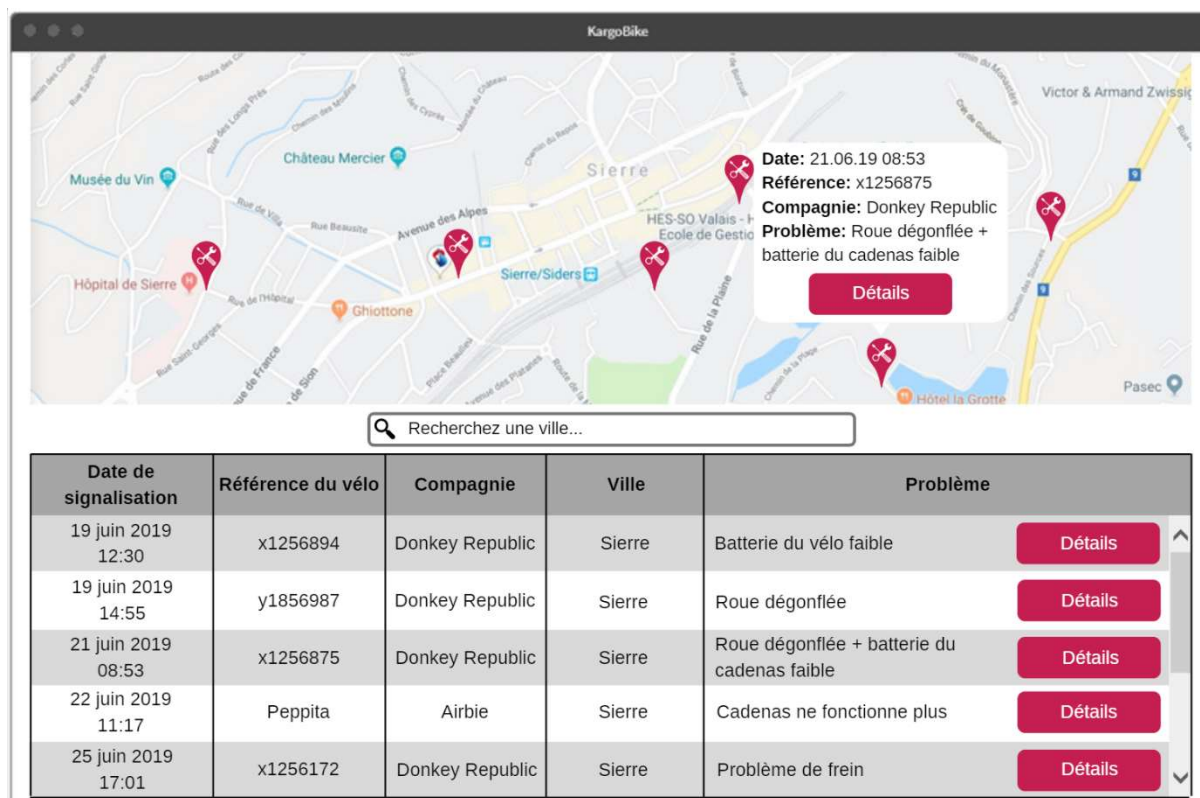


Figure 37 : Mockup de la page d'accueil

Cette page est divisée en 2 parties, la première, supérieure, étant la carte. Celle-ci représente visuellement tous les vélos signalés sur la carte. C'est un aspect intéressant pour les employés de pouvoir situer l'emplacement des vélos pour visualiser quel vélo se trouve à proximité de lui pour qu'il puisse aller le réparer. Lorsque l'on clique sur un pointeur, on peut avoir un détail du vélo et de son signalement.

La seconde partie est le tableau ; celui-ci liste tous les vélos signalés par ordre de date, du signalement le plus vieux au plus récent. C'est un autre aspect important pour les employés car il ne faudrait pas laisser un vélo trop longtemps sans réparation.

Un autre aspect intéressant est le champ de recherche, il permettra à l'employé de pouvoir effectuer une recherche sur un champ spécifique du tableau.

6.1.2. Page de détail de signalement

La page de détail d'un signalement affiche toutes les informations nécessaires pour la réparation d'un vélo.

Détails du vélo x1256875

Date de signalisation: 21 juin 2019 08:53

Compagnie: Donkey Republic

Ville: Sierre

Titre du problème: Roue dégonflée + batterie du cadenas faible

Détails du problème: Vélo inutilisable, la roue avant est totalement dégonflée. Il est possible qu'elle soit percée. Et la batterie du cadenas est à moins de 20%

Historique des réparations:

Date	Type de réparation	Prix
06 juin 2019 12:30	Remplacement de la roue arrière	25.-
19 mai 2019 17:35	Changement des freins	75.-
19 jan. 2019 15:02	Chargement de la batterie du cadenas	12.50

Réparer maintenant

Retour

Figure 38 : Mockup de la page de détail d'un signalement

On retrouve dans un premier temps les informations du vélo ainsi que les informations détaillées de son problème. En complément de ces informations, on peut voir l'emplacement du vélo sur la carte.

En plus de cela, l'employé pourra voir l'historique des réparations sur ce vélo avec la date, de la plus récente à la plus ancienne, le type de réparation et le prix.

6.1.3. Page de réparation

La page de réparation est la page dans laquelle l'employé viendra remplir toutes les informations importantes concernant la réparation qu'il vient d'effectuer.

Réparation du vélo x1256875

Titre du problème:
Roue dégonflée + batterie du cadenas faible

Détails du problème:
Vélo inutilisable, la roue avant est totalement dégonflée. Il est possible qu'elle soit percée. Et la batterie du cadenas est à moins de 20%

Titre

Détails

Prix de la réparation: Fr.

Rép. impossible **Réparé**

Retour

Figure 39 : Mockup de la page de réparation d'un vélo

Sur cette page, on retrouve un petit rappel au sujet du problème du vélo et un formulaire à remplir concernant la réparation. Ce formulaire a un titre qui est une courte phrase résumant la réparation (ce titre est important pour afficher la réparation dans l'historique), un détail où seront indiqués tous les détails de la réparation et enfin le prix de la réparation.

On peut également signaler l'impossibilité d'une réparation en remplissant les mêmes champs détaillés précédemment. Une réparation est impossible lorsqu'un vélo n'est tout simplement plus réparable et donc qu'il est juste bon à être jeté.

6.2. Guide technique

Le guide technique est destiné à permettre à d'autres personnes, un jour, de pouvoir reprendre le développement de notre outil de maintenance. Ce guide va reprendre toutes les spécificités techniques qui permettront de continuer le projet.

6.2.1. Version des technologies

Cette partie va lister toutes les versions des différentes technologies utilisées par notre projet. Les numéros de version indiqués ci-dessous ont été utilisés lors du développement.

Tableau 7 : version des technologies présentes dans le projet

Technologie	Utilisation dans le projet	Version
PHP	Langage de programmation pour la conception d'application web	7.3.2
Apache	Serveur Web pour les pages HTTP et l'interprétation PHP	2.4.38
MySQL	Serveur de base de données pour stocker les données de notre site	5.7.21
Laravel	Framework PHP pour garder le code organisé et maintenable	5.8.27
Eloquent	Composant Laravel pour l'utilisation d'ORM.	Version utilisée dans Laravel 5.8
SwiftMailer	Bibliothèque qui permet l'envoi de courriel, utilisée pour notifier une nouvelle tâche et confirmer la réparation du vélo	Version utilisée dans Laravel 5.8
Leaflet	Bibliothèque JavaScript permettant d'afficher et d'interagir avec OpenStreetMaps, utilisée pour afficher les vélos à réparer	1.5.1

OpenStreetMap	Cartographie terrestre utilisée par Leaflet pour localiser les vélos à réparer	Version utilisée dans Leaflet 1.5.1
Markdown	Langage informatique qui permet d'afficher du texte en html ou en format texte normal	Version utilisée dans Laravel 5.8
Bootstrap	Un framework CSS pour garder l'affichage et la mise en pages côté client organisées et adaptables	4.3.1
Laravel Excel	Bibliothèque qui permet d'exporter des données dans un fichier Excel.	3.1
laracasts/flash	Bibliothèque qui permet d'afficher des messages dans une page HTML, utilisée pour demander à l'utilisateur de se connecter ou pour signaler l'envoi réussi d'un formulaire.	3.0

6.2.2. Arborescence du projet

Ci-dessous, on peut voir l'arborescence des fichiers dans Laravel :

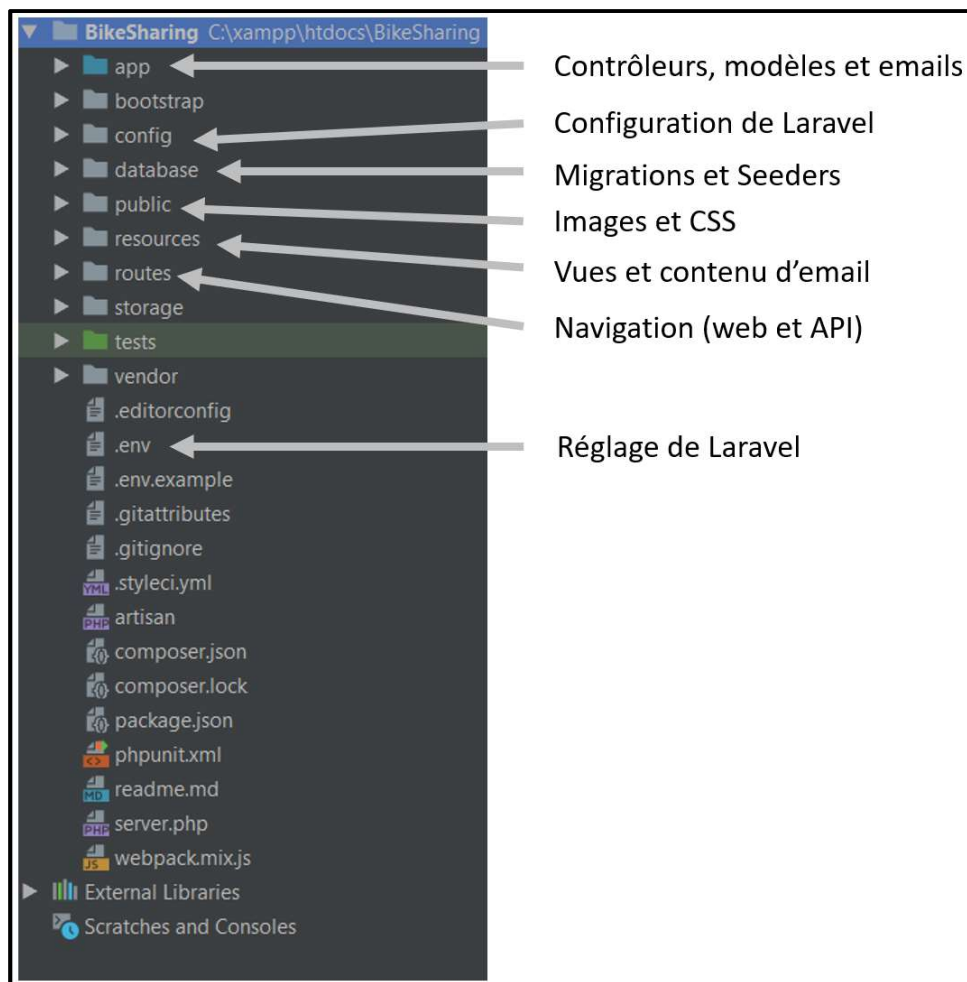


Figure 40 : arborescence des fichiers Laravel

Le dossier «app» contient tous les contrôleurs, modèles et les fichiers de la logique des courriels.

Le dossier «config» contient tous les fichiers de configuration de Laravel. On retrouve par exemple la configuration qui permet que Laravel sache dans quel fichier se situe les vues (le chemin pour que Laravel accède aux vues) ou alors on peut retrouver la configuration pour se connecter à une base de données avec des valeurs par défaut.

Le dossier «database» contient toutes les migrations et les Seeders (voir le point 6.2.5. Création de la base de données). Avec les migrations, nous créons les tables de notre base de

données et insérons les valeurs par défaut dans les tables. Les Seeders permettent de remplir la base de données avec des données prédéfinies.

Le dossier «public» contient le code CSS ainsi que les images qui seront appelées dans les vues (voir le point 6.2.8. Vues).

Le dossier «resources» contient toutes les vues qui seront utilisées pour générer le visuel des pages HTML. On retrouve également les fichiers contenant le texte des courriels écrits en Markdown.

Le dossier «routes» contient le code qui est responsable de la navigation du site web. C'est ici que se retrouve tout le routage des pages web mais également de l'API.

Le fichier «.env» contient toutes les variables de configurations de Laravel. Il est alors possible de configurer énormément de fonctionnalités directement ici sans devoir aller dans le dossier «config» pour le faire. Ici, on peut par exemple configurer l'accès à la base de données ainsi que l'accès à une boîte de messagerie pour envoyer des courriels depuis Laravel.

6.2.3. Configuration du fichier «.env»

Laravel fonctionne grâce à un fichier nommé «.env». Ce fichier décrit l'environnement dans lequel Laravel doit être intégré. On retrouve à l'intérieur de celui-ci les variables de configurations qui permettent de configurer notre application et tout cela est géré dans un seul fichier.

Les variables qui doivent être définies pour que le projet fonctionne correctement sont les suivantes :

Tableau 8 : variable du fichier de configuration «.env»

Variable	Valeur	Description
APP_NAME	KargoBike	Le nom de notre projet. Ce nom est notamment utilisé dans l'envoi de courriel
APP_ENV	local	Sert à identifier l'environnement actuel
APP_URL	http://kargobike-maintenance.local	L'URL qui permet d'accéder au dossier Laravel/public
DB_CONNECTION	mysql	Le nom du type de base de données utilisée

DB_HOST	127.0.0.1	L'hôte fournissant le service de base de données
DB_PORT	3306	Le port pour accéder à la base de données
DB_DATABASE	bikesharing	Le nom de la base de données
DB_USERNAME	<nom utilisateur>	Le nom de l'utilisateur qui a l'accès à la base de données
DB_PASSWORD	<mot de passe>	Le mot de passe de l'utilisateur qui a l'accès à la base de données
MAIL_DRIVER	smtp	Le nom du protocole utilisé pour l'envoi de courriel
MAIL_HOST	smtp.googlemail.com	L'hôte fournissant le service d'envoi de courriel
MAIL_PORT	465	Le port pour accéder au service d'envoi de courriel
MAIL_USERNAME	<adresse email>	L'adresse email utilisée pour envoyer les courriels
MAIL_PASSWORD	<mot de passe>	Le mot de passe du compte de la messagerie
MAIL_ENCRYPTION	ssl	L'encryption (chiffrement) utilisé pour envoyer des courriels

6.2.4. Schéma de la base de données

Le schéma est une représentation graphique d'une base de données. C'est la configuration logique de la base de données. La conception d'un schéma comme celui-ci, permet au développeur de se faire une première idée de comment sont connectés entre elles les différentes tables, c'est essentiel d'effectuer cela avant de commencer à programmer.

Dans notre cas, notre base de données à 3 tables (Utilisateur, Signalement et Réparation).
Sa structure est effectuée comme suit :

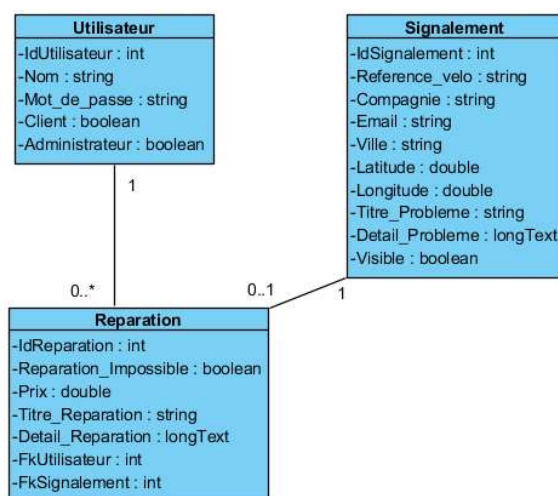


Figure 41 : schéma de la base de données

Table Utilisateur

La table Utilisateur est celle qui contient tous les utilisateurs qui ont l'accès au site internet.

Tableau 9 : détails de la table Utilisateur

Nom du champs	Description	Type
idUtilisateur	L'ID de l'utilisateur	int
nom	Le nom de l'utilisateur qui servira pour se connecter au site	string
mot_de_passe	Le mot de passe de l'utilisateur qui servira pour se connecter au site	string
client	« true » si la personne est un client et « false » si elle ne l'est pas	boolean
administrateur	« true » si la personne est un administrateur et « false » si elle ne l'est pas	boolean

Table Signalement

La table Signalement contient tous les signalements envoyés par les clients

Tableau 10 : détails de la table Signalement

Nom du champs	Description	Type
idSignalement	L'ID du signalement	int
reference_velo	La référence du vélo qui lui permet d'être unique (la même référence ne peut pas être attribuée à deux vélos)	string
compagnie	Le nom de la compagnie cliente	string
email	Le courriel de la compagnie qui signale un vélo (courriel utilisé pour signaler quand un vélo est réparé)	string
ville	Nom de la ville où se trouve le vélo	string
latitude	Latitude du point géographique où se trouve le vélo	double
longitude	Longitude du point géographique où se trouve le vélo	double
titre_probleme	Le titre du problème du vélo rempli par le client	string
detail_probleme	Les détails du problème du vélo rempli par le client	longText
visible	« true » si le vélo n'a jamais été réparé et « false » s'il a déjà été réparé (afin de l'afficher ou non dans la liste des vélos à réparer)	boolean

Table Réparation

La table Réparation contient toutes les réparations de vélo.

Tableau 11 : détails de la table Réparation

Nom du champs	Description	Type
idReparation	L'ID de la réparation	int
reparation_impossible	« true » si la réparation est impossible a effectué et « false » si la réparation a été faite	boolean
prix	Le prix de la réparation qui sera facturé au client	double
titre_reparation	Le titre de la réparation rempli par l'employé	string
detail_reparation	Les détails de la réparation rempli par l'employé	longText
fkUtilisateur	Clef étrangère de la table Utilisateur	int
fkSignalement	Clef étrangère de la table Signalement	int

6.2.5. Création de la base de données

La création de la base de données se fait à l'aide des fichiers de migrations. Ces fichiers nous permettent de créer la structure de notre base de données ainsi que toutes les tables et les attributs. On retrouve ces fichiers à l'emplacement `/database/migrations` comme ci-dessous :

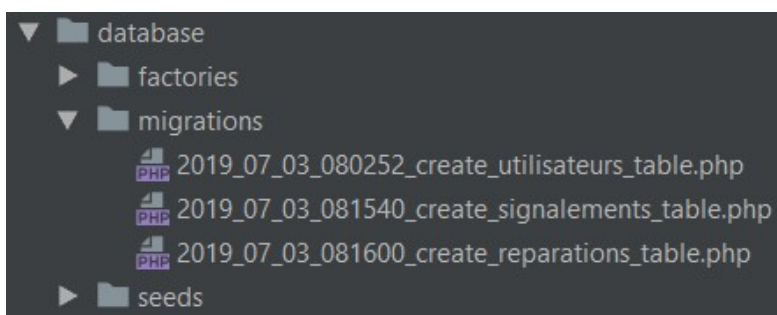


Figure 42 : fichier de migrations dans Laravel

Pour créer la structure de la base de données, il faut lancer la commande suivante :

php artisan migrate:fresh

Cette commande va supprimer toutes les tables existantes dans la base de données et recréer toute la structure. Pour le faire, elle appelle la méthode `up()` comme dans l'exemple ci-dessous pris dans la migration de la table Réparation :

```
public function up()
{
    Schema::create( table: 'reparations', function (Blueprint $table) {
        $table->increments( column: 'idReparation');
        $table->boolean( column: 'reparation_impossible');
        $table->double( column: 'prix');
        $table->string( column: 'titre_reparation');
        $table->longText( column: 'detail_reparation')->nullable();

        $table->unsignedInteger( column: 'fkUtilisateur');
        $table->unsignedInteger( column: 'fkSignalement');

        $table->foreign( columns: 'fkUtilisateur')->references( columns: 'idUtilisateur')->on( table: 'utilisateurs');
        $table->foreign( columns: 'fkSignalement')->references( columns: 'idSignalement')->on( table: 'signalements');

        $table->timestamps();
    });
}
```

Figure 43 : exemple de fichier de migrations pour la table Réparation

Ces fichiers sont configurés d'une manière bien précise. Le type d'attributs «increments» permet de créer l'ID de la table ; celui-ci est auto-incrémenté lorsque l'on crée un nouveau champ dans la table. Il est alors possible de mettre tout type d'attributs différents tels que "boolean" (vrai/faux), "double" (chiffre à virgule), "int" (chiffre entier), "string" (chaîne de caractères) et "longText" (grande chaîne de caractères) par exemple.

Pour la création de clef étrangère (relation avec une autre table), il faut utiliser le type `unsignedInteger` puis créer la connexion entre les deux tables à l'aide du mot-clef "foreign". Cela permettra de récupérer l'ID d'une autre table afin de créer la relation entre deux tables.

"timestamps()" permet pour sa part de créer les champs "created_at" et "updated_at" dans la table ; ce sont respectivement la date/heure de création du champ et la date/heure de la dernière modification du champ dans la base de données.

Pour remplir la base de données avec des valeurs, Laravel utilise un système appelé Seeders. On peut retrouver les fichiers Seeders à l'emplacement `/database/seeds`. Ces fichiers permettent de préremplir la base de données comme on peut le voir ci-dessous :

```
$rep = new \App\Models\Reparation();
$rep->reparation_impossible = false;
$rep->prix = 32.50;
$rep->titre_reparation = 'Batterie rechargée';
$rep->detail_reparation = 'La batterie est à 100%';
$rep->reparateur()->associate(\App\Models\Utilisateur::where('idUtilisateur', 2)->first());
$rep->signalement()->associate(\App\Models\Signalement::where('idSignalement', 1)->first());
$rep->save();
```

Figure 44 : exemple de fichier Seeder

Pour préremplir la base de données, il faut exécuter la commande suivante :

```
php artisan migrate:fresh --seed
```

Cette commande va faire comme précédemment : supprimer toutes les tables et recréer ces dernières juste après, mais, en plus de cela, elle va également remplir les tables avec les données entrées dans les Seeders.

6.2.6. Modèles

Notre projet utilise le composant Eloquent de Laravel. Il permet d'utiliser l'ORM pour interagir facilement et intuitivement avec la base de données grâce à des classes spéciales utilisées comme modèles. Les modèles mis en œuvre dans le cadre du projet suivent la description donnée ci-dessous dans le présent document. Les modèles se trouvent dans le dossier `/app/models`.

```
class Reparation extends Model
{
    protected $primaryKey = 'idReparation';

    protected $fillable = [
        'reparation_impossible', 'prix', 'titre_reparation', 'detail_reparation',
    ];

    public function reparateur()
    {
        return $this->belongsTo(related: 'App\Models\Utilisateur', foreignKey: 'fkUtilisateur', ownerKey: 'idUtilisateur');
    }

    public function signalement()
    {
        return $this->belongsTo(related: 'App\Models\Signalement', foreignKey: 'fkSignalement', ownerKey: 'idSignalement');
    }
}
```

Figure 45 : exemple du modèle Reparation

Comme on peut le voir ci-dessus (page précédente), tous les champs (sauf les clefs étrangères) sont initialisés dans la variable “fillable”. Tandis que les clefs étrangères doivent être initialisées dans une méthode qui permet de créer la relation avec un autre modèle.

6.2.7. Contrôleurs

Le contrôleur est le lieu où toute la logique d’un site internet se trouve. C’est la logique concernant toutes les interactions de l’utilisateur. C’est-à-dire que c’est lui qui va modifier les données présentes dans la vue et dans les modèles. On retrouve nos contrôleurs dans /app/Http/Controllers.

ConnexionController

Le contrôleur de connexion est utilisé lors de la connexion d’un utilisateur via le login.

```
public function traitement()
{
    request()->validate([
        'name' => ['required'],
        'password' => ['required']
    ]);
    $resultat = auth()->attempt([
        'nom' => request( key: 'name'),
        'password' => request( key: 'password')
    ]);

    if($resultat){
        if(auth()->user()->client){
            return redirect( to: '/signalement');
        }
        return redirect( to: '/');
    }

    return back()->withInput()->withErrors([
        'name' => 'Vos identifiants sont incorrects'
    ]);
}
```

Lorsqu’un utilisateur veut se connecter, il est primordial que le nom et le mot de passe soient rentrés. La méthode ci-jointe va vérifier si les identifiants correspondent à ceux qui se trouvent dans la base de données.

Puis il va vérifier si l’utilisateur est un client ou pas afin de le rediriger sur la bonne page le cas échéant. Si les identifiants ne sont pas corrects, un message d’erreur apparaîtra.

Figure 46 : méthode de connexion sur le site

HomeController

Le contrôleur pour la page d’accueil est le contrôleur qui permet d’afficher tous les signalements. La requête ci-dessous va récupérer tous les signalements et les trier par ordre du plus ancien au plus récent.

```
$table = DB::table( table: 'signalements')
->where( column: 'visible', operator: 'LIKE', value: true)
->orderBy( column: 'created_at', direction: 'asc')
->get();
return view( view: 'home')->with('table', $table);
```

Figure 47 : requête pour récupérer tous les signalements

Dans ce contrôleur, on retrouve également la méthode pour la recherche d'un signalement. Elle permet de pouvoir rechercher un mot ou un bout de mot dans une suite de caractères. Cette recherche se fait comme suit :

```
public function search(Request $request)
{
    $request->validate([
        'searchField' => 'required|string',
    ]);

    $table = DB::table('signalements')
        ->where([
            ['' . $request->typeSearch, 'LIKE', '%' . $request->searchField . '%'],
            ['visible', 'LIKE', true]
        ])
        ->orderBy('created_at', 'asc')
        ->get();

    if (count($table) < 1) {
        flash('Aucun vélo ne correspond à la recherche')->error();
    }

    return view('home')->with('table', $table);
}
```

Figure 48 : méthode pour la recherche de signalement

DetailController

Ce contrôleur permet d'afficher les détails d'un signalement, il va sélectionner les informations d'un signalement ainsi que tous les signalements qui ont été déjà effectués pour ce vélo afin de créer un historique.

```
//récupère le signalement dont on veut des détails
$probleme = DB::table('signalements')
    ->where('idsignalement', 'LIKE', request('id'))
    ->get()
    ->first();

//récupère toutes les réparations avec la même référence de vélo
$reparation = Reparation::whereHas('signalement', function ($query) use ($probleme) {
    $query->where('reference_velo', 'LIKE', $probleme->reference_velo);
})->orderBy('created_at', 'desc')
    ->get();
```

Figure 49 : requêtes de la page de détail d'un signalement

ReparationController

Le contrôleur de réparation est celui qui va lancer la requête pour insérer une réparation effectuée dans la base de données. Pour le faire, on vérifie d'abord si on reçoit toutes les données correctement et ensuite on vérifie si c'est une réparation impossible ou si c'est une réparation réussie. Et une fois tout cela effectué, on insère le contenu dans la base de données. On peut voir que, dans la requête d'insertion, on fait appel aux méthodes d'Eloquent vues dans la partie précédente pour remplir les clefs étrangères.

```
$request->validate([
    'titre' => 'required|string',
    'detail' => 'required|string',
    'prix' => 'required|numeric',
]);

if(request( key: 'impossible')==1){
    $impossible=true;
}
else{
    $impossible=false;
}

$rep = new Reparation();
$rep->reparation_impossible = $impossible;
$rep->prix = $request->prix;
$rep->titre_reparation = $request->titre;
$rep->detail_reparation = nl2br($request->detail);
$rep->repareteur()->associate(\App\Models\Utilisateur::where('idUtilisateur', auth()->user()->idUtilisateur)->first());
$rep->signalement()->associate(\App\Models\Signalement::where('idSignalement', request( key: 'idSign'))->first());
$rep->save();
```

Figure 50 : requête d'insertion d'une réparation.

SignalementController

C'est le contrôleur qui est appelé lorsqu'on utilise l'API (voir point 6.2.13. API), on y retrouve la requête pour insérer un nouveau signalement dans la base de données. Ce contrôleur est également utilisé pour la page d'ajout d'un signalement ; c'est la même requête qui est utilisée pour insérer le contenu dans la base de données.

AdminController

Le contrôleur admin est celui qui permet d'afficher toutes les réparations qui ont été effectuées. Pour le faire, il va récupérer toutes les informations de la table « réparations » ainsi que celles de la table « signalements » car il est intéressant de pouvoir voir notamment le numéro du vélo ainsi que le nom de la compagnie qui se trouvent eux dans la table « signalements ».

La requête ci-dessous est celle qui envoie les données après les avoir filtrées selon la demande de l'utilisateur. Elle va permettre de filtrer selon la compagnie, la référence du vélo ou la ville dans la table « signalements » ou alors par le titre de la réparation ou le prix pour la table « réparations ». Il est également possible de filtrer par date ; pour cela le filtre va trouver tous les champs de la base de données compris entre deux dates.

```
//récupérer toutes les réparations et le signalement correspondant à la recherche
$query = Repair::whereHas('signalement', function ($query) use($request){
    //filtrer les données si ça doit être trier sur un champs de la table signalement
    if($request->typeSearch=='compagnie' || $request->typeSearch=='reference_velo' || $request->typeSearch=='ville'){
        $query->where('' . $request->typeSearch, 'LIKE', '%' . $request->searchField . '%');
    }
})->orderBy('created_at', 'desc');

//filtrer les données si ça doit être trier sur un champs de la table réparation
if($request->typeSearch=='titre_reparation' || $request->typeSearch=='prix'){
    $query->where('' . $request->typeSearch, 'LIKE', '%' . $request->searchField . '%');
}

//filtrer les données par date si l'utilisateur le veut
if(!$request->dateFieldStart==null){
    //+1 days pour que toutes les réparation effectués dans la même journée soit considéré
    $query->whereBetween('created_at', array($request->dateFieldStart, date('format: Y-m-d', strtotime($request->dateFieldEnd . '+1 day'))));
}

$stable = $query->get();
```

Figure 51 : requête de filtre pour la page administrateur

C'est cette même requête qui est utilisée pour récupérer les données qui seront exportées dans un fichier en format Excel.

6.2.8. Vues

Dans Laravel, les vues sont en « blade.php » ; c'est un format qui permet de coder en HTML à l'intérieur tout en y intégrant le code PHP directement au milieu du code HTML.

On retrouve nos vues dans /resources/views. La structure des dossiers est la suivante :

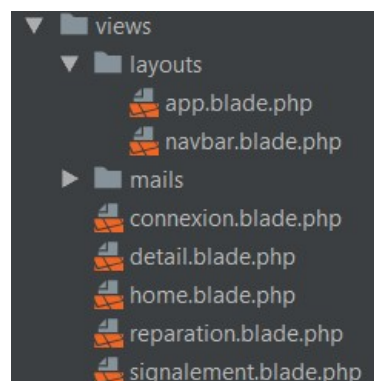


Figure 52 : structure des fichiers vues

Le dossier «layouts» contient la vue app qui est celle de base ; c'est la structure de base de toutes les vues ; c'est dans celle-ci que sont appelées toutes les dépendances utiles au bon

fonctionnement de nos pages html ainsi que du fichier «navbar» qui contient la barre de navigation.

6.2.9. Routage

Les routes dans Laravel sont toutes configurées dans le même fichier /routes/web.php. C'est ici que l'on vient configurer chacune des routes de notre application. Nos routes sont faites de la manière suivante :

```
Route::get('/login', 'ConnexionController@formulaire');
Route::post('/login', 'ConnexionController@traitement');
Route::get('/', 'CompteController@accueil');
Route::group([
    'middleware' => 'App\Http\Middleware\Auth',
], function(){
    Route::get('/signalement', 'SignalementController@accueil');
    Route::post('/signalement', 'SignalementController@store');
    Route::get('/deconnexion', 'CompteController@deconnexion');
});
Route::group([
    'middleware' => 'App\Http\Middleware\AuthClient',
], function(){
    Route::get('/home', 'HomeController@accueil');
    Route::post('/home/search', 'HomeController@search');
    Route::get('/detail/{id}', 'DetailController@accueil');
    Route::get('/reparation/{id}', 'ReparationController@accueil');
    Route::post('/reparation/{impossible}/{idSign}', 'ReparationController@store');
});
Route::group([
    'middleware' => 'App\Http\Middleware\AuthAdmin',
], function(){
    Route::get('/admin', 'AdminController@accueil');
    Route::post('/admin/search', 'AdminController@search');
    Route::post('/admin', 'AdminController@download');
});
```

Figure 53 : routage du site internet

Nous utilisons deux types de routes dans notre projet :

- les routes «get» qui sont là pour afficher du contenu
- les routes «post» qui n'affichent rien mais qui envoient uniquement des données.

Laravel permet également d'utiliser des middlewares. Cela permet de fournir un mécanisme pratique pour filtrer les requêtes HTTP entrant dans notre site. Dans notre cas, nous l'utilisons pour vérifier si un utilisateur qui veut aller à une page spécifique y accède ou pas.

Pour le faire, nous avons créé trois middlewares différents :

- **Auth** : celui-ci contrôle si un utilisateur est connecté ou s'il ne l'est pas ; dans ce cas, il le renvoie sur la page de login.
- **AuthClient** : vérifie comme le précédent si l'utilisateur est connecté mais il vérifie en plus si l'utilisateur connecté est un client parce que les clients n'ont le droit que de voir la page de signalement.
- **AuthAdmin** : vérifie tout comme **AuthClient** mais contrôle en plus si l'utilisateur est un administrateur ou pas.

6.2.10. Carte Leaflet

Les cartes sont générées par Leaflet. Leur configuration est assez simple ; il suffit d'exécuter tous les points suivants pour avoir une carte opérationnelle :

[illegible]

Figure 54 : structure du code de configuration d'une carte

Il est possible d'ajouter autant de marqueurs que l'on veut et surtout de les personnaliser.

Il est également possible d'utiliser la géolocalisation afin de reprendre l'emplacement de l'utilisateur en temps direct. Dans notre cas, on utilise cette technologie pour centrer la carte sur l'emplacement de l'utilisateur pour qu'il puisse apercevoir les vélos à réparer dont il est proche. La configuration de la géolocalisation se fait comme cela :

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
}

function showPosition(position) {
    mymap.setView([position.coords.latitude, position.coords.longitude], 13);
    var geoIcon = L.icon({
        iconUrl: 'http://' + location.host + '/images/markergeo.png',

        iconSize: [10, 10],
        iconAnchor: [5, 10],
        popupAnchor: [1, -1]
    });
    var marker = L.marker([position.coords.latitude, position.coords.longitude], {
        icon: geoIcon
    }).addTo(mymap).bindPopup("Vous êtes ici");
}
```

Figure 55 : obtenir la géolocalisation d'un utilisateur

Dans un premier temps, nous vérifions si le navigateur internet permet la géolocalisation et si c'est le cas il va envoyer une notification à l'utilisateur pour qu'il autorise la géolocalisation (cette étape se fait automatiquement). Puis, après, il est facile de récupérer la longitude et la latitude de l'emplacement afin de pouvoir positionner un marqueur de géolocalisation à cet endroit.

6.2.11. Envoi de courriel

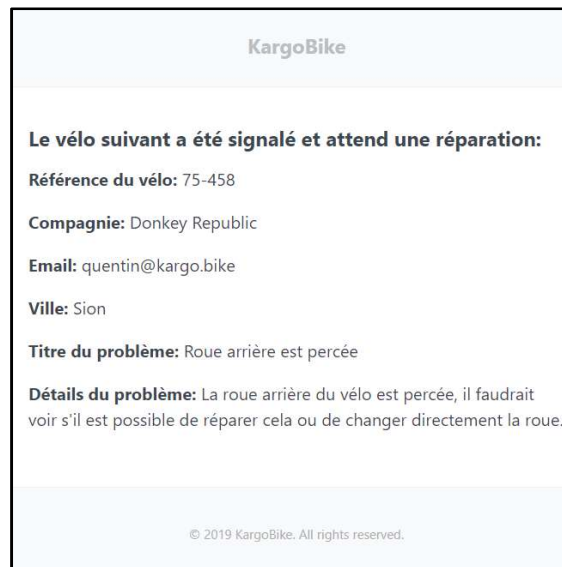


Figure 56 : exemple de courriel envoyé depuis Laravel

Configuration

Dans un premier temps, il faut configurer le fichier « .env » afin d'accéder à la messagerie électronique et d'avoir tous les accès pour le faire. Dans notre cas, nous utilisons la messagerie Gmail car celle que l'entreprise KargoBike utilise.

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.googlemail.com
MAIL_PORT=465
MAIL_USERNAME=quentin@kargo.bike
MAIL_PASSWORD=<mot de passe>
MAIL_ENCRYPTION=ssl
```

Figure 57 : configuration de courriel dans le fichier .env

La dernière étape à effectuer pour avoir accès à la messagerie est d'autoriser, dans celle-ci, l'accès à des applications externes. Pour le faire, il faut aller dans les paramètres du compte Google, puis dans l'onglet « sécurité » et enfin activer l'« Accès moins sécurisé des applications » comme sur la figure 58 ci-après (page suivante) :

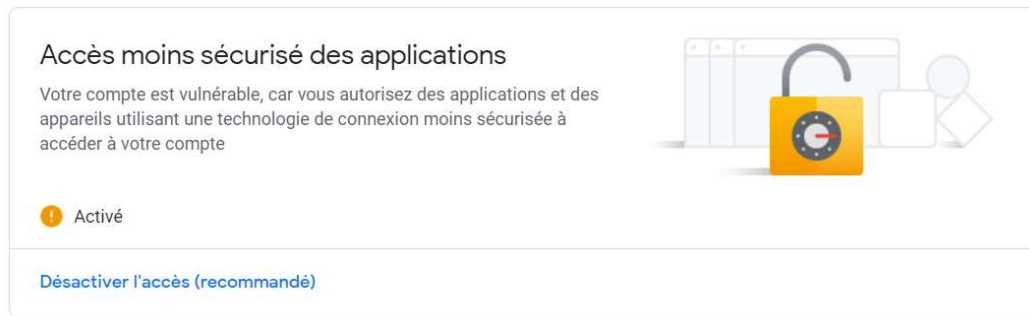


Figure 58 : autorisation d'accéder à la messagerie Gmail via des applications externes

Utilisation

L'envoi de courriel utilise la technologie de SwiftMailer, pour faire appel à cette bibliothèque, il faut appeler l'objet Mail dans le contrôleur où l'on souhaite envoyer un courriel. D'ici, on crée le contrôleur de courriel en lui passant en paramètre les données qui seront présentes dans le courriel.

```
//création et envoi d'email
Mail::to($sign->email)->send(new ConfirmationReparation($rep, $sign));
```

Figure 59 : code de création d'un courriel dans un contrôleur

On retrouve tous les contrôleurs de courriel au chemin /app/Mail. Voici un exemple de configuration d'un envoi de courriel : on indique que l'on crée un courriel en Markdown, on y indique le chemin pour accéder à la vue qui contient le corps du courriel, on peut ajouter un sujet et les données du destinataire.

```
public function build()
{
    //création de l'email
    return $this->markdown( VIEW: 'mails.confirmationRep')
        ->subject( subject: 'Réparation du vélo '.$this->signalement->reference_velo)
        ->from( address: 'quentin@kargo.bike', name: 'KargoBike');
}
```

Figure 60 : contrôleur de courriel

Concernant la vue, elle est codée en Markdown ; la différence avec du code HTML normal, c'est que le Markdown converti le texte, soit en HTML, soit en texte normal selon le langage autorisé dans les messageries électroniques. Sa structure se fait comme dans l'exemple ci-dessous : on voit que l'on peut utiliser du code PHP de la même manière que pour les autres vues.

```
@component('mail::message')
@if (!$reparation->reparation_impossible)
# Votre vélo a été réparé avec succès.
@else
# Votre vélo n'a malheureusement pas pu être réparé.
@endif
**Référence du vélo: **
[[$signalement->reference_velo]]

**Rappel du problème: **
[[$signalement->titre_probleme]]

**Titre de la réparation: **
[[$reparation->titre_reparation]]

**Détails de la réparation: **

[!!nl2br($reparation->detail_reparation)!!]

**Prix: **
[[$reparation->prix]] CHF

L'équipe KargoBike vous souhaite une excellente journée.
@endcomponent
```

Figure 61 : contenu du courriel créé en Markdown

6.2.12. Exporter en Excel

Pour exporter des données en format Excel, il suffit d'écrire la ligne suivante ; cette ligne va appeler le contrôleur d'exportation pour convertir les données en un fichier Excel.

```
return Excel::download(new ReparationExport($table), 'reparation.xlsx');
```

Figure 62 : code pour exporter des données dans un fichier Excel

De son côté, le contrôleur d'exportation va récupérer les données qui sont mises en paramètre afin d'appeler la vue qui va structurer les données dans le fichier Excel. La structure de la vue est un tableau créé en html. Ce tableau possède le titre des colonnes ainsi que le contenu des colonnes.

```
class ReparationExport implements FromView
{
    private $data;
    public function __construct($data)
    {
        $this->data = $data;
    }
    public function view(): View
    {
        return view('view:exports.reparationExcel', [
            'table' => $this->data
        ]);
    }
}
```

Figure 63 : exemple de contrôleur d'exportation

6.2.13. API

Configuration

L'API permet d'accéder aux données de notre site internet via un autre système. Dans notre cas, cela permet à des compagnies d'intégrer cela à leur système afin de signaler plus facilement un vélo qui doit être réparé.

La configuration de la route de l'API se fait dans le fichier /routes/api.php. Dans notre cas, nous utilisons une route de type « post » puisque l'on a uniquement besoin de recevoir des données provenant de cette API.

```
//route pour donner l'accès à l'API de signalement
Route::post('/signalement', 'SignalementController#create');
```

Figure 64 : route pour l'appel de l'API

Utilisation

Pour accéder de façon externe à l'API depuis un autre site internet, il faut suivre les points suivants :

- Ajouter les dépendances à Asynchronous JavaScript and XML (AJAX) et JQuery qui permettent d'envoyer des données sans devoir recharger une page. Cela nous permettra d'utiliser cette technologie dans notre code.
- Configurer la balise « form » pour que l'action corresponde à l'URL de notre API et que la méthode soit en « post ».
- Configurer chaque balise « input » avec l'attribut « name » qui correspond aux champs que nous souhaitons recevoir. Il est important de respecter cela, parce que le code de transformation va directement reprendre le nom indiqué dans la balise « input » pour convertir les données en JSON. Il faut donc utiliser les noms suivants :
 - reference
 - compagnie
 - email
 - ville
 - latitude
 - longitude
 - titre
 - detail

- Appeler, dans le bouton, la méthode JavaScript qui enverra les données en format JSON.
- Créer, dans la balise « script » la méthode qui va transformer les données dans le format JSON, puis elle enverra ces données à l'API, tout cela avec l'aide de AJAX et JQuery.

```
<!DOCTYPE html>
<html lang="en">
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js" charset="utf-8"></script>
</head>
<body>
<form action="http://kargobike-maintenance.local/api/signalement" method="post" name="myForm">

  <input type="text" id="referenceSignIn" name="reference"
    placeholder="Référence...">
  <input type="text" class="form-control" id="compagnieSignIn" name="compagnie"
    placeholder="Compagnie...">
  <input type="text" class="form-control" id="emailSignIn" name="email"
    placeholder="Email...">
  <input type="text" class="form-control" id="villeSignIn" name="ville"
    placeholder="Ville...">
  <input type="number" step="0.0000001" class="form-control" id="latitudeSignIn" name="latitude"
    placeholder="Latitude...">
  <input type="number" step="0.0000001" class="form-control" id="longitudeSignIn" name="longitude"
    placeholder="Longitude...">
  <input type="text" class="form-control" id="titreSignIn" name="titre"
    placeholder="Titre...">
  <textarea class="form-control" rows="5" id="detailSignIn" name="detail"
    placeholder="Detail..."></textarea>
  <button type="submit" class="btn btn-primary"
    style="padding-left: 60px; padding-right: 60px" onclick="submitform()">
    Envoyer
  </button>
</form>
<script type="text/javascript">
function submitform(){
  var formData = JSON.stringify($("#myForm").serializeArray());
  $.ajax({
    type: "POST",
    url: "http://kargobike-maintenance.local/api/signalement",
    data: formData,
    success: function() {},
    dataType: "json",
    contentType: "application/json"
  });
}
```

Figure 65 : exemple d'utilisation de l'API

6.2.14. Test Unitaire

Les tests unitaires sont des tests effectués sur une petite partie d'un logiciel. Leur but est de vérifier que le code fonctionne correctement. Ces tests sont très utiles dans le cas où un développeur continue à développer une fonctionnalité ou continue de modifier un système : en effet les tests signaleront un problème dans le cas où une modification a un impact négatif sur une fonctionnalité. Ces tests sont là pour vérifier qu'un logiciel est fonctionnel et utilisable. On retrouve ces tests dans Laravel au chemin /tests/unit.

Dans notre cas, nous avons utilisé les tests unitaires pour contrôler le middleware qui renvoie un utilisateur non connecté vers la page de connexion. On peut voir le test comme suit :

```
public function testAdminPage()
{
    $this->get( url: '/admin')
        ->assertStatus( status: 302);

    $this->get( url: '/admin')->assertLocation( url: '/login');
}
```

Figure 66 : tests unitaires pour la page administrateur

Dans cette situation, nous avons deux tests. Le premier vérifie, lorsque l'on va sur le chemin /admin sans être connecté, que nous recevons bien une redirection vers une autre page. En effet le status http 302 est l'indice de redirection, c'est-à-dire l'indice qui indique comme quoi nous sommes redirigés vers une autre page. Le second test, quant à lui, va vérifier que l'on soit bien redirigé sur la page de connexion.

6.3. Guide d'utilisateur

L'utilité d'un guide d'utilisateur est d'aider les personnes qui utilisent le système. Il est censé être comme un mode d'emploi pour l'utilisateur. Cela permet de faciliter la compréhension du site internet afin que les utilisateurs puissent l'utiliser de façon optimale sans avoir recours à un développeur.

Dans ce guide, nous allons parler de chaque page du site internet, ainsi que des fonctionnalités s'y trouvant.

6.3.1. Connexion

La page de connexion (ou login) est la page qui permet de se connecter à notre site internet, il suffit de rentrer les identifiants nous correspondant (nom et mot de passe) pour pouvoir accéder au site.

Trois types d'utilisateurs peuvent y accéder :

- Le client, c'est une personne qui souhaite signaler un problème sur son vélo afin que celui-ci soit réparé. Cette personne n'a accès qu'au formulaire de demande de réparation (voir le point 6.3.5 Signalement).

- L'employé, c'est l'ouvrier de KargoBike qui répare les vélos. Il a l'accès à tout le site sauf à la partie administrateur.
- L'administrateur, c'est celui qui a tous les droits. Il a accès à toutes les pages du site sans exception.

6.3.2. Accueil

Cette page est accessible par l'employé et l'administrateur.

La page d'accueil est la page qui regroupe tous les signalements des clients. Pour le faire, elle liste les signalements de deux façons différentes :

- la première est de façon cartographique ; tous les signalements sont disposés sur la carte selon leur emplacement.
- la seconde est ordonnée dans un tableau trié du signalement le plus ancien au plus récent.

The screenshot shows the 'Panel admin' interface for KargoBike. At the top, there's a 'Déconnexion' button. Below it is a map of the Sierre region with several red location pins. A popup window is open over one pin, displaying: 'Date: 2019-07-06 10:28:57', 'Référence: Pepita', 'Compagnie: Airbie', and a 'Détails' button. Below the map is a search bar with a dropdown menu labeled 'Référence du vélo' and a 'Rechercher' button. Underneath is a table with the following data:

Date de signalisation	Référence du vélo	Compagnie	Ville	Problème	
2019-07-02 10:28:57	x5625995	PubliBike	Sierre	La batterie du cadenas est faible	Détails
2019-07-04 10:28:57	Lorenzo	Airbie	Sierre	Roue et guidon détruits	Détails
2019-07-05 10:28:57	x5625997	PubliBike	Sion	Le cadenas ne répond plus	Détails

Figure 67: page d'accueil du site

La carte affiche tous les signalements mais également la géolocalisation de l'utilisateur. Il est également possible de cliquer sur un signalement afin de voir les détails de celui-ci dans un popup.

Le tableau, pour sa part, détaille les différents signalements, il est possible, depuis celui-ci, en appuyant sur le bouton de localisation du vélo, de zoomer sur la carte directement sur l'emplacement du vélo afin de pouvoir voir quel vélo se trouve à quel emplacement.

Depuis le popup dans la carte ou depuis le tableau, il est possible d'accéder à la page de détails d'un signalement en cliquant sur le bouton « détails ».

Sur cette page, il est également possible de filtrer les signalements selon plusieurs critères (référence du vélo, compagnie, ville et problèmes). Ce filtre va alors afficher uniquement ce que l'utilisateur souhaite voir sur la carte et dans le tableau.

6.3.3. Détails d'un signalement

Cette page est accessible par l'employé et l'administrateur.

On peut retrouver sur cette page tous les détails d'un signalement. On peut également voir le vélo sur la carte, ainsi qu'un historique de toutes ses réparations passées.

Karg?Bike

Détails du vélo x5625995

Date de signalisation: 2019-07-02 10:28:57
Compagnie: PubliBike
Ville: Sierre
Titre du problème: La batterie du cadenas est faible
Détail du problème: La batterie est à moins de 20%, il faudrait la recharger dans un délai de 2 semaines.

Historique de réparation

Date	Type de réparation	Prix
2019-06-29 10:28:57	Batterie du cadenas rechargée	20
2019-06-02 10:28:57	Batterie du vélo rechargée	20
2019-05-14 10:28:57	Phare avant remplacé	82.25

Réparer maintenant
Retour

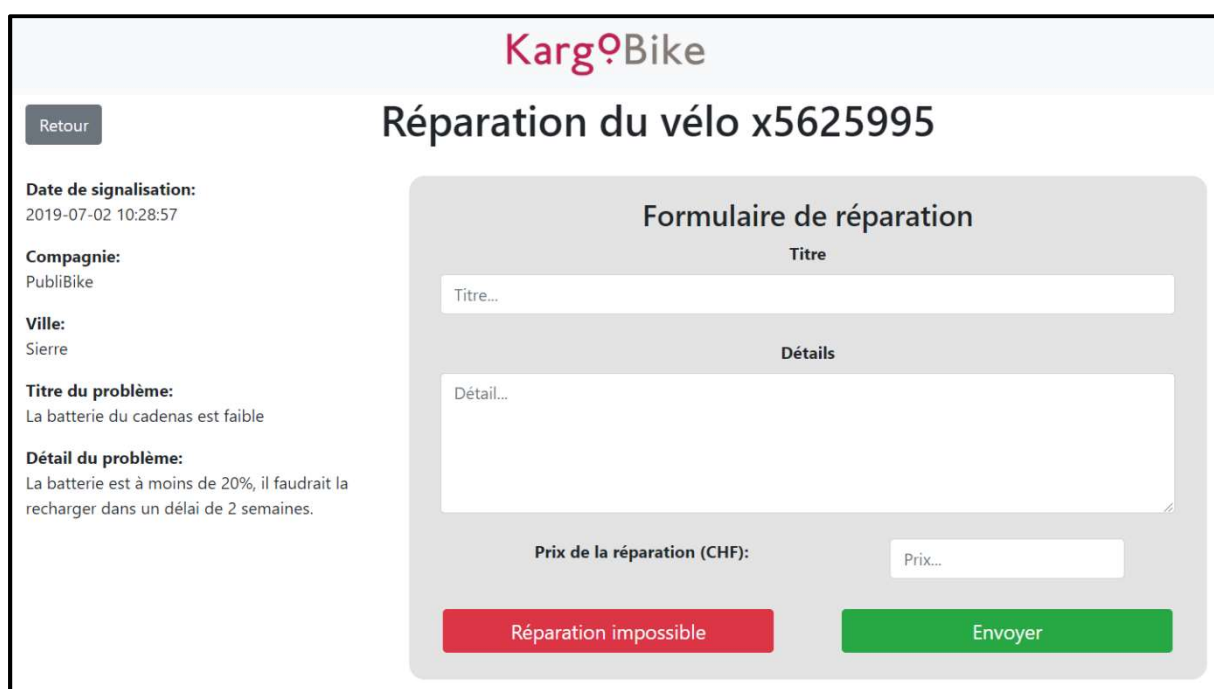
Figure 68 : page de détails d'un signalement

Depuis cette page on peut accéder au formulaire de réparation du vélo en cliquant sur le bouton « Réparer maintenant ».

6.3.4. Réparation

Cette page est accessible par l'employé et l'administrateur.

La page de réparation est celle qui permet de signaler qu'un vélo a bien été réparé. On retrouve dans un premier temps un rappel du signalement du vélo et ensuite le formulaire de réparation.



Kargobike

Réparation du vélo x5625995

Retour

Date de signalisation:
2019-07-02 10:28:57

Compagnie:
PubliBike

Ville:
Sierre

Titre du problème:
La batterie du cadenas est faible

Détail du problème:
La batterie est à moins de 20%, il faudrait la recharger dans un délai de 2 semaines.

Formulaire de réparation

Titre
Titre...

Détails
Détail...

Prix de la réparation (CHF):
Prix...

Réparation impossible **Envoyer**

Figure 69 : page de réparation

Chaque champ de ce formulaire doit être rempli impérativement. Il est également possible de signaler comme quoi un vélo a bien été réparé ou s'il est irréparable.

Une fois le formulaire rempli et envoyé, un courriel de confirmation de réparation est envoyé à la personne qui avait signalé ce vélo afin qu'il soit informé du résultat de la réparation.

6.3.5. Signalement

Cette page est accessible par tous les utilisateurs.

La page dédiée au signalement est celle qui permet aux clients de signaler leurs vélos qui ont un problème.

Figure 70 : formulaire de réparation

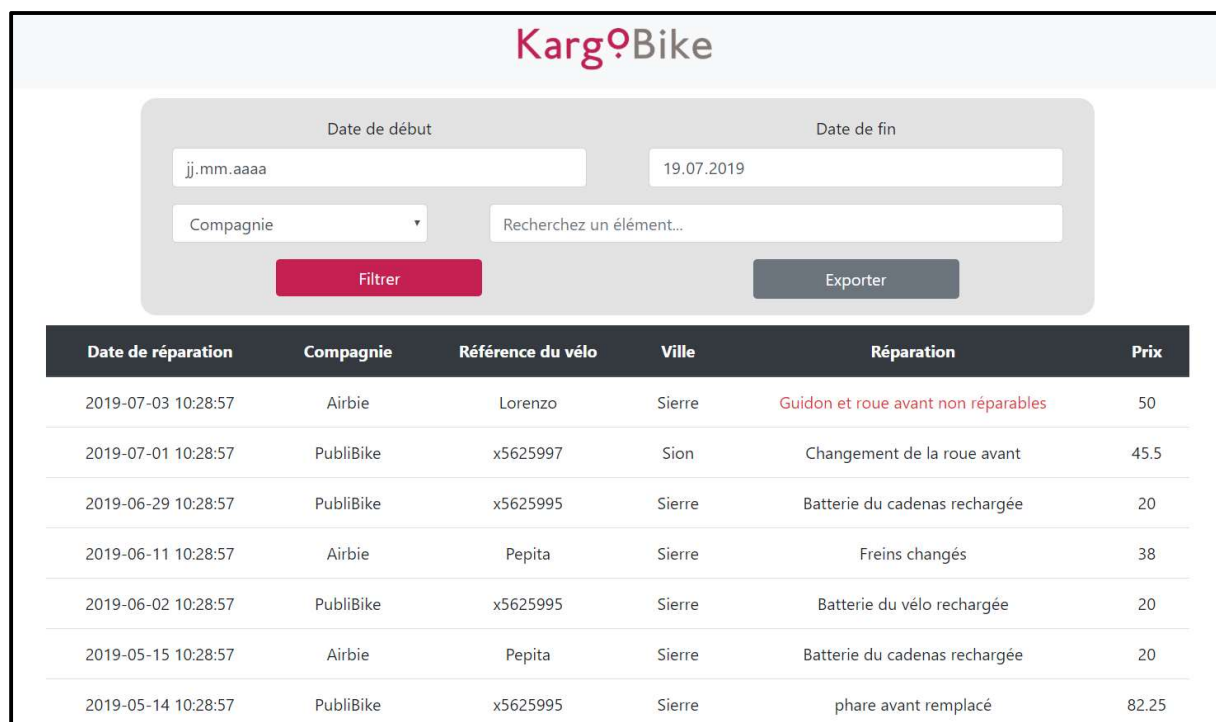
Cette page est sous forme de formulaire. Chaque champ de ce formulaire doit être impérativement rempli.

Une fois le formulaire envoyé, un courriel est envoyé à tous les employés, leur notifiant qu'un nouveau signalement a été ajouté.

6.3.6. Administration

Cette page est accessible uniquement par l'administrateur.

La partie administrative du site affiche toutes les réparations effectuées sur des vélos. Ces réparations sont listées de la plus récente à la plus ancienne.



The screenshot shows the Kargobike administration interface. At the top, there's a header with the logo. Below it, a filter form allows users to search by date range (Date de début, Date de fin), company (Compagnie), and a search bar (Recherchez un élément...). There are buttons for 'Filtrer' and 'Exporter'. Below the form is a table listing repairs with columns for Date de réparation, Compagnie, Référence du vélo, Ville, Réparation, and Prix.

Date de réparation	Compagnie	Référence du vélo	Ville	Réparation	Prix
2019-07-03 10:28:57	Airbie	Lorenzo	Sierre	Guidon et roue avant non réparables	50
2019-07-01 10:28:57	PubliBike	x5625997	Sion	Changement de la roue avant	45.5
2019-06-29 10:28:57	PubliBike	x5625995	Sierre	Batterie du cadenas rechargée	20
2019-06-11 10:28:57	Airbie	Pepita	Sierre	Freins changés	38
2019-06-02 10:28:57	PubliBike	x5625995	Sierre	Batterie du vélo rechargée	20
2019-05-15 10:28:57	Airbie	Pepita	Sierre	Batterie du cadenas rechargée	20
2019-05-14 10:28:57	PubliBike	x5625995	Sierre	phare avant remplacé	82.25

Figure 71 : page d'administration du site

Le tableau affiche toutes les réparations ainsi que leurs détails. Si un vélo avait été signalée comme irréparable, le texte de la réparation est en rouge pour l'indiquer.

Dans cette page, il est également possible de filtrer les réparations selon la date et selon des critères (compagnie, référence du vélo, ville, réparation et prix). Il est également possible d'exporter ces données en format Excel en utilisant les mêmes champs que pour le filtre.

7. Bilan

7.1. Difficultés rencontrées

Les difficultés qu'on a pu rencontrer durant ce travail de Bachelor furent toutes centrées sur le même point qui fut de choisir la technologie à développer. En effet tout au long du travail, le thème a changé de nombreuses fois ; passant d'une application de vélos en libre-service, à un partenariat avec Airbie pour compléter leur service et pour finir avec un outil de maintenance de vélos en libre-service. Tout cela sans parler des idées qui ont été évoquées entretemps, qui ont demandé un peu de recherches et qui n'ont pour finir pas été développées (partenariat avec Linka, avec Donkey Republic, etc.).

Il a fallu à plusieurs reprises repartir, entre guillemets, à zéro lorsque l'on partait sur une nouvelle idée. Tout cela n'était pas forcément simple à gérer, vu l'incertitude concernant le choix final de la technologie à développer et de la durée toujours plus restreinte du travail de Bachelor.

Mais dès lors que le choix final a été effectué et qu'il a été question de partir sur un outil de maintenance, la mise en œuvre a été rapide et le projet a pu s'élaborer dans d'excellentes conditions.

7.2. Améliorations futures

Le développement du travail s'est effectué en respectant les priorités des User Stories du Product Backlog. Mais vu le temps imparti pour le travail de Bachelor, toutes les fonctionnalités souhaitées n'ont pu être réalisées. Cette section va parler des fonctionnalités qui seraient à développer dans le futur.

7.2.1. Statistique du site internet

Une demande du client était d'avoir accès à des statistiques du site basées sur la productivité de KargoBike. C'est-à-dire des statistiques tels que le temps d'attente moyen pour qu'un vélo soit réparé ou alors le nombre de vélos réparés sur une certaine période donnée.

Bien sûr toutes ses statistiques seraient uniquement présentes du côté administrateur.

7.2.2. Facturation automatique

Un autre souhait du client était d'avoir un système qui crée directement les factures selon des critères prédéfinis tels que le total des coûts des réparations d'un client ou d'un vélo pour une période donnée. Cette facture serait générée, par après, en format PDF.

Cette fonctionnalité serait intégrée au panel administrateur et permettrait un gain de temps à KargoBike pour tout le système de facturation.

7.2.3. Optimisation du formulaire de signalisation

Le dernier point à développer serait l'optimisation du formulaire actuel de signalement d'un vélo. En effet, actuellement pour signaler un vélo, il faut rentrer manuellement la latitude et la longitude la localisation du vélo. Cependant, il serait possible de récupérer directement la géolocalisation de la personne ou/et d'intégrer une carte où il est alors possible de cliquer dessus pour indiquer son emplacement.

7.2.4. Ouverture vers un marché de vélos privés

Une des idées du client est d'ouvrir, par après, l'accès au signalement de vélos à réparer à tout le monde. Actuellement ce logiciel est créé pour fournir un service à des systèmes de vélos en libre-service, mais le but, à long terme, serait de permettre à un tout à chacun de pouvoir demander une réparation pour son propre vélo.

Cette ouverture demanderait de créer un système d'enregistrement pour le login ainsi que quelques modifications de champs du formulaire de signalement afin de permettre l'accès à n'importe qui au système.

Conclusion

Ce projet a permis de développer un outil de maintenance de vélos. Cet outil répond en grande partie à la majorité des besoins du client bien que certains points restent à être développés.

Tout au long de notre travail, nous avons pu constater que de nombreux systèmes de vélos en libre-service existaient sur le marché et que développer un tel système supplémentaire n'était peut-être pas des plus judicieux. C'est pourquoi ce travail a exigé de nous remettre en question pour voir qu'elles étaient les fonctionnalités manquantes de ces services. Un point qui revenait souvent était d'avoir un système pour entretenir ces flottes de vélos en libre-service. C'est comme cela que l'idée d'un outil de maintenance a vu le jour pour répondre à ce besoin réel de la plupart des systèmes actuels de vélos en libre-service.

Ce nouvel outil devrait permettre à KargoBike de vendre un service de réparation à plusieurs entreprises de vélos en libre-service. En effet cela permettra à ces entreprises de ne plus s'occuper de la maintenance des vélos qui est souvent compliquée et coûteuse, tout en dédiant ce travail à une seule entreprise qui fournira ce service. Ce projet permet de fournir un service centralisé de réparation de vélos. En effet, avant, chaque entreprise devait se débrouiller chacune de son côté pour pallier ce problème alors que, grâce à cet outil, tout est centralisé dans un même système.

Ce travail de Bachelor m'a offert l'opportunité de développer mes connaissances en PHP et Laravel. Il m'a également permis de développer mes connaissances sur le développement d'un projet de A à Z, c'est-à-dire de l'idée de début au résultat final entre guillemets, en passant par toutes les étapes de réflexion pour la conception d'un tel projet. Ce projet m'a également donné l'occasion de développer le contact développeur-client. En effet, cela m'a montré comment la discussion entre les deux parties étaient importantes et qu'il fallait parfois faire certains sacrifices pour le bon déroulement d'un projet.

Bien que cet outil de maintenance n'en soit encore qu'au stade de développement, il n'est pas impossible de le voir dans quelques années comme un outil important pour les systèmes de vélos en libre-service.

Références

Airbie. (2019). *Smart Bike Lock*. Consulté le mai 29, 2019, sur

Airbie:<http://airbie.io/fr/description-2/>

Auffray, C. (2018, septembre 4). *Chiffres clés : les OS pour smartphones*. Consulté le juin 12,

2019, sur ZDNet: <https://www.zdnet.fr/actualites/chiffres-cles-les-os-pour-smartphones-39790245.htm>

Cadeau, X. (2018, février 13). *En 1976, La Rochelle voulait déjà réduire son trafic automobile*.

Consulté le juin 18, 2019, sur Weelz: <https://www.weelz.fr/fr/la-rochelle-1976-velo-jaune-libre-service-automobile-pollution/>

Castelo, A. (s.d.). *Laravel API Tutorial: How to Build and Test a RESTful API*. Consulté le juillet

12, 2019, sur TopTal: <https://www.toptal.com/laravel/restful-laravel-api-tutorial>

Chrzanowska, N. (2017, septembre 7). *Node.js vs. PHP: Which Environment To Choose For Your Next Project?*

Consulté le juin 26, 2019, sur Netguru: <https://www.netguru.com/blog/nodejs-vs-php>

Dabi-Schwebel, G. (2014, octobre 23). *Open Source (logiciel)*. Consulté le juin 27, 2019, sur 1

min 30: <https://www.1min30.com/dictionnaire-du-web/open-source-logiciel>

de Préval, G. (2017, mai 4). *Petite histoire des vélos en libre-service*. Consulté le juin 18, 2019,

sur La Croix: <https://www.la-croix.com/Economie/Petite-histoire-velos-libre-service-2017-05-04-1200844504>

Développement front-end et back-end : Quelles différences? (2018, juillet 13). Consulté le

juin 27, 2019, sur Les jeudis: <https://blog.lesjeudis.com/developpement-front-end-et-back-end-queelles-differences>

Donkey Republic. (2019). *Donkey Republic*. Consulté le mai 12, 2019, sur Donkey Republic:

<https://www.donkey.bike/fr/>

Donkey Republic. (s.d.). *Villes*. Consulté le juillet 1, 2019, sur Donkey Republic:

<https://www.donkey.bike/fr/villes/>

- Easy Flash Messages for Your Laravel App.* (2018, juin 20). Consulté le juillet 13, 2019, sur Github: <https://github.com/laracasts/flash>
- Fox Tech. (2019). *OTO Hunter*. Consulté le mai 12, 2019, sur Fox Tech: <http://www.fox-tech.co/OTOHunter/>
- Franck. (2015, mars 30). *Pourquoi choisir Django ?* Consulté le juin 27, 2019, sur a2h Conseils: <http://blog.a2h-conseils.com/weblog/2015/03/30/pourquoi-choisir-django/>
- Gribaumont, C. (2019, mai 22). *Administrez vos bases de données avec MySQL*. Consulté le juillet 12, 2019, sur OpenClassrooms: <https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql>
- Hajir, Z. E. (2018, septembre 13). *Qu'est ce qu'Apache ? Une description complète du Serveur Web Apache*. Récupéré sur Hostinger: <https://www.hostinger.fr/tutoriels/quest-ce-quapache-serveur-web-apache/>
- Hansen, S. (2017, mai 23). *Advantages and Disadvantages of Django*. Consulté le juin 27, 2019, sur Hackernoon: <https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5>
- Hicham, M. (2017, septembre 21). *Comment utiliser la nouvelle fonctionnalité API resource dans Laravel 5.5*. Consulté le juin 27, 2019, sur <https://blog.medhicham.com/fr/blog-fr/comment-utiliser-la-nouvelle-fonctionnalite-api-resource-dans-laravel-5-5.html>
- Hory, J.-F. (2015, octobre 31). *Présentation et installation de XAMPP pour Windows*. Consulté le juillet 12, 2019, sur SupInfo: <https://www.supinfo.com/articles/single/1387-presentation-installation-xampp-windows>
- Interface de programmation : API ou Application Programming Interface.* (2019, janvier 20). Consulté le juin 8, 2019, sur Journal du Net: <https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203559-api-application-programming-interface-definition-traduction/>
- Intermobility SA. (2016). *velospot® - Le système novateur de vélos en libre-service « Swiss made »*. Récupéré sur

https://www.mobilservice.ch/admin/data/files/news_section_file/file/3762/salon_du_velo_2016_velospot_pa_sarrasin.pdf?lm=1458665156

KargoBike. (s.d.). Consulté le mai 28, 2019, sur KargoBike: <https://kargo.bike>

Kasimov, N. (2017, mars 19). *À quoi sert le framework Django ?* Consulté le juin 27, 2019, sur Quora: <https://fr.quora.com/À-quoi-sert-le-framework-Django>

Laravel – Pourquoi choisir ce framework pour développer votre MVP ? (2018, août 6). Consulté le juin 27, 2019, sur Startup-Bootcamp: <https://startup-bootcamp.fr/laravel-developper-mvp/>

Laurent, S. (2016, novembre 4). *Développement: faut-il se tourner vers Node.js à l'avenir?* Consulté le juin 26, 2019, sur Alioze: <https://www.alioze.com/developpement-nodejs-2017>

Lee, E. (2016, Juillet 7). *This Ex-Uber Exec Is Creating China's Uber For Bicycles*. Consulté le mai 7, 2019, sur Technode: <https://technode.com/2016/07/07/mobike-uber/>

Linka. (2019). Consulté le mai 17, 2019, sur Linka: <https://www.linkalock.com>

Linka. (2019). *Fitting Guide*. Consulté le mai 15, 2019, sur Linka: <https://www.linkalock.com/pages/fitting-guide-linka-leo#linka-leo-1>

Linka. (2019). *Linka Leo*. Consulté le mai 11, 2019, sur Linka: <https://www.linkalock.com/pages/linka-leo>

Linka. (2019). *Original Linka*. Consulté le mai 11, 2019, sur Linka: <https://www.linkalock.com/pages/original-linka>

Maatwebsite. (s.d.). *Laravel Excel*. Récupéré sur <https://laravel-excel.com>

Mobike. (2018). *Qu'est ce que Mobike ?* Consulté le mai 6, 2019, sur Mobike: <https://mobike.com/fr/about>

Mobile Operating System Market Share Switzerland. (2019, mai). Consulté le juin 12, 2019, sur StatCounter: <http://gs.statcounter.com/os-market-share/mobile/switzerland>

Mockup. (s.d.). Consulté le juin 4, 2019, sur infowebmaster: <http://glossaire.infowebmaster.fr/mockup/>

Murray, C. (2017, juillet 24). *Android vs iPhone: The Best Platform to Develop Your App*.

Consulté le juin 12, 2019, sur AppInstitute: <https://appinstitute.com/android-vs-iphone/>

Nebra, M. (2017, novembre 29). *Rédigez en Markdown !* Consulté le juillet 12, 2019, sur

OpenClassrooms: <https://openclassrooms.com/fr/courses/1304236-redigez-en-markdown>

Nebra, M. (2018, décembre 18). *Node.js : mais à quoi ça sert ?* Consulté le juin 26, 2019, sur

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

Nokē. (2019). *API Integration*. Consulté le mai 8, 2019, sur Nokē: <https://noke.com/api-integration>

Nokē. (2019). *Mobile app*. Consulté le mai 8, 2019, sur nokē: <https://noke.com/mobile-app>

Oamkumar, R. (2018, mars 27). *Advantages and Disadvantages of Laravel*. Consulté le juin

27, 2019, sur Software Developer India: <https://www.software-developer-india.com/advantages-and-disadvantages-of-laravel/>

Pourquoi utiliser le framework Laravel ? (2018). Consulté le juin 27, 2019, sur Webilio:

<https://webilio.ca/pourquoi-utiliser-le-framework-laravel/>

Pro Vélo Jura. (2013). *Vélos en libre service (VLS) - Questionnaire 2013*. Récupéré sur Pro

Vélo Jura: <http://www.provelojura.ch/velos-en-libre-service-vls-questionnaire-2013.html>

PubliBike. (2019). *Abonnements*. Consulté le mai 7, 2019, sur PubliBike:

<https://www.publibike.ch/fr/publibike/pricing/>

PubliBike. (2019). *Comment ça marche*. Consulté le mai 7, 2019, sur PubliBike:

<https://www.publibike.ch/fr/publibike/how-it-works/>

Quelle est la différence entre développement back end, front end, et full stack ? (s.d.).

Consulté le juin 27, 2019, sur Némésis Studio: <https://www.nemesis-studio.com/quelle-est-la-difference-entre-developpement-back-end-front-end-et-full-stack/>

Qu'est-ce que Django? (s.d.). Consulté le juin 27, 2019, sur Django Girls Tutorial:

<https://tutorial.djangogirls.org/fr/django/>

Rarchaert, G. (2017, octobre 21). *Node.js*. Consulté le juin 26, 2019, sur Supinfo -

International University: <https://www.supinfo.com/articles/single/6191-nodejs#idm46133027673312>

Ravalet, E., & Bussière, Y. (2012). *Recherche Transports Sécurité: Les systèmes de vélos en libre-service expliquent-ils le retour du vélo en ville ?* Springer-Verlag. Récupéré sur

<https://link.springer.com/article/10.1007/s13547-011-0020-6>

Sciara, E., & Cardoso, K. (2018). *Comment démarrer avec les Story Points*. Consulté le juin 4, 2019, sur Xebia Blog: <https://blog.xebia.fr/2018/05/31/comment-demarrer-avec-les-story-points/>

Sebastien. (2016, septembre 16). *Les 5 cadenas connectés qui vous empêchent de chercher vos clés*. Consulté le mai 8, 2019, sur Objects Connectés:

<https://www.objetconnecte.net/top-5-cadenas-connectes-160916/>

Sfez, J. (2017, août 30). *6 raisons de choisir Node.js*. Consulté le juin 26, 2019, sur smooth code: <https://www.smooth-code.com/articles/6-raisons-de-choisir-node-js>

Sidorenko, V. (s.d.). *The Advantages And Disadvantages of Using Django*. Consulté le juin 27, 2019, sur DataFloq: <https://datafloq.com/read/advantages-and-disadvantages-of-using-django/3050>

Soula, C. (2017, octobre 19). *Ofo : le vélo chinois qui ringardise méchamment le Vélib*.

Consulté le mai 7, 2019, sur L'OBS:

<https://www.nouvelobs.com/economie/20171013.OBS5979/ofo-le-velo-chinois-qui-ringardise-mechamment-le-velib.html>

Systèmes en Suisse. (s.d.). Consulté le mai 29, 2019, sur Forum Bikesharing Suisse:

<https://www.bikesharing.ch/fr/systemes-en-suisse/>

The Bike-sharing World Map. (2019, juin). Consulté le juin 18, 2019, sur Google My Maps:

<http://www.bikesharingmap.com/>

- Transitec ingénieurs-conseils SA. (2009). *Vélos en libre-service en Suisse: harmonisation des systèmes d'accès*. Lausanne. Récupéré sur
https://www.velokonferenz.ch/download/pictures/75/h4gpvo2pvwjcqvxo8eryyvkhd2rr0/velokonferenz_veloverleih_09.pdf
- ValaisRoule. (s.d.). Consulté le mai 7, 2019, sur ValaisRoule - Wallisrollt:
<https://wallisrollt.ch/>
- ValaisRoule. (s.d.). *En bref*. Consulté le mai 7, 2019, sur ValaisRoule - WallisRollt:
<https://wallisrollt.ch/fr/home/ueber-uns/>
- Viennet, R. (2012, mai 4). *CarPostal Suisse dessine un réseau national de vélos en libre-service*. Consulté le juin 18, 2019, sur MobiliCités: <http://www.mobilicites.com/011-1398-CarPostal-Suisse-dessine-un-reseau-national-de-velos-en-libre-service.html>
- Wang, Y. (2017, Août 10). *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems*. Consulté le juillet 12, 2019, sur Medium:
<https://medium.com/@yangforbig/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems-afd5afd6566>

Références des figures

- Figure 1 aperçue sur : <https://bajau.com/v7/What-we-do/Consultancy/DevOps/Watfall-vs-Agile/>
- Figure 2 aperçue sur : <https://www.visual-paradigm.com/scrum/why-fixed-length-of-sprints-in-scrum/>
- Figure 3 aperçue sur : <https://www.linkedin.com/pulse/rédiger-une-user-story-cest-simple-mais-difficile-dominique-houdier/>
- Figure 4 aperçue sur : https://medium.com/@AUTONOMY_95370/bike-sharing-on-steroids-862c625b5d5e
- Figure 5 aperçue sur : <http://transports.blog.lemonde.fr/2016/04/11/a-la-rochelle-les-militants-du-velo-imaginent-la-societe-de-la-voiture/>
- Figure 6 aperçue sur : <http://www.bikesharingmap.com/>
- Figure 7 aperçue sur : <http://www.bikesharingmap.com/>
- Figure 8 aperçue sur : <https://www.publibike.ch/fr/publibike/>
- Figure 9 aperçue sur : <https://mobike.com/wp-content/themes/mobike/2017/assets/themes/moby/img/logo.svg>
- Figure 10 aperçue sur : <http://www.ofo.com>
- Figure 11 aperçue sur : <https://wallisrollt.ch/fr/>
- Figure 12 aperçue sur : <https://nordiceye.com/portfolio/donkey-republic/>
- Figure 13 aperçue sur : <https://www.investiere.ch/airbie/jobs>
- Figure 14 aperçue sur : https://shop.sanitas.com/noke-kabelset-fur-velo.html?__store=sanitas_fr&__from_store=sanitas_it
- Figure 15 aperçue sur : <https://www.linkafleets.com/bikes>
- Figure 16 aperçue sur : <https://partners.sigfox.com/products/oto-hunter-rc3>
- Figure 17 aperçue sur : Photo de l'auteur

Figure 18 aperçue sur :	Schéma de l'auteur créé avec PowerPoint
Figure 19 aperçue sur :	https://www.techadvisor.fr/tutoriel/telephones/passer-android-vers-ios-3657860/
Figure 20 aperçue sur :	https://www.casinonewsdaily.com/articles/mobile-gambling/ios-compatible-mobile-casinos/
Figure 21 aperçue sur :	https://appinstitute.com/android-vs-iphone/
Figure 22 aperçue sur :	http://gs.statcounter.com/os-market-share/mobile/switzerland
Figure 23 aperçue sur :	https://nodejs.org/en/about/resources/
Figure 24 aperçue sur :	https://github.com/voodootiki-god/logo.js/blob/1544bdeed/js.svg
Figure 25 aperçue sur :	https://medium.com/@vsvaibhav2016/django-python-web-framework-project-setup-in-on-windows-part-1-957134455a58
Figure 26 aperçue sur :	www.python.org
Figure 27 aperçue sur :	https://www.ambient-it.net/formation/formation-laravel/
Figure 28 aperçue sur :	https://www.php.net/download-logos.php
Figure 29 aperçue sur :	https://sql.sh/1313-pagination-sql_calc_found_rows/mysql-square-206
Figure 30 aperçue sur :	https://www.clubic.com/telecharger-fiche70674-xampp.html
Figure 31 aperçue sur :	https://www.apache.org/foundation/press/kit/
Figure 32 aperçue sur :	https://blog.openstreetmap.org/2011/05/13/new-openstreetmap-logo/
Figure 33 aperçue sur :	https://leafletjs.com
Figure 34 aperçue sur :	https://daveismyname.blog/sending-html-emails-attachments-using-php-swiftmailer

Figure 35 aperçue sur :	https://www.iconfinder.com/icons/298823/markdown_icon
Figure 36 aperçue sur :	https://fuzati.com/technology/bootstrap-logo/
Figure 37 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 38 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 39 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 40 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm et modifier dans PowerPoint
Figure 41 aperçue sur :	Capture d'écran de l'auteur prise dans Visual Paradigm
Figure 42 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 43 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 44 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 45 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 46 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 47 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 48 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 49 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 50 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 51 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 52 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 53 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 54 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm et modifiée dans PowerPoint
Figure 55 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 56 aperçue sur :	Capture d'écran de l'auteur prise dans Gmail
Figure 57 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm

Figure 58 aperçue sur :	Capture d'écran prise dans https://myaccount.google.com/security
Figure 59 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 60 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 61 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 62 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 63 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 64 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 65 aperçue sur :	Capture d'écran de l'auteur prise dans Atom
Figure 66 aperçue sur :	Capture d'écran de l'auteur prise dans PhpStorm
Figure 67 aperçue sur :	Capture d'écran de l'auteur prise sur le site internet du projet
Figure 68 aperçue sur :	Capture d'écran de l'auteur prise sur le site internet du projet
Figure 69 aperçue sur :	Capture d'écran de l'auteur prise sur le site internet du projet
Figure 70 aperçue sur :	Capture d'écran de l'auteur prise sur le site internet du projet
Figure 71 aperçue sur :	Capture d'écran de l'auteur prise sur le site internet du projet
Figure 72 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 73 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 74 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 75 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 76 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 77 aperçue sur :	Mockup de l'auteur créé sur https://www.canva.com
Figure 78 aperçue sur :	Capture d'écran de l'auteur prise dans Excel
Figure 79 aperçue sur :	Capture d'écran de l'auteur prise dans Excel
Figure 80 aperçue sur :	Capture d'écran de l'auteur prise dans Excel

Annexes I : Mockups du premier système de BikeSharing

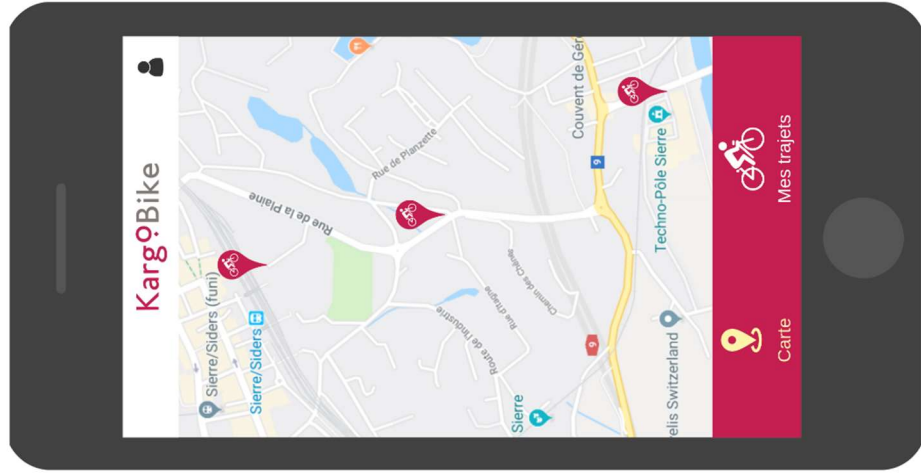


Figure 73 : page d'accueil de l'application de vélos en libre-service

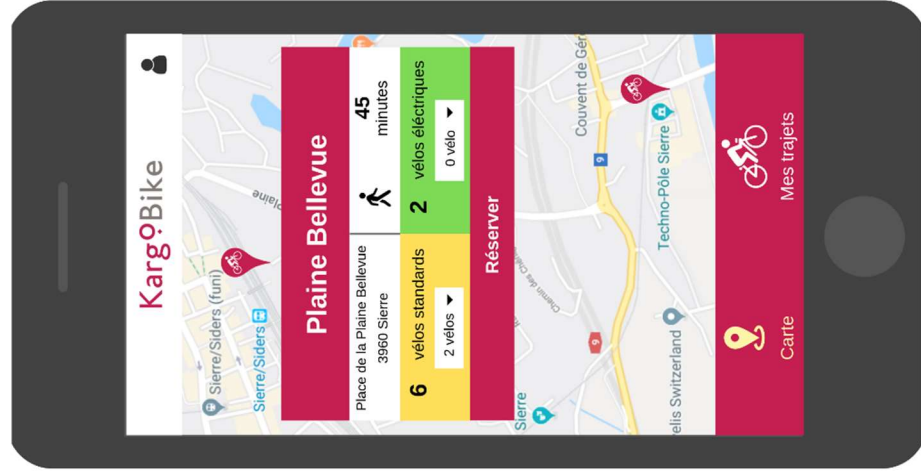


Figure 72 : popup de réservation d'un vélo

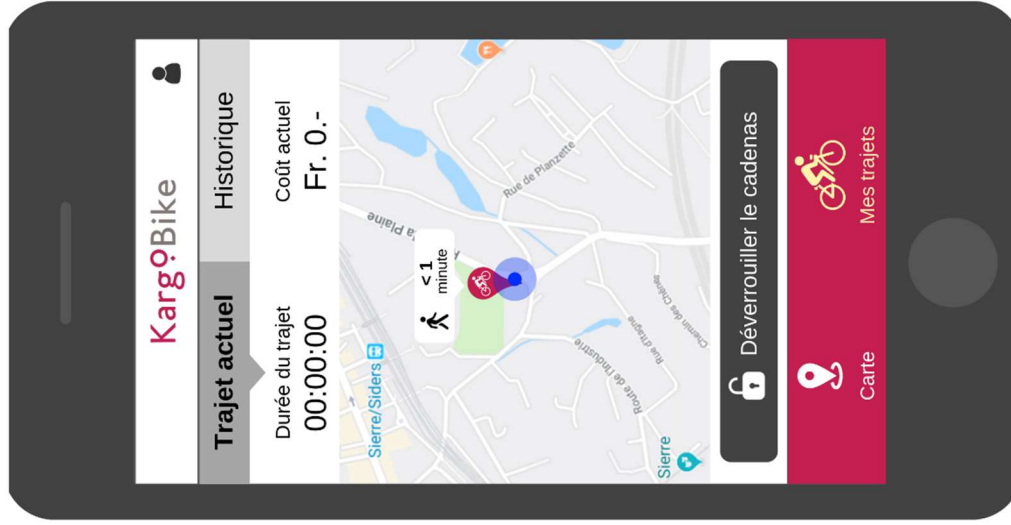


Figure 75 : localisation du vélo réservé

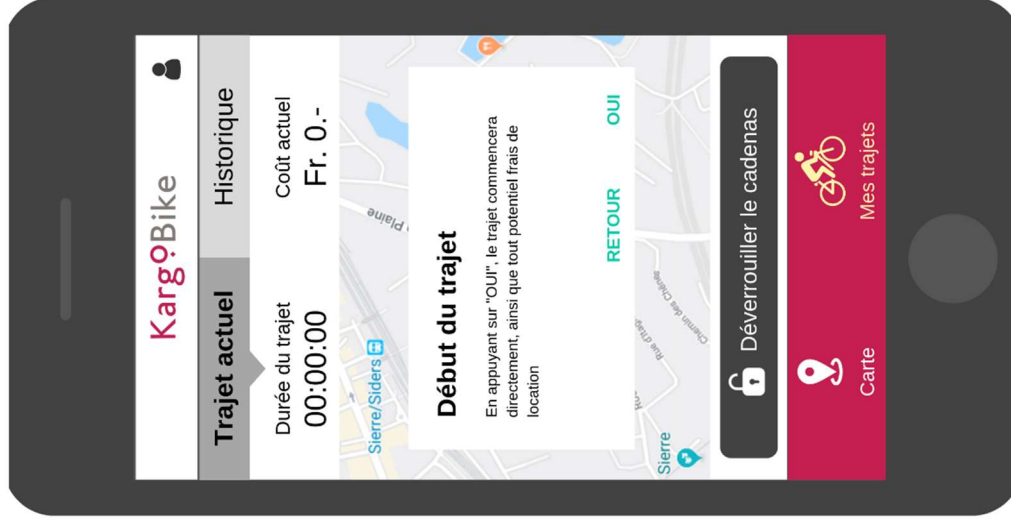


Figure 74 : début d'un trajet

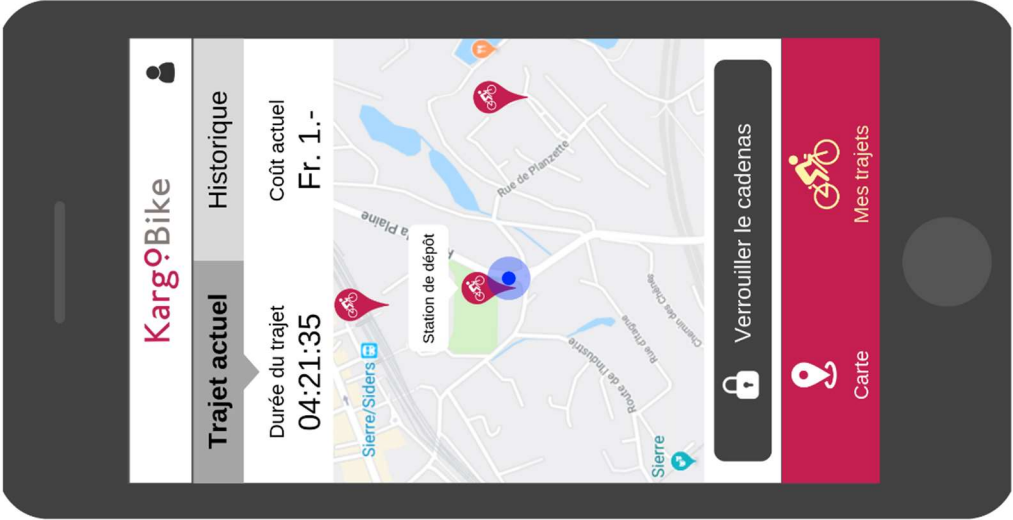


Figure 77 : trajet en cours

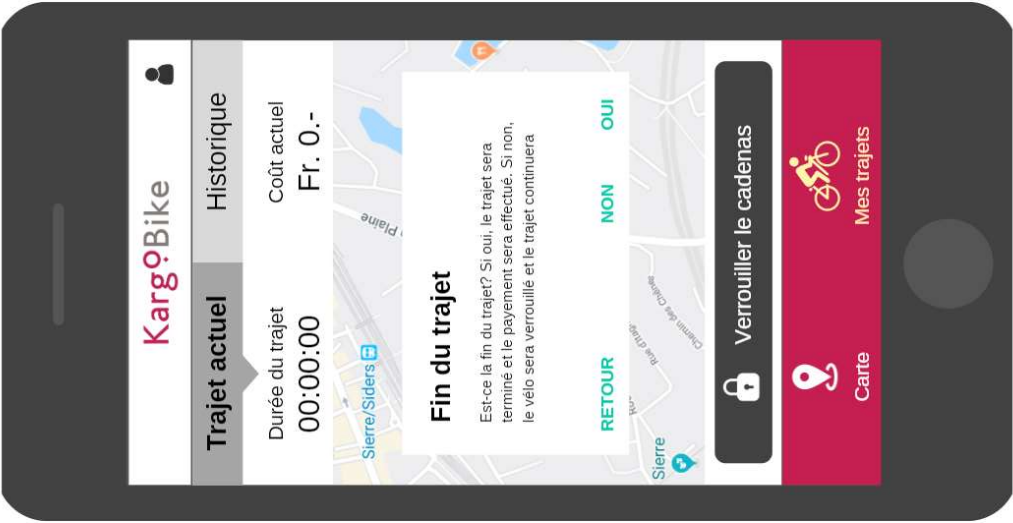


Figure 76 : fin de trajet

Annexes II : Product Backlog du premier système

US Nr.	Thème	As an/a ...	User Stories		so that ...	Acceptance Criteria	Priority	Status	Story Points	MosCoW
1	Préparation	Développeur	préparer l'environnement de travail		avoir un environnement prêt pour commencer le projet	environnement fonctionnel + connaître comment l'utiliser	1000	<div></div>	8	Must
2	Recherche	Développeur	dessiner les mockups		avoir une meilleure vue d'ensemble de l'application		990	<div></div>	5	Must
10	Application	Utilisateur	déverrouiller le vélo et commencer un trajet		pouvoir l'utiliser		950	<div></div>	5	Must
11	Application	Utilisateur	verrouiller le vélo		pouvoir faire des pauses durant mon trajet		950	<div></div>	3	Must
3	Application	Utilisateur	créer un profil sur l'application		pouvoir accéder à tout le contenu	enregistrement fonctionnel + être compatible et respecter la loi des protections des données	940	<div></div>	5	Must
12	Application	Utilisateur	clorre mon trajet		mettre fin au trajet et savoir combien m'a coûté la location		940	<div></div>	3	Must
4	Application	Utilisateur	me logger sur l'application		pouvoir louer un vélo	login fonctionnel	940	<div></div>	2	Must
5	Application	Utilisateur	visualiser les informations de mon profil		avoir une vision globale des informations entrées dans l'application		920	<div></div>	1	Must
6	Application	Utilisateur	modifier mon profil		pouvoir rentrer toutes mes informations personnelles	toutes mes informations sont enregistrées	920	<div></div>	3	Must
7	Application	Utilisateur	visualiser sur la carte tous les vélos disponibles à proximité		voir à quel temps de trajet se trouvent les autres vélos		900	<div></div>	13	Must
9	Application	Utilisateur	réserver un vélo		pouvoir aller le récupérer		900	<div></div>	8	Must
8	Application	Utilisateur	avoir des détails sur le vélo sélectionné sur la carte		choisir quel vélo correspond à mes besoins	visualiser quel est le type de vélo	800	<div></div>	5	Must
13	Application	Utilisateur	avoir le détail de mon trajet actuel		voir le temps de trajet actuel et le coût provision		700	<div></div>	5	Must
16	Application	Utilisateur	voir l'historique de mes trajets		avoir une vue d'ensemble de mon utilisation		600	<div></div>	3	Should
14	Application	Utilisateur	signaler un vélo		avoir des vélos de bonne qualité constamment		500	<div></div>	3	Should
24	Website	Administrateur	voir les statistiques de l'application		avoir une vue d'ensemble de l'application		500	<div></div>	13	Should

Figure 78 : Product Backlog du premier système – Partie 1

US Nr.	Thème	As an/a ...	User Stories		so that ...	Acceptance Criteria	Priority	Status	Story Points	MoSCoW
26	Website/app	Administrateur	ajouter un vélo		de répondre aux besoins du client		500	<div></div>	5	Should
15	Application	Utilisateur	pouvoir changer de langues		pouvoir voir plusieurs langues différentes	possibilité de changer de langue	450	<div></div>	3	Should
17	Application	Loueur	pouvoir ajouter mon vélo		pouvoir le mettre à disposition pour d'autres personnes		400	<div></div>	5	Should
18	Application	Loueur	modifier les informations de mon vélo		avoir des informations à jour		400	<div></div>	3	Should
19	Application	Loueur	pouvoir indiquer les jours et horaires auxquels mon vélo sera disponible à la location		organiser la location du vélo		400	<div></div>	8	Should
20	Application	Loueur	pouvoir visualiser l'historique de location de mon vélo		avoir les détails concernant l'utilisation de mon vélo		400	<div></div>	3	Should
21	Application	Loueur	voir en direct les détails concernant le trajet actuel de mon vélo		visualiser l'emplacement de mon vélo		400	<div></div>	5	Should
22	Application	Loueur	pouvoir signaler un utilisateur		maintenir une application qui correspond à l'esprit de l'entreprise		400	<div></div>	2	Should
23	Website	Administrateur	voir le trafic en temps réel des vélos		avoir une vue d'ensemble de l'application		400	<div></div>	8	Should
25	Website	Administrateur	voir tous les signalements des utilisateurs et des loueurs		pouvoir manager au mieux la flotte de vélos et les utilisateurs		400	<div></div>	8	Should
27	Website	Administrateur	bannir un utilisateur		maintenir une application qui correspond à l'esprit de l'entreprise		400	<div></div>	3	Should
28	Application	Utilisateur	voir le niveau de batterie du vélo sélectionné		connaître l'autonomie/les incentives potentielles si recharge		400	<div></div>	5	Should
29	Website	Administrateur	voir le niveau de batterie du vélo sélectionné		suivi de la flotte et de sa disponibilité		400	<div></div>	5	Should
30	Application	Utilisateur	visualiser sur la carte les points de recharge		recharger le vélo		400	<div></div>	2	Should
31	Application	Utilisateur	accéder à ses points incentives		les utiliser		400	<div></div>	8	Should
32	Website	Administrateur	gérer les incentives		encourager les comportement positifs		400	<div></div>	8	Should
33	Application	Utilisateur	synchroniser avec Strava		Intégrer les commutes dans		400	<div></div>	21	Should
34	Application	Utilisateur	partager sur les réseaux sociaux		promotion/self promotion		400	<div></div>	8	Should
35	Cadenas	Utilisateur	déverrouiller avec swisspass				200	<div></div>	13	Could
36	Application	Utilisateur	accéder aux vélostations				100	<div></div>	3	Could
37	Application	Utilisateur	visualiser les vélostations sur la carte				100	<div></div>	2	Could
38	Application	Administrateur	transmettre des informations aux utilisateurs selon leur localisation		promouvoir des offres/partenaires/comportements		100	<div></div>	13	Would

Figure 79 : Product Backlog du premier système – Partie 2

Annexes III : Product Backlog du second système

US Nr.	Thème	As an/a ...	User Stories		so that ...	Acceptance Criteria	Priority	Status	Story Points	Sprint	US accepted (done done)	MoScow
1	Préparation	Développeur	préparer l'environnement de travail	I want to ...	avoir un environnement prêt pour commencer le projet	Environnement fonctionnel + connaître comment l'utiliser	990	●	8	1	done	Must
2	Recherche	Développeur	dessiner les mockups		avoir une meilleure vue d'ensemble de l'application		980	●	5	1	done	Must
3	Website	Employé	me logger sur la plateforme		accéder aux tâches et suivi	Login fonctionnel	970	●	3	1	done	Must
14	Website	Client	accéder à la plateforme		rentrer les données du problème	Login fonctionnel	960	●	2	1	done	Must
18	Website	Client	signaler un vélo qui doit être réparé		pouvoir faire réparer le vélo		950	●	2	1	done	Must
16	Website	Employé	visualiser la liste de tous les vélos		optimiser la charge de travail	Liste + filtre pour les villes	940	●	5	1	done	Must
9	Website	Employé	voir l'ordre par date de signalements		éviter les délais de réparation trop longs		935	●	1	1	done	Must
19	Website	Employé	filtrer les champs de recherche du tableau des signalements		voir plus en détail le travail à effectuer		933	●	3	1	done	Must
4	Website	Employé	visualiser sur la carte tous les vélos en attente d'entretien		optimiser la charge de travail	Avoir les détails en cliquant sur un marker	930	●	13	1	done	Must
5	Website	Employé	avoir des détails sur le vélo sélectionné		identifier le vélo à entretenir		920	●	5	1	done	Must
8	Website	Employé	journal d'entretien du véhicule		suivi des coûts par véhicule	Lister par date	915	●	5	1	done	Must
11	Website	Employé	valider la réparation		enregistrer les détails		910	●	5	1	done	Must
12	Website	Employé	signaler un vélo non réparable		sortir le vélo du système		900	●	1	1	done	Must
17	Website	Employé	envoyer une confirmation de réparation		pour prévenir le client de la réparation		860	●	8	2	done	Should
10	Website	Employé	être notifié d'une nouvelle tâche		éviter les délais de réparation trop longs		850	●	2	2	done	Should
6	Website	Client	pouvoir envoyer un signalement depuis mon système		assurer l'entretien	Utiliser une API	800	●	21	2	done	Should
20	Website	Administrateur	pouvoir lister tous les vélos réparés		avoir une vision globale des réparations pertinentes		750	●	3	2	done	Should
21	Website	Administrateur	filtrer le tableau des vélos réparés				700	●	5	2	done	Should
22	Website	Administrateur	exporter les données des vélos réparés		pouvoir faire la facturation		650	●	13	2	done	Should
7	Website	Administrateur	voir les statistiques de l'application		avoir une vue d'ensemble de l'activité		500	■	21			Should
13	Website	Administrateur	générer automatiquement des rapports de facturation par client/vélo/global		simplifier les tâches administratives en configurant les modalités de facturation		450	■	13			Should
23	Website	Client	pouvoir utiliser la géolocalisation ou une carte pour signaler mon vélo		rentrer les coordonnées de l'emplacement automatiquement		400	■	4			Could

Figure 80 : Product Backlog du second système (outil de maintenance)

Annexes IV : Livre de bord

Semaine	Jour	Heure	
Semaine 1	30 avril 2019	3	8
	1 mai 2019	5	
Semaine 2	6 mai 2019	6,5	24,5
	7 mai 2019	6	
	8 mai 2019	5	
	11 mai 2019	5,5	
	12 mai 2019	1,5	
Semaine 3	13 mai 2019	6	21,5
	14 mai 2019	7,5	
	15 mai 2019	8	
Semaine 4	20 mai 2019	9,5	27
	21 mai 2019	9,5	
	22 mai 2019	8	
Semaine 5	27 mai 2019	9	19
	28 mai 2019	4,5	
	29 mai 2019	5,5	
Semaine 6	3 juin 2019	6	29
	4 juin 2019	8	
	5 juin 2019	9	
	6 juin 2019	1	
	8 juin 2019	5	
Semaine 7	10 juin 2019	4,5	18
	11 juin 2019	6,5	
	12 juin 2019	7	
Semaine 8	17 juin 2019	4	17,5
	18 juin 2019	6,5	
	19 juin 2019	4	
	20 juin 2019	3	
Semaine 9	24 juin 2019	7,5	44,5
	25 juin 2019	8	
	26 juin 2019	7	
	27 juin 2019	7,5	
	28 juin 2019	5,5	
	29 juin 2019	5,5	
	30 juin 2019	3,5	

Semaine	Jour	Heure	
Semaine 10	1 juillet 2019	6,5	40,5
	2 juillet 2019	1,5	
	3 juillet 2019	9	
	4 juillet 2019	8	
	5 juillet 2019	8,5	
	6 juillet 2019	7	
Semaine 11	8 juillet 2019	7	31
	9 juillet 2019	6	
	10 juillet 2019	7	
	11 juillet 2019	4,5	
	12 juillet 2019	6,5	
Semaine 12	15 juillet 2019	7	34,5
	16 juillet 2019	6	
	17 juillet 2019	8	
	18 juillet 2019	7,5	
	19 juillet 2019	6	
Semaine 13	22 juillet 2019	8	50
	23 juillet 2019	5	
	24 juillet 2019	6	
	25 juillet 2019	8	
	26 juillet 2019	7	
	27 juillet 2019	8	
	28 juillet 2019	8	
	29 juillet 2019	4	
Semaine 14			4
		369	

Total : 369 heures

8. Déclaration de l'auteur

« Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après : Yann Bocchi, Gaël Ribordy ».

Sierre, le 30 juillet 2019

Remion Quentin