Table des matières

# Liste des tableaux

# Liste des figures

# 1. Introduction

En Suisse, notre société de consommation actuelle produit 300Kg de déchets par année par personne. Ce qui représente au total 2.3 millions de tonnes de déchets pour notre si petit pays. Ces pertes ont lieu sur toutes les étapes de la chaine, du producteur au consommateur alors si on pouvait déjà limiter ne serait-ce que 5% cela représenterait près de 115'000kg.

Je ne suis pas un écologiste acharné, mais en tant qu'étudiant je ne serais pas contre des prix plus abordables pour des produits qui vont de toutes manières finir à la poubelle dans les heures ou jours qui suivent. Cela peut permettre aux petits commerces, comme aux producteurs d'écouler leurs invendus et d'arrondir leurs fins de mois et aux consommateurs de faire des économies substantielles. On est tous gagnants.

Le projet initial pour mon travail de Bachelor vient de mon père. Nous en avions parlé lors de mon TPI de fin d'apprentissage qui avait pour intitulé « Quel avenir pour l'agriculture contractuelle de proximité » qui était déjà en relation avec la vente directe et il avait évoqué l'intérêt que pourrait susciter un site Internet pour écouler les invendus des commerçants. Avec le développement des applications mobiles, mon projet a été donc de reprendre le projet non-abouti de mon père et de l'adapter aux tendances actuelles. Les gens vivent en permanence avec leur smartphone et tout se fait maintenant au travers de celui-ci.

D'autres options pour lutter contre les invendus existent et le créneau est devenu toujours plus porteur. Le projet ne cherche pas à concurrencer les associations ou les mouvements qui essaient de lutter contre le gaspillage alimentaire mais de proposer une autre alternative. Mon application, bien qu'adaptable à tous types de commerces, est destinée principalement aux petits commerces et principalement à la vente de produits locaux.

Ce ne serait pas qu'un projet pour mon travail de Bachelor mais un projet qui me tient à cœur à moi et mon père. Celui de mettre à disposition cette application au bénéfice d'une association qui développe le commerce de proximité et par la suite, si le succès escompté est au rendez-vous la développer ailleurs en Suisse, voire dans le monde.

# 2. Le gaspillage alimentaire

Il ne faut pas confondre « déchets alimentaires » et « gaspillage alimentaire ». Par « déchets alimentaires » on entend non seulement les déchets non évitables (épluchures des fruits et légumes, les os, les coquilles d'œuf…) mais aussi les déchets évitables (tomates jetées en raison de surproduction, fromages retirés des rayons car proche de la limite de consommation, morceau de viande oubliée dans le réfrigérateur, pain rassis…).

Définition de base (d'après FAO, 1981)[1]

Est considéré comme gaspillage alimentaire la partie des produits comestibles destinés à la consommation humaine qui est jetée, perdue, dégradée ou consommée par des parasites à n'importe quelle étape de la filière agroalimentaire. (Organisation des Nations Unies pour l'alimentation et l'agriculture 2012)

Si l'on prenait toute la nourriture gaspillée dans le monde, on pourrait nourrir la planète entière. Le part de gaspillage auprès des consommateurs est très différentes en fonction des lieux de vie de ceux-ci. Plus les consommateurs vivent dans des pays pauvres moins ils gaspillent car ils vont valoriser au maximum chaque aliment. Dans les pays industrialisés, la nourriture représente moins de 10% du budget d'un ménage, mais la production des déchets alimentaires atteint entre 95 à 115 kilos annuels par personne. Dans les pays en voie de développement le budget du ménage consacré à l'alimentation se situe entre 40 et 70% mais ils ne produisent qu'entre 6-11 kg de déchets alimentaires (Afrique subsaharienne). (Denis Corboz 2015)

[1] In PARFITT ET AL. (2010): «Wholesome edible material intended for human consumption, arising at any point in the Food Supply Chain FSC that in instead discarded, lost, degraded or consumed by pests. »

## 2.1 Régions

### 2.1.1 Monde

Un tiers de la nourriture mondiale produite est jetée. Ce gaspillage alimentaire est très différent en fonction du pays dans lequel on se trouve. 40% des pertes sont dues, dans les pays en développement, aux conditions de récoltes, aux transports ou au stockage alors que dans les pays à fort pouvoir d'achat, 45% du gaspillage alimentaire est imputé directement aux consommateurs. (Stevan 2016)

D'après les estimations de l'Organisation des Nations Unies pour l'alimentation et l'agriculture (FAO), le tiers de toutes les denrées alimentaires produites dans le monde est jeté ou perdu pour l'alimentation. (France Nature Environnement 2013)

Outre les 45% de pertes imputées aux consommateurs, 30% sont dues à la transformation des aliments, 13 % à l'agriculture, 5% à la restauration et 5% au commerce de détail. (Stevan 2016)

### 2.1.2 Suisse

En Suisse 7 millions d'habitants, le gaspillage est estimé à 2 millions de tonnes par année. Ce qui représente pour chaque foyer suisse entre 500.- et 1000.- (Denis Corboz 2015)

Selon le groupe de projet « gaspillage alimentaire » de l'Office fédéral de l'agriculture OFAG, les acteurs de l'agriculture estiment que les pertes au niveau de la production sont minimes. Toutefois, en cas de récoltes particulièrement abondantes, ils peuvent se trouver devant une impasse et être contraints de jeter des fruits ou des légumes. Il est donc important d'assurer et d'informer rapidement les secteurs de distribution pour pouvoir vendre ces excédents au plus vite. Un autre problème qui favorise le gaspillage alimentaire au niveau de l'agriculture est le calibrage et l'esthétique des fruits et légumes. Les fruits et légumes esthétiquement moins appétissants ne sont pas acceptés par le commerce de gros et le commerce de détails.(OFAG 2016)

Dans le cadre du commerce de détails, l'une des raisons principales qui tend à favoriser le gaspillage, selon moi, est le conditionnement des aliments qui incite les consommateurs à acheter plus de nourriture qu'ils n'en ont vraiment besoin (le paquet de 3kg de pommes de terre au lieu d'acheter au détail). (OFAG 2016)

La planification des promotions se fait sur l'année et est souvent centralisée. De ce fait, il est très difficile de mettre en place une promotion à court terme qui permettent aux

acteurs en amont d'écouler les excédents en cas de récoltes abondantes (i.e. Tomates et abricots en 2018). (OFAG 2016)

### 2.1.3 Genève

221 producteurs agricoles sont sur le territoire Genevois et depuis une dizaine d'années le phénomène de la vente directe tend à s'accentuer (Genève Terroir 2018). Dans le cadre de mon projet, j'ai rencontré différents producteurs de vente directe à Genève lors de l'inauguration de la Maison de l'Alimentation du Territoire « Ma-Terre » à la ferme de Budé. Pour les producteurs qui fournissent principalement des fruits et légumes ainsi que des produits de garde tels que lentilles, huiles, graines et autres, les pertes sont identiques à la moyenne nationale. Tous les fruits et légumes sont distribués par le biais des paniers à leurs clients et les productions restent, malgré tout, de dimensions modestes.

Par contre, d'autres producteurs qui travaillent avec des denrées périssables peuvent avoir des pertes dues à un écoulement mal maitrisé ou à des produits que les consommateurs ne sont plus enclins à cuisiner (abats, .bas morceaux, pieds de porc…)

Au niveau des maraîchers genevois qui travaillent avec la grande distribution, ils sont plus impactés en cas de récoltes abondantes et également tributaires du critère de qualité des grandes surfaces qui refusent systématiquement un produit dès qu'il présente le moindre défaut.

## 2.2 Concurrents

Deux applications « Too Good to Go » et « Sav'Eat » commencent à se développer en Suisse mais elles sont principalement axées sur les invendus des restaurants qui désirent écouler leurs invendus**.** Mon application quant à elle, est plus destinée à permettre aux producteurs locaux d'écouler leurs marchandises et de favoriser le commerce de proximité entre producteurs et consommateurs. (Toninato 2018)

## 2.3 Différentes alternatives

D'autres alternatives existent à Genève pour lutter contre le gaspillage alimentaire. Les invendus des grandes enseignes de l'alimentation sont donnés à l'association Partage qui redistribuent ces denrées à des personnes dans le besoin (Toninato 2018). L'Union maraîchère genevoise, quant à elle, propose des ventes de légumes deuxième choix tous les jours de la semaine dans ses deux magasins à Carouge et Perly. Cela leur permet d'écouler les légumes et les fruits qui ont été refusé par la grande distribution. (Union Maraîchère de Genève 2018)

## 2.4 Accueil de l'application

Les différents producteurs approchés, dans le cadre de mon projet, ont été intéressés par celui-ci mais n'arrivaient pas vraiment à imaginer le fonctionnement de l'application sans l'avoir sous les yeux. Mon approche de rencontrer les producteurs lors d'un évènement (inauguration de la maison de l'Alimentation du Territoire « Ma-Terre ») ne s'est pas fait au moment le plus opportun, car ils n'avaient pas vraiment de temps à me consacrer du fait qu'ils étaient occupés avant tout à vendre et à faire connaître leurs produits. Cependant, ils m'ont tous demandé de les recontacter dès que celle-ci serait sur le point d'être opérationnelle.

### 2.4.1 Résumé des interviews

René Stalder du Domaine des Bougeries à Vandœuvres, produit des huiles (colza, caméline), des lentilles et des œufs. Il voit pour lui une utilité dans cette application pour faire connaître ses produits et surtout pour écouler ses œufs proches de la date d'expiration.

Grégoire Stoky de la ferme Moniati fait des paniers de légumes bio, vend des agneaux et des poulets et s'est lancé dans la production de tofu. Il est intéressé par une telle application pour la mise en avant de son tofu, pour pouvoir écouler également les abats et les parties moins nobles des agneaux et poulets.

La Vacherie du Carré est une fromagerie locale à Meinier et malgré sa petite taille, elle pense qu'une telle application lui permettrait d'éviter d'éventuelles pertes et de mieux maitriser le stock.

La Casa Mozzarella à Plan-les-Ouates est intéressée de faire de la promotion de ses produits mais n'a que peu de pertes alimentaires.

Myriam Dupraz du Jardin de Max à Vésenaz explique que pour eux les pertes sont extrêmement minimes, car ils font principalement des paniers et que leur clientèle est constante. Toutefois, ponctuellement, comme par exemple pendant les vacances de cet été, les clients étant absents et les récoltes surabondantes, une telle application leur aurait permis d'éviter des pertes.

Au regard des avis cités précédemment, cela permet de démontrer qu'il y a un intérêt certain dans la création d'une telle application.

# 3. Analyse des technologies

## 3.1 Framework de développement applicatif

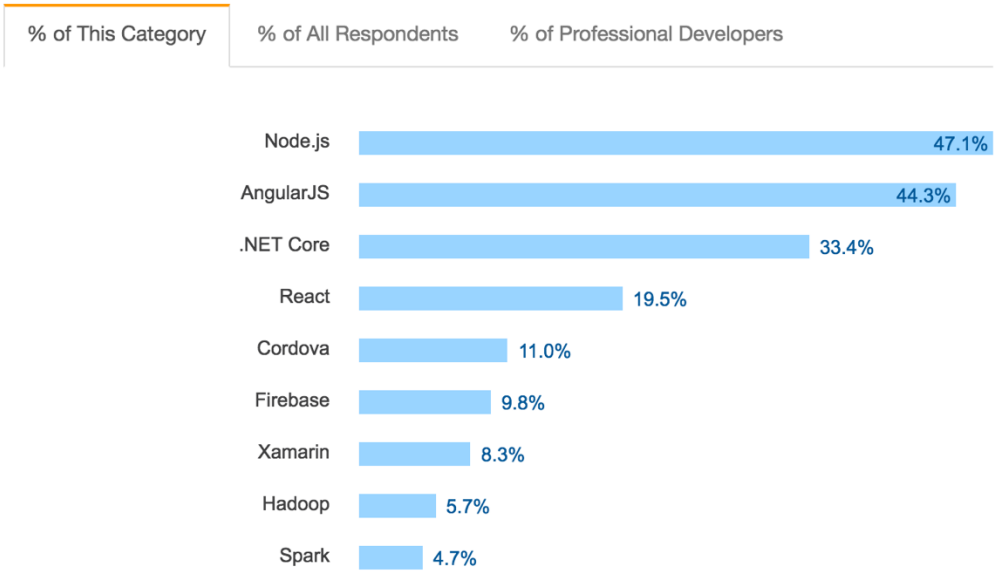
Pour la réalisation de ce projet, la première décision à prendre concernant le développement technique de celle-ci a été de savoir quel Framework utiliser. Un Framework désigne un ensemble de composants logiciels servant à la création de tout ou d'une partie d'un logiciel (Wikipedia 2018).

### 3.1.1 Framework hybride

Pourquoi choisir un développement hybride plutôt qu'un développement natif ? Car cela permet avec un seul code source de déployer le même projet sur iOS et Android. Il n'est donc pas nécessaire de connaître les langages Swift (iOS) et Java (Android) pour réaliser la même application sur ces deux plateformes. Une telle méthodologie permet donc un gain considérable en temps de développement si le but est de les déployer sur les deux marchés. Dans le cas d'une entreprise, il n'est donc pas nécessaire d'embaucher une équipe de développement par plateforme, ce qui représente au niveau financier un coût moindre.

Le choix du Framework s'est fait entre React Native et Ionic, étant les deux plus renommés pour le développement d'applications hybrides selon GitHub. Leur popularité peut aussi être constatée selon les tendances d'utilisation de ces dernières années. (Clockwise 2018)

Figure 1 : Tendance d'utilisation des Framework et des librairies




(Stack Overflow 2018)

### 3.1.2 Ionic 3

Ionic est un Framework de développement d'application hybride créée par Drifty.co en 2013. Ce dernier permet déployer des applications riches, élégantes et modernes sur la majorité des plateformes avec le même code source. Ionic implémente les technologies Web telles que le HTML, le CSS ainsi que le JavaScript et utilise les plateformes de Cordova ou PhoneGap permettant de délivrer une expérience utilisateur similaire à une application native. (Spec India 2018)

#### 3.1.2.1 Avantages

- Gratuit et Open Source
- Facile à prendre en main
- Documentation complète et bien amenée
- Permet de développer sur iOS, Android, Windows, Desktop, Web, and PWA
- Communauté solidaire
- Développement rapide

#### 3.1.2.2 Inconvénients

- Nécessite l'utilisation des plugins Cordova pour avoir accès au hardware de l'appareil
- Utilise le WebView pour simuler une expérience similaire à une application native

### 3.1.3 React Native

Développée par la comunauté de Facebook, React Native est un Framework de développement d'application Cross-Platform basée sur du JavaScript. React Native permet au développeur de créer des applications avec une expérience utilisateur et une performance aussi proche que possible à celle des applications natives. (Spec India 2018)

#### 3.1.3.1 Avantages

- Plateforme stable pour développer des applications d'envergure importante
- Performances comparables à une application native
- Utilise JavaScript, langage le plus utilisé au monde (Stack Overflow 2018)

#### 3.1.3.2 Inconvénients

- Nécessite plus de temps de développement, le code devant être adapté à chaque plateforme, mais rends l'expérience utilisateur similaire à celle d'une application native
- Nécessite une certaine expérience avec le développement d'applications natives

### 3.1.4 Comparaison

Pour mieux visualiser les différences entre Ionic et React Native j'ai listé chacune de leurs caractéristiques dans un tableau.

Tableau 1 : Comparaison des Framework

| | **Ionic 3** | **React Native** |
|---|---|---|
| **Concept** | « Write once, use anywhere » | « Learn once, write anywhere » |
| **Langage** | Technologies Web – HTML, CSS, JavaScript, Angular JS, TypeScript | React et JavaScript |
| **Type de Framework** | Hybride | Cross-Platform |
| **Développeurs** | Drifty.co | Communauté Facebook |
| **Populaire pour** | Avec un seul code source, peut être déployé sur iOS, Android, Windows, Web, PWA | Interfaces utilisateurs élégantes et similaires aux applications natifs |
| **Réutilisabilité du code** | 98% du code réutilisable | Adaptation du code nécessaire pour chaque plateforme |
| **Performances** | Plus lent que React Native à cause du WebView | Comparable à une application Native |
| **Débogage** | N'importe quel navigateur | Nécessite un appareil ou émulateur |
| **Prise en main** | Rapide grâce à la simplicité des technologies Web | Nécessite plus de temps |
| **Communauté** | Importante et stable | Importante et stable |
| **Documentation** | Complète et bien organisée | Pauvre et mal organisée |
| **Plateformes** | Android, iOS, UWP, PWA | Android, iOS, UWP |
| **Type d'applications** | Petites applications | Tout type |

(Spec India 2018)

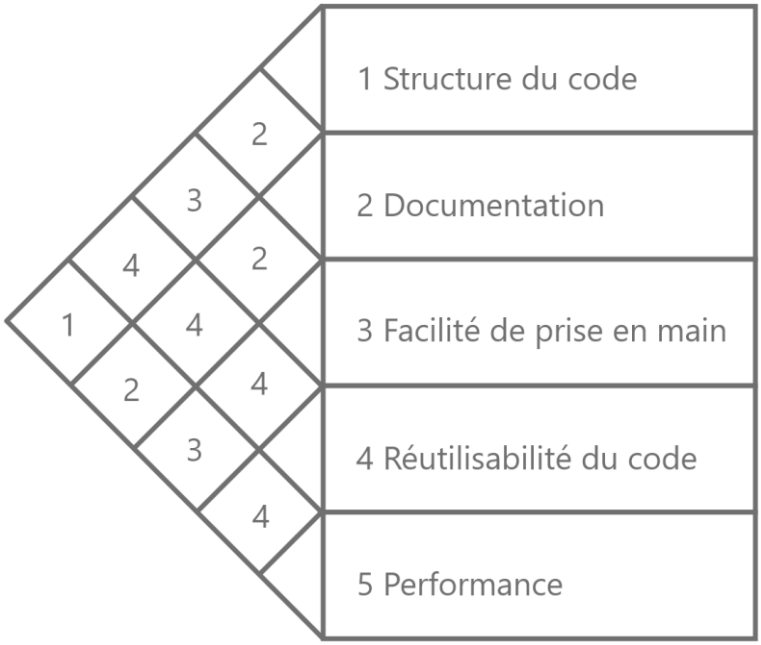
### 3.1.5 Choix du Framework

Afin de m'assurer que le Framework choisi corresponde bien à mes besoins je me sers d'une matrice de préférence.

Voici les critères qui sont pour moi les plus importants :

1) Structure du code
2) Documentation
3) Facilité de prise en main
4) Réutilisabilité du code
5) Performances

Figure 2 : Matrice de préférence des critères de choix du Framework




(Alex Perritaz 2018)

Tableau 2 : Pondération des critères pour le choix du Framework

| Numéro | Critère | Pondération |
|---|---|---|
| 1 | Structure du code | 1 |
| 2 | Documentation | 3 |
| 3 | Facilité de prise en main | 2 |
| 4 | Réutilisabilité du code | 4 |
| 5 | Performance | 0 |

Pour le développement d'une application j'estime qu'il est nécessaire de pouvoir séparer les différentes méthodes et en faire appel lorsque celle-ci est nécessaire au lieu de réécrire le même morceau de code à chaque fois. Cela permet d'éviter de perdre du temps de refaire quelque chose que l'on a déjà fait.

Il est aussi important d'avoir une bonne documentation bien structurée, on ne va pas perdre du temps à essayer de comprendre par nous-même ce qu'une instruction est censée faire, surtout lorsque l'on est en train développer une application et qu'on a que 8 semaines pour le faire. Une bonne documentation permet de savoir comment utiliser une fonction, ce qu'elle permet de faire et quelles sont ses limites.

Il m'est donc nécessaire de choisir un Framework facile à prendre en main pour pouvoir réaliser ce que j'ai prévu. Je me dois d'éviter de perdre trop de temps à essayer de maitriser un langage qui m'est complètement inconnu et de devoir passer plusieurs heures dans la documentation, qui peut aussi s'avérer incomplète, alors que je pourrais déjà être en train d'attaquer une des fonctionnalités principales de l'application.

Tableau 3 : Analyse multicritère du choix du Framework

| Critères | Pondération | Ionic | | React Native | |
|---|---|---|---|---|---|
| | | Points | Pondérés | Points | Pondérés |
| Structure | 1 | 8 | 8 | 4 | 4 |
| Documentation | 3 | 9 | 27 | 5 | 15 |
| Prise en main | 2 | 7 | 14 | 5 | 10 |
| Réutilisabilité | 4 | 8 | 32 | 7 | 28 |
| Performance | 0 | x | x | x | x |
| Total | | 32 | 81 | 21 | 57 |

Ionic est donc le Framework qui répond le mieux à mes attentes pour développer mon application. Ce Framework permet, grâce à la réutilisabilité de son code, par sa documentation riche et complète, par sa prise en main très rapide ainsi qu'à sa structure séparant le visuel de la logique permettant de créer des applications à partir de rien et d'en sortir un beau produit en un très peu de temps. De plus, avoir un code bien structuré et séparé en plusieurs méthodes réutilisables, rend l'application plus maintenable et permet de continuer à développer ce projet par la suite sans trop s'y perdre.

Mais ce n'est pas pour autant que je ne souhaite pas essayer d'utiliser React Native, au contraire, j'ai envie par la suite d'y investir du temps à apprendre comme il faut l'utilisation de ce dernier et non devoir me dépêcher pour développer des application Cross-Platform.

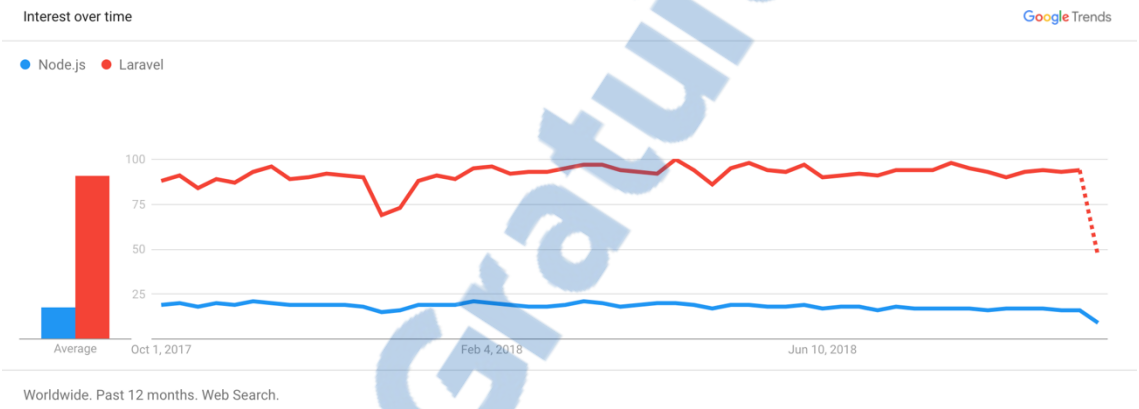
En ce qui concerne les performances, tous deux sont plus que suffisant pour l'application que je souhaite réaliser, je ne vois donc pas l'importance de ce critère là pour ce projet.

Le Framework de développement choisi, il faut maintenant passer au Web Service.

## 3.2 Web Service

Un web service est une méthode permettant à des applications de communiquer entre eux par le biais d'internet sans tenir compte de leurs plateformes ni du langage avec lequel elles ont été codées. Il existe 2 types de web service, le premier s'agit du SOAP (Simple Object Access Protocol) et le second du REST (Representational State Transfer). C'est à ce dernier que nous allons nous intéresser dans les points suivants. (Swati Dhingra 2018)

Tableau 4 : Tendances d'utilisation des deux Web Service des derniers 12 mois




(Stackshare 2018)

### 3.2.1 Laravel

Lorsqu'il s'agit de se décider entre Laravel et Node.js il suffit de regarder le graphique ci-dessus pour comprendre que la majorité des gens utilisent maintenant Laravel et ce pour une bonne raison.

Je vais simplement expliquer pourquoi Laravel est une meilleure alternative que Node.js pour moi :

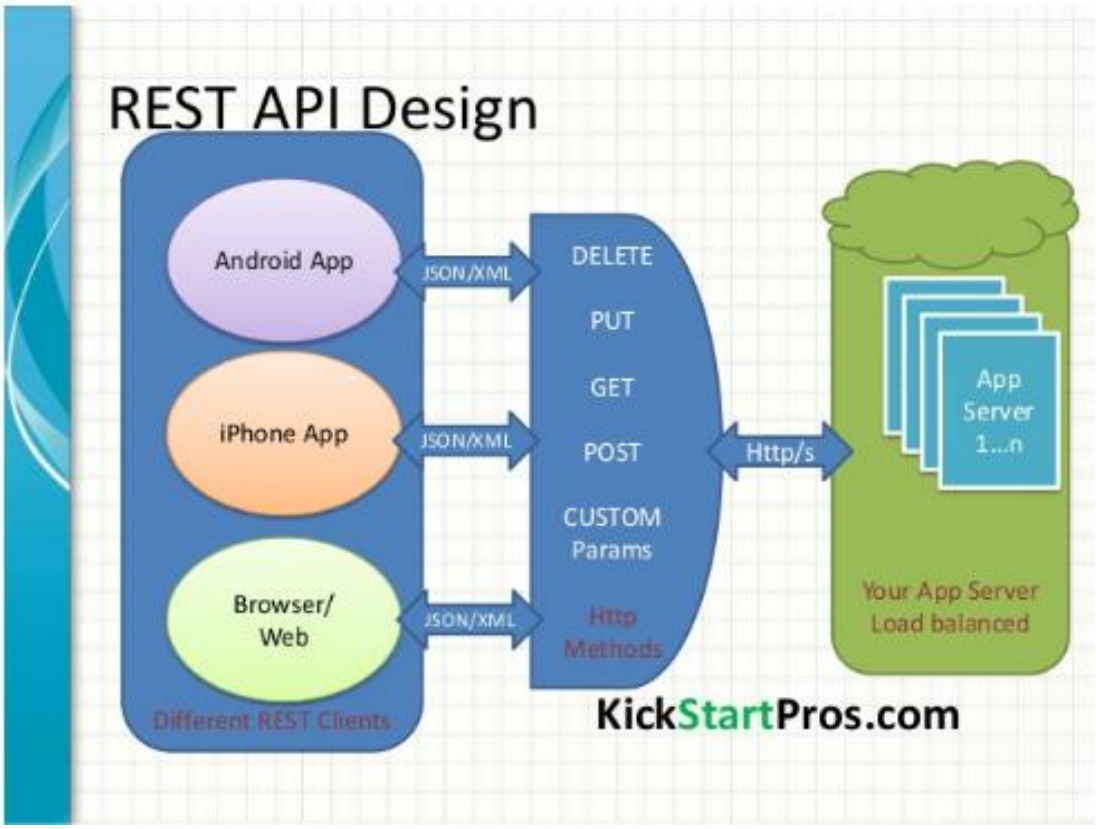
- Il permet un développement et une mise en place rapide du web service
- Il adopte le pattern MVC séparant le modèle, la vue et le controller permettant d'avoir un code bien structuré et plus propre selon moi.
- Possibilité d'utiliser Composer pour la gestion des dépendances du projet
- Laravel dispose d'une communauté importante et solidaire permettant d'aider rapidement à la résolution des problèmes rencontré.
- Laravel est **le** Framework à utiliser si on aime le PHP

(Stackshare 2018)

### 3.2.2 API Rest

Une API REST se doit avant tout d'être sans état, c'est-à-dire que la requête d'un client ne doit pas dépendre d'un quelconque contexte du serveur. Il doit contenir toutes les informations nécessaires pour l'intégralité de son traitement, ce qui permet au serveur de traiter indifféremment les différentes requêtes reçues.

Figure 3 : Design d'API REST




(Hugo HOUYEZ 2017)

Indépendamment des plateformes et du langage utilisé, l'API REST permet le dialogue entre plusieurs applications à distance par le biais d'internet. Cette technologie se base sur les protocoles http pour pouvoir communiquer.

Une fois que l'on sait avec quoi on va réaliser notre application, il s'agit de savoir ce qu'il va faire.

# 4. Fonctionnalités de haut niveau

Les fonctionnalités principales de l'application sont réalisées si possible avec les plugins créés par les développeurs du Framework pour assurer une compatibilité maximale tout en garantissant une bonne maintenabilité. Les plugins tels que Geolocation et Geocoder qui seront utilisés font donc partis de ceux développés par Ionic.

## 4.1 Autocomplétion

Le service d'auto-complétion de l'API de Google Places permet de récupérer des prédictions d'adresses basée sur la saisie de l'utilisateur. Chaque prédiction est une réponse d'une requête HTTP retournant soit une sortie JSON (JavaScript Object Notation) soit une sortie XML. (Google 2018a)

Voici un exemple de lien généré par une de ces requêtes HTTP :

*https://maps.googleapis.com/maps/api/place/autocomplete/output?parameters*

Le format JSON a été utilisé dans ce cas-là, étant plus court, plus facile à lire et peut utiliser des tableaux pour récupérer les données. (w3schools 2016)

Il est aussi possible de restreindre les résultats à des établissements, adresses ou à un Geocode (objet contenant les informations concernant la latitude et la longitude d'un emplacement) ainsi que de les limiter à une région spécifique. (Google 2018a)

Le fait d'utiliser cette méthode permet de s'assurer que les données reçues lors de la sélection d'une adresse soient traitées de manière similaire à chaque fois, évitant ainsi des erreurs de formatage des adresses.

Les objets récupérés par la requête sont ensuite utilisés pour afficher les prédictions de l'adresse saisie par l'utilisateur dans le composant *Searchbar* (section 6.1.4.4.1). Elle affiche la liste des adresses possibles se rapprochant de la saisie de l'utilisateur et se précisant plus il aura saisi de caractères.

## 4.2 Géolocalisation

Les fonctionnalités de géolocalisation permettent d'obtenir des informations concernant des emplacements géographiques des adresses, des utilisateurs ainsi que des distances séparant deux coordonnées. Chacune d'entre elles sera développée dans les points suivants.

### 4.2.1 Geocoder

Le plugin Cordova Geocoder permet d'obtenir une adresse postale à partir de repères géographique composé d'une latitude, d'une longitude et vice-versa. On peut également y ajouter en paramètre des options qui permettent d'affiner les résultats de la recherche. Dans le cadre de cette application elle est utilisée pour récupérer les coordonnées de l'adresse saisie grâce à la fonctionnalité d'auto-complétion précédemment traitée. (Ionic Framework 2018a)

### 4.2.2 Geolocation

Le plugin Cordova Geolocation permet quant à lui d'obtenir la latitude et longitude grâce aux données GPS et aux données mobile de l'appareil. Elle est utilisée ici pour obtenir la position actuelle de l'utilisateur afin de pouvoir obtenir et afficher les offres les plus proches de lui dans un certain périmètre. (Ionic Framework 2018b)

### 4.2.3 Formule de Haversine

En ce qui concerne le calcul de distances entre deux points géographiques, il a été nécessaire de passer du temps dessus pour identifier la méthode qui serait la plus adéquate à l'application. Il s'agissait de traiter le plus de points possibles dans le laps de temps le plus court sur toutes les entrées de la base de données.

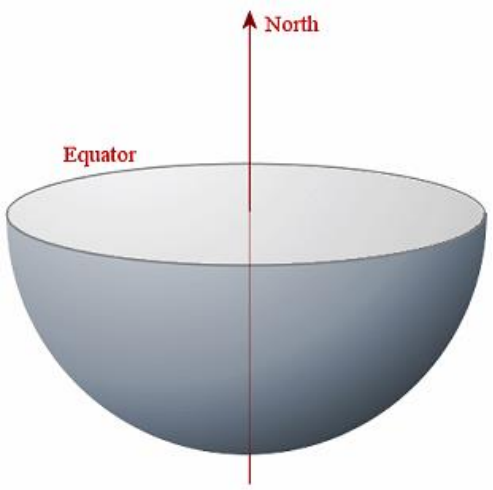
Dans un premier temps lorsqu'il s'agit de calculer la distance entre deux points, la première idée à venir à l'esprit est le théorème de Pythagore mais il s'agit là d'une erreur car le théorème ne s'applique qu'aux surfaces en deux dimensions alors que les coordonnées géographiques en latitude et longitude sont des points de repères sur une sphère (PolyGeo 2016) .

Par exemple si nous prenons pour emplacements sur la Terre, Tokyo et Genève avec le théorème de Pythagore, nous aurions une distance en ligne droite qui passerais au travers de la planète. La distance idéale serait celle qui suit la courbe terrestre comme celle empruntée par les avions. (Murray Bourne 2018)

Pour cela il faut utiliser la formule de Haversine qui utilise la latitude et la longitude pour représenter les coordonnées géographiques afin de calculer la distance entre deux emplacements.

Pour comprendre comment cette formule fonctionne il faut tout d'abord comprendre le concept de latitude et de longitude. La Terre sera représentée en tant que sphère avec le nord au sommet ainsi que l'équateur à l'angle de 0°N.

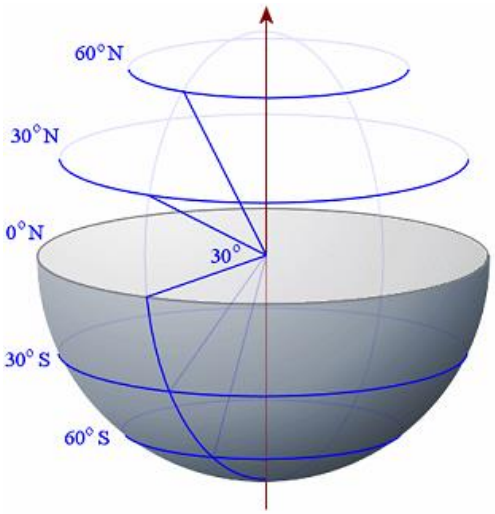
Figure 4 : Représentation de la Terre




(Murray Bourne 2018)

La latitude est indiquée en degrés et représente un point situé au nord ou au sud de la référence qu'est l'équateur. Les angles vont de -90° (correspondant à 90°S) en passant par l'équateur qui se situe à l'angle 0° et allant jusqu'à 90° (correspondant à 90°N) qui se situe lui dans la partie supérieure du globe. Lorsque tous les points d'une même latitude sont reliés entre eux, cela forme un cercle parallèle à celui de l'équateur.

Figure 5 : Représentation de la latitude

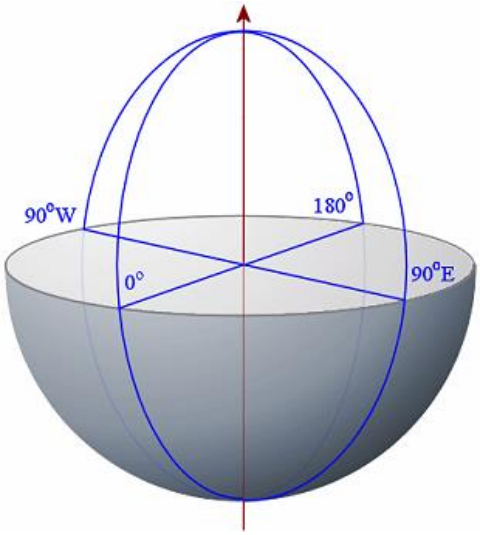



(Murray Bourne 2018)

A noter que 1° de latitude correspond à peu près à 69 miles nautique ce qui équivaut à approximativement 111 kilomètres (longitudestore 2018). Elle peut varier très légèrement en se rapprochant des pôles car la Terre n'est pas parfaitement sphérique mais reste suffisamment précise dans le cadre de l'application. (@natgeofrance 2018)

La longitude contrairement à la latitude ne se base pas sur le rapport aux pôles et à l'équateur mais sur le Méridien de Greenwich situé en Angleterre. Elle est cependant aussi mesurée en degrés mais par rapport au Premier Méridien allant de -180° (-180W) en passant par le méridien jusqu'à 180°(180°E). (Murray Bourne 2018)

Figure 6 : Représentation de la longitude




(Murray Bourne 2018)

La distance pour 1° de longitude n'est pas fixe, elle dépend fortement de la latitude. Plus les points se trouvent proche d'un pôle plus petit sera la distance entre ces deux sur la même latitude. Pour cela pour 1° il faut calculer : cos(latitude) * 69 pour obtenir le résultat en miles ou bien cos(latitude) * 111 pour la distance en kilomètres. (Alexander Rubin 2018)

Une fois ces deux notions comprises il est possible de passer à la suite et effectuer le calcul de distance entre deux points en utilisant la formule de Haversine.

Pour deux points sur une sphère :

- Latitude 1 : **φ1**
- Latitude 2 : **φ2**
- Longitude 1 : **λ1**
- Longitude 2 : **λ2**
- Rayon de la sphère : **R**
- Différence des latitudes : **Δφ**
- Différence des longitudes : **Δλ**
- Distance : **d**

Avec pour formule :

Figure 7 : Formule de Haversine

$$\operatorname{haversin}\left(\frac{d}{R}\right) = \operatorname{haversin}(\Delta\phi) + \cos(\phi_1)\cos(\phi_2)\operatorname{haversin}(\Delta\lambda)$$

(Alexander Rubin 2018)

Pour Haversine de l'angle θ en question, la formule est la suivante :

**Haversine(θ)** = $\sin^2(\theta / 2)$

**versin(θ)** = $1 - \cos(\theta) = 2\sin^2(\theta / 2)$

La formule devient donc :

Haversine $(d / R) = \sin^2(\theta / 2) + \cos(\mathbf{\varphi 1}) * \cos(\mathbf{\varphi 1}) * \sin^2(\theta / 2)$

Pour comprendre plus simplement comment le calcul de distance fonctionne, il est plus simple de la décomposer et de l'analyser en plusieurs parties.

Avec comme paramètres les valeurs suivantes :

**R** = 6731 → Rayon de la Terre en Km

**Δlat** = **φ1**– **φ2** → La différence des latitudes

**Δlong** = **λ2** – **λ1** → La différence des longitudes

Remplacer l'équation de la formule à l'aide des valeurs :

**a** = $\sin^2$(**Δlat**/2) + cos(**φ1**) * cos(**φ2**) * $\sin^2$(**Δlong**/2)

**c** = 2 * atan2(√**a**, √(1-**a**))

Finalement pour obtenir la distance on multiple le rayon de la Terre par l'équation précédente :

Distance = **R** * **c**

La formule utilisée dans la requête finale pour obtenir la distance est donc la suivante :

```
6371 * 2 * ASIN( SQRT( POWER( SIN((latitude2 - latitude1) * pi()/180 / 2), 2)
+COS (latitude2 * pi() / 180) * COS(latitude1* pi() / 180)
* POWER( SIN((longitude2 – longitude1) * pi() / 180 / 2), 2) ))
```

A noter qu'il faut multiplier **Δlat** et **Δlong** par π et diviser par 180 pour obtenir les angles en radians pour pouvoir effectuer le calcul de l'arc sinus (ASIN).

Cette formule est donc la requête de base pour obtenir les distances avec les entrées de la base de données.

On peut améliorer les performances de cette formule en limitant les recherches dans la base de données en réduisant ceux-ci à ceux qui se trouvent dans le rayon de recherche défini par l'utilisateur. On effectue alors une recherche entre les latitudes et longitudes se trouvant à X distance de la position définie par l'utilisateur, évitant ainsi un calcul des distances de chaque entrée dans la base de données. Cela évite donc d'utiliser la formule de calcul de distance qui est plus gourmande en ressources qu'une simple comparaison.

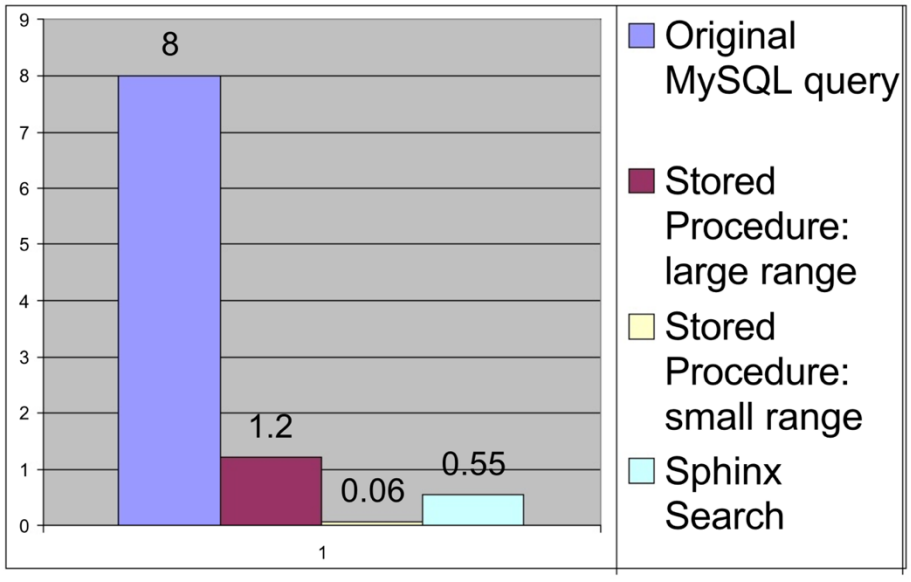
Exemple avec une expression SQL :

```
WHERE λ2
BETWEEN λ1 – d / abs(cos(radians(ϕ1)) * 111)
AND λ1 + d / abs(cos(radians(ϕ1)) * 111)
AND ϕ2
BETWEEN ϕ1 -(d / 111)
AND ϕ1 +(d / 111))
```

**Rappel** : 1° de latitude correspond à 111km

L'efficacité de ces lignes supplémentaires peut se remarquer sur le graphique ci-dessous affichant les performances des différentes variantes des requêtes SQL. On peut remarquer que cette dernière requête ne prend qu'entre 1.2 et 0.06 secondes, ce qui représente une diminution de 6 à 133 fois le temps de la requête de base (variation en fonction de la distance de recherche).

Figure 8 : Performances des différentes requêtes SQL de calcul de distance




(Alexander Rubin 2018)

La formule du calcul de distance de Haversine est bien plus que suffisante pour l'application actuelle en raison du peu de données à calculer. Il est toujours possible d'améliorer le temps de calcul en utilisant la méthode de GeoHash (section 9.1.1) si nécessaire, mais la différence ne se verrait pas avec si peu de données.

## 4.3 Google Maps

L'API de Google Maps a été inclue dans l'application dans le but d'afficher une carte à l'emplacement actuel de l'utilisateur grâce aux services de Geolocation (section 4.2.2) ainsi que les vendeurs qui se situent à proximité (Google 2018b)

Cependant depuis le 11 juin 2018, les services de maps de Google sont devenus payants et il est par conséquent nécessaire d'entrer ses informations bancaires afin de pouvoir utiliser cette fonctionnalité. (Marco 2018)

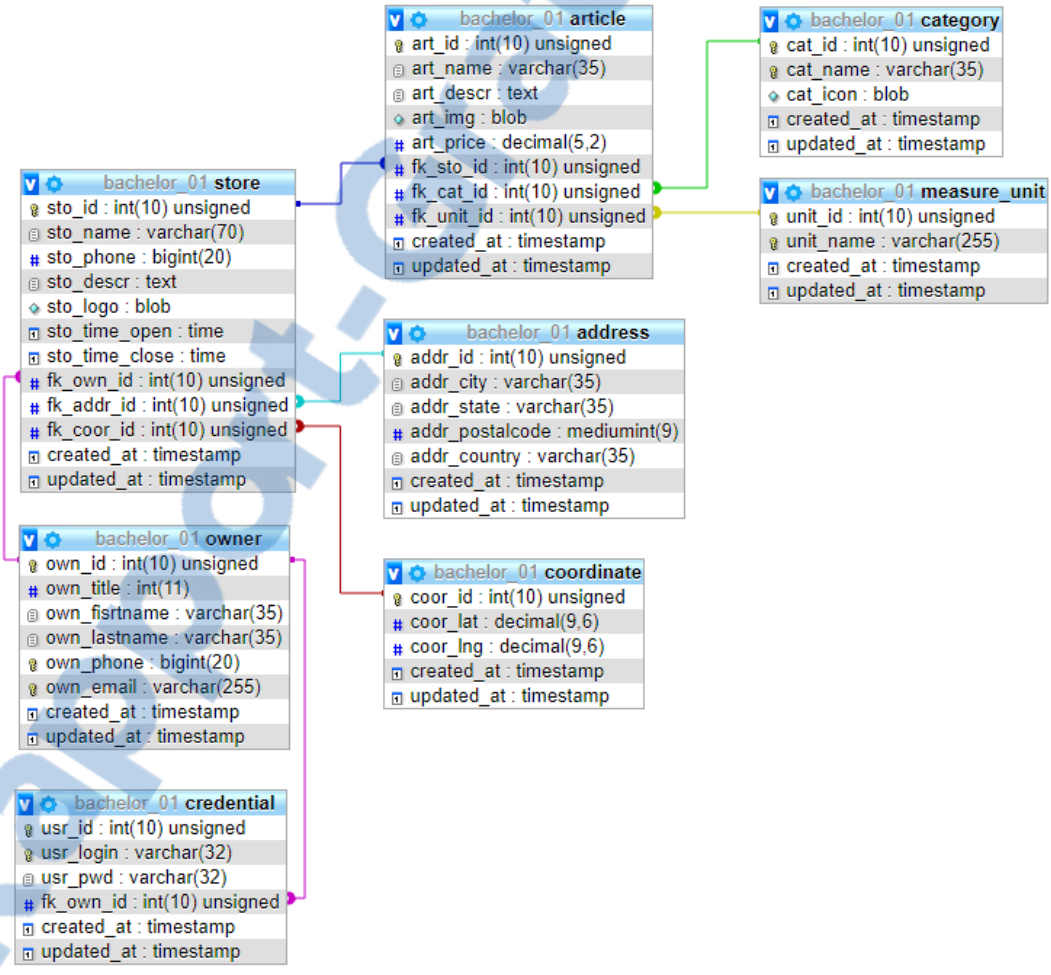
# 5. Modélisation des données

Maintenant que l'on sait aussi quelles sont les fonctionnalités principales de notre application, il faut savoir comment entreposer toutes ces données et comment les utiliser. Pour ce faire, on va les représenter avec un modèle UML.

## 5.1 Modèle UML

Le modèle de données suivant représente la structure de la base de données que l'application utilise. Il est important de comprendre comment les différentes tables sont liées entre elles et la manière dont elles interagissent les unes avec les autres pour pouvoir récupérer les données souhaitées.

Figure 9 : Modèle UML de la base de données




(Alex Perritaz 2018)

## 5.2 Différentes tables

Les différentes tables ci-dessus ont été créés pour contenir les données des commerces et des articles qu'ils proposent. Chacun des commerces est géré par un propriétaire, mais ceux-ci peuvent en gérer plusieurs. Les informations concernant les différentes tables sont décrites un peu plus en détail ci-dessous.

### 5.2.1 Store

Contient les données concernant le commerce :

Tableau 5 : Base de données - Table Store

| | | | |
|---|---|---|---|
| sto_id | ID du commerce | sto_logo | Logo |
| sto_name | Nom | sto_time_open | Heure d'ouverture |
| sto_phone | Numéro de téléphone | sto_time_close | Heure de fermeture |
| sto_descr | Description | | |

### 5.2.2 Address

Contient les données concernant l'adresse du commerce lié :

Tableau 6 : Base de données - Table Address

| | | | |
|---|---|---|---|
| addr_id | ID de l'adresse | addr_postalcode | Numéro postal |
| addr_city | Nom de la ville | addr_country | Nom du pays |
| addr_state | Nom du canton | | |

### 5.2.3 Coordinates

Contient les coordonnées du commerce lié :

Tableau 7 : Base de données - Table Coordinates

| | | | |
|---|---|---|---|
| coor_id | ID coordonnée | coor_lng | Longitude |
| coor_lat | Latitude | | |

Les coordonnées sont séparées de l'adresse afin d'éviter de devoir y accéder pour les obtenir et les récupérer directement.

### 5.2.4 Owner

Contient les informations concernant le propriétaire d'un ou plusieurs commerces :

Tableau 8 : Base de données - Table Owner

| | | | |
|---|---|---|---|
| own_id | ID du propriétaire | own_lastname | Nom de famille |
| own_title | Genre | own_phone | Téléphone privé |
| own_firstname | Prénom | own_email | Adresse e-mail |

### 5.2.5 Credential

Contient les informations de connexion d'un propriétaire de commerce(s) :

Tableau 9 : Base de données - Table Credential

| | | | |
|---|---|---|---|
| usr_id | ID utilisateur | usr_pwd | Mot de passe |
| usr_login | Identifiant | | |

### 5.2.6 Article

Contient les informations d'un article d'un commerce :

Tableau 10 : Base de données - Table Article

| | | | |
|---|---|---|---|
| art_id | ID de l'article | art_img | Image |
| art_name | Nom | art_price | Prix de l'article |
| art_descr | Description | | |

### 5.2.7 Category

Contient les informations d'une catégorie d'un article :

Tableau 11 : Base de données - Table Category

| | | | |
|---|---|---|---|
| cat_id | ID catégorie | cat_icon | Icône |
| cat_name | Nom | | |

### 5.2.8 Measure_Unit

Contient les informations de l'unité de mesure d'un article :

Tableau 12 : Base de données - Table Measure_Unit

| | | | |
|---|---|---|---|
| unit_id | ID unité de mesure | unit_name | Nom |

# 6. Architecture de l'application

Les données maintenant structurées, il faut en faire de même pour l'architecture de notre application.

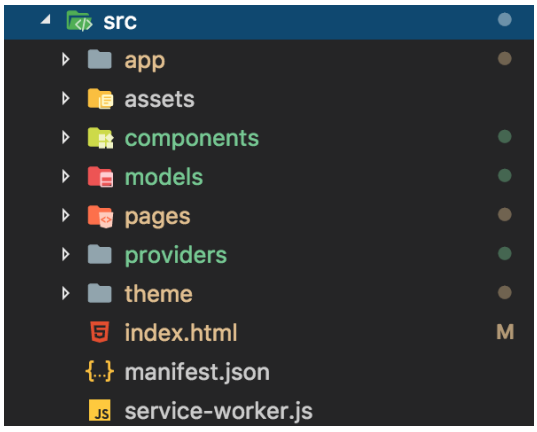
L'architecture de l'application peut être divisée en deux grandes parties distinctes. Le front end qui s'occupe principalement de la couche de l'application qui concerne les interfaces qui interagissent avec l'utilisateur. Elle traite principalement les vues et l'affichage des informations. La partie back end est celle qui se trouve du côté serveur et qui traite toutes les interactions avec la base de données ainsi que le traitement des données brutes sans éléments d'affichage pour l'utilisateur final.

## 6.1 Front end

### 6.1.1 Structure Ionic

Dans le projet créé sur Ionic, les dossiers contenant les parties les plus importants du front end permettant l'affichage des informations sont au nombre de trois. On y trouve les trois dossiers suivants : *app*, *pages* et *components.*

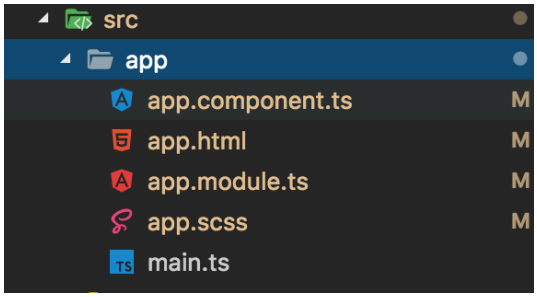
Figure 10 : Structure du projet Ionic




(Alex Perritaz 2018)

Le dossier app est le premier auquel l'application accède lors de son lancement. Il contient tous les fichiers nécessaires à l'amorçage ainsi que la structure principale de l'application. (Ionic Framework 2017)

Figure 11 : Structure du dossier app



(Alex Perritaz 2018)

Le fichier *app.component.ts* définit la structure principale du projet et ainsi que la redirection sur la page principale de l'application. Lorsque la plateforme ainsi que les plugins sont prêts (Cordova et d'autres plugins natifs), l'application est notifiée et l'utilisation de ces derniers en est possible. Ceci permettant donc l'utilisation de fonctionnalités supplémentaires nécessaires au projet. (Ionic Framework 2017)

Le fichier *app.html*, quant à lui, définit la navigation ainsi que la racine du projet. Il peut aussi afficher du texte, un logo ou toutes autres informations de type HTML contenu dans le fichier lors du chargement des ressources et composants de l'application. Dans le cas de ce projet, il affiche uniquement le logo ainsi qu'une animation de chargement. (Ionic Framework 2017)

Le fichier *app.module.ts* sert de point d'entrée et permet l'importation du module de *Angular2 NgModule.* Ce dernier est basé sur TypeScript, langage avec lequel tous les fichiers ayant pour extension *.ts* sont codés. C'est aussi dans ce fichier que sont faites toutes les déclarations de pages (sauf dans le cas du lazy loading, section 6.1.3 ) components et providers. Cela permet d'assembler les pages du projet, afin qu'elles soient accessibles et liées entre elles. (Ionic Framework 2017)

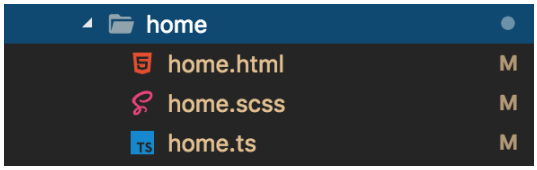
La feuille de style permet de styliser les éléments HTML. *App.scss* permet dans le cas de ce projet de gérer les propriétés du logo ainsi que l'animation des éléments de chargement. Il est possible de modifier le thème global de l'application en ajoutant des règles spécifiques dans le fichier */src/theme/variables.scss* mais ne se fait en aucun cas ici. (Ionic Framework 2017)

Le fichier *main.ts* du répertoire *app* est un fichier auto-généré par Ionic qui s'occupe simplement de l'amorçage de l'application.(Ionic Framework 2017)

### 6.1.2 Pages

Une page créée avec le Framework Ionic est représentée par un dossier contenant les informations relatives à celle-ci. Par défaut une page Home est générée en même temps que la création du projet et en voici sa structure :

Figure 12 : Structure du dossier home




(Alex Perritaz 2018)

#### 6.1.2.1 Création

Pour créer une nouvelle page comme pour la plupart des éléments dans Ionic, il suffit de passer par le CLI (Command Line Interface) du Framework et de saisir la commande suivante : *$ ionic generate page [<nom>] (Ionic Framework 2018c)*

#### 6.1.2.2 Structure

Lors de la génération d'une nouvelle page, un dossier du même nom est créé contenant les 4 fichiers suivants :

- Un fichier *HTML* servant à structurer le composant
- Un fichier *module.ts* servant à importer et déclarer les éléments utilisés par le composant
- Un fichier *scss* servant à styliser la structure du fichier *HTML*
- Un fichier *.ts* servant à définir les attributs, propriétés et méthodes du composant

Figure 13 : Structure d'un exemple de page


(Alex Perritaz 2018)

#### 6.1.2.3 Implémentation

Une fois la seconde page ajoutée au projet, il faut l'importer et l'ajouter aux déclarations et aux entryComponent dans le fichier *app.module.ts*.

Figure 14 : Contenu app.module.ts - Eager loading

```
import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';
import { SecondPage } from '../pages/second/second';

@NgModule({
  declarations: [
    MyApp,
    HomePage,
    SecondPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    SecondPage
  ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
})
```

(Alex Perritaz 2018)

Une fois le fichier *app.module.ts* complété, il faut aussi l'importer dans le composant dans lequel on veut en faire l'appel, ainsi que dans le code.

Figure 15 : Exemple de composant - Eager loading

```
import { SecondPage } from "../second/second";

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(public navCtrl: NavController) {
    this.navCtrl.push(SecondPage);
  }
}
```

(Alex Perritaz 2018)

### 6.1.3 Lazy Loading

Afin d'éviter de devoir importer les pages dans le *app.module.ts*, les ajouter aux *declarations*, aux *entryComponents* et ensuite en faire un autre import à chaque fois que l'on veut faire appel à la page en question. Il est préférable d'utiliser le concept du Lazy Loading.

Contrairement au Eager Loading (la méthode par défaut), cette méthode ne nécessite pas l'import ou l'ajout de quoi que ce soit dans le fichier *app.module.ts*. Il faut donc retirer chacun des éléments déjà ajoutés au module.

```
import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';
import { SecondPage } from '../pages/second/second';

@NgModule({
  declarations: [
    MyApp,
    HomePage,
    SecondPage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage,
    SecondPage
  ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
})
```

```
import { MyApp } from './app.component';
import { HomePage } from '../pages/home/home';

@NgModule({
  declarations: [
    MyApp,
    HomePage
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    HomePage
  ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}
  ]
})
```

(Alex Perritaz 2018)

Aucun import n'a besoin d'être fait sur aucune des classes ou composants nécessitant l'appel à une page. Un simple appel avec le nom de l'élément entre apostrophes suffit.

```
import { SecondPage } from "../second/second";

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(public navCtrl: NavController) {
    this.navCtrl.push(SecondPage);
  }
}
```

```
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(public navCtrl: NavController) {
    this.navCtrl.push('SecondPage');
  }
}
```

(Alex Perritaz 2018)

Il faut cependant s'assurer que dans chaque page le fichier *[nom].module.ts* soit présent même si celui-ci est généré par défaut lors de la création d'une nouvelle page (via le CLI ionic comme mentionné plus haut). Par exemple pour la page home qui est créé par défaut il faudra l'ajouter à la main.

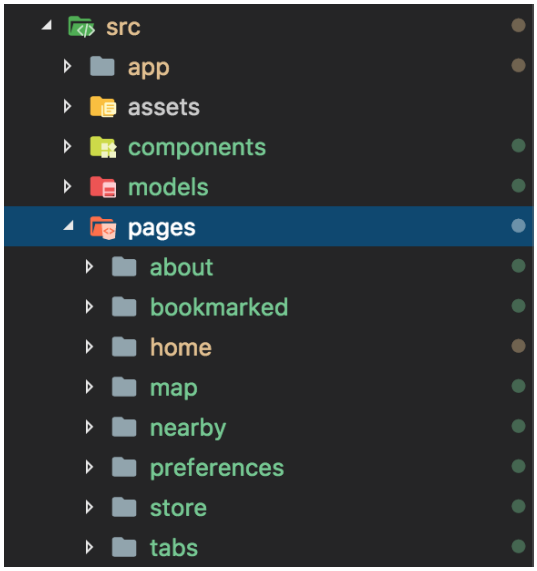
Le lazy loading a été beaucoup utilisé lors du développement du projet. Mais qu'est-ce que c'est, à quoi sert-il et pourquoi l'avoir utilisé ?

Tout d'abord le Lazy Loading est un concept implémenté dans la troisième version de Ionic et n'est encore qu'en version bêta. Il s'agit d'un design pattern qui consiste à n'appeler les éléments et différentes pages de l'application uniquement lorsque l'on en a besoin et pas avant. Son grand avantage est avant tout de simplifier l'implémentation. (Mike 2017)

C'est une bonne habitude de l'utiliser, mais il n'est pas nécessairement plus performant que son opposé le Eager Loading. Cela dépend simplement de l'application créée, il faut en analyser les performances pour savoir lequel correspond le mieux aux besoins. (Michael Maddox 2010)

Chacune des pages du projet a été créée en utilisant le concept de Lazy Loading et leur utilité sera présenté plus loin dans le document (section 0)

Figure 16 : Dossier des pages du projet




(Alex Perritaz 2018)

### 6.1.4 Components

Tout d'abord, il faut bien comprendre la différence entre un composant (component) et une directive dans Ionic. Bien que leurs fondements soient identiques, il existe cependant une différence subtile à bien assimiler. Tous deux permettent de créer un composant réutilisable dans une page HTML. Néanmoins, une directive est utile pour modifier un élément déjà existant pour l'adapter aux besoins. Tandis qu'un composant, bien que similaire à une directive, aura son propre design composé de plusieurs éléments. (Morony 2015)

#### 6.1.4.1 Création

Pour créer une nouvelle page comme pour la plupart des éléments dans Ionic, il suffit de passer par le CLI (Command Line Interface) du Framework et de saisir la commande suivante : *$ ionic generate component [<nom>] (Ionic Framework 2018c)*

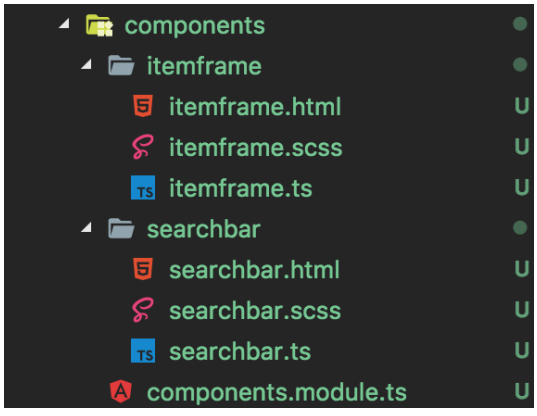
#### 6.1.4.2 Structure

La structure d'un composant est similaire à celle d'une page Ionic normale à une exception près : le fichier *components.module.ts* ne se trouve pas dans chaque composant mais est un module partagé à la racine du dossier « components ».

Chaque composant contient donc

- Un fichier *HTML* servant à structurer le composant
- Un fichier *scss* servant à styliser la structure du fichier *HTML*
- Un fichier *.ts* servant à définir les attributs, propriétés et méthodes du composant

Figure 17 : Structure Component




(Alex Perritaz 2018)

#### 6.1.4.3 Implémentation

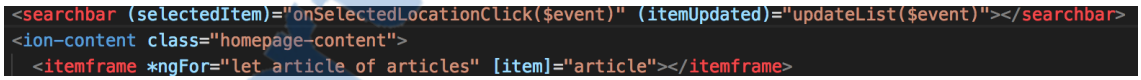
Dans le fichier partagé *components.module.ts*, il faut importer, déclarer et ajouter les composants aux exports pour en permettre la déclaration dans un tag html d'un autre composant.

Figure 18 : Déclaration de composants partagé


(Alex Perritaz 2018)

Ci-dessus nous pouvons voir les différentes déclarations et exports, tandis que ci-dessous nous avons l'implémentation de l'élément dans une structure HTML.

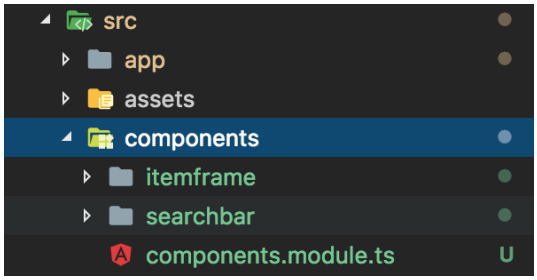
Figure 19 : Utilisation des composants créés




(Alex Perritaz 2018)

La création de ces composants permet une utilisation simplifiée de ceux-ci. En outre, il permet d'éviter le code redondant et en améliore la structure.

Figure 20 : Dossier des composants créés




(Alex Perritaz 2018)

Dans le cas de cette application, comme pour la plupart des petites applications il sera préférable de n'avoir qu'un seul module. Il est aussi possible d'avoir un module pour chaque composant mais il n'est pas recommandé de le faire de cette manière pour simple raison de maintenabilité du code. Le plus efficace, maintenable et facile à gérer

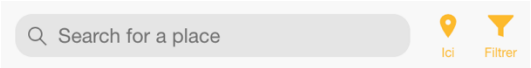
pour des projets comportant beaucoup de composants est de créer plusieurs modules partagés les regroupant par catégories ou fonctionnalité. (Joshua Morony 2017)

#### 6.1.4.4 Mes composants

##### *6.1.4.4.1 Searchbar*

Ce composant a été créé pour pouvoir réutiliser la barre de recherche avec la fonctionnalité d'auto-complétion de l'API Google mentionné précédemment (Autocomplete, section 4.1). Il est aussi composé de deux boutons, le premier permettant de réinitialiser l'emplacement de l'utilisateur à son emplacement actuel, et le second permet de filtrer les recherches par catégories et par distance.

Figure 21 : Composant Searchbar




(Alex Perritaz 2018)

La barre de recherche permet à l'utilisateur de saisir une adresse Suisse et de se voir suggérer des propositions en fonction de sa saisie afin qu'il n'ait pas besoin d'écrire l'adresse en entier. De plus, elle permet de s'assurer que le format des adresses soit traité de manière identique.

Elle est incluse dans les pages d'accueil (HOME) dans laquelle les résultats sont affichés par ordre de nouveauté, ainsi que la page (NEARBY) qui elle les affiche par ordre de proximité.

#### *6.1.4.4.2 Itemframe*

Suite à la validation de l'adresse saisie, du choix des catégories et de la distance définie dans le filtre du composant *Searchbar*, une requête est envoyée à la base de données. Les éléments retournés correspondent aux articles correspondant aux critères fixés et chacun d'eux sont affichés dans un composant *ItemFrame*.

Figure 22 : Composant Itemframe

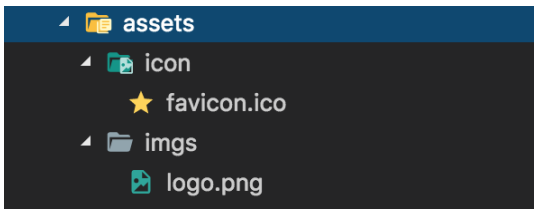

(Alex Perritaz 2018)

Lorsque ce composant est cliqué, une nouvelle requête est envoyée à la base de données pour obtenir les informations correspondant au commerce à qui l'article appartient et en affiche les détails sur la page *Store*.

### 6.1.5 Assets

Ce dossier contient principalement les images, logos et icônes de l'application. (Ionic Framework 2017)

Figure 23 : Dossier des assets

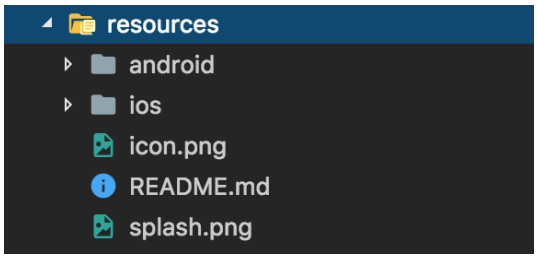



(Alex Perritaz 2018)

### 6.1.6 Ressources

Ce dossier contient les icônes et images de chargement des tailles nécessaires pour s'adapter parfaitement aux différentes résolutions d'écrans. Elles sont générées automatiquement par Ionic. Lorsque l'icône ou l'image de chargement est modifié, il faut régénérer tout le contenu du dossier en effectuant la commande suivante : *$ ionic cordova resources [<platform>]* avec comme paramètres la plateforme *ios* ou *android* (Ionic Framework 2018c)

Figure 24 : Dossier des ressources




(Alex Perritaz 2018)

### 6.1.7 Index.html

Le fichier *.src/index.html* est le point d'entrée principal de l'application, elle initialise les scripts, importe le CSS et s'occupe de l'amorçage.

Figure 25 : Fichier index.html


(Alex Perritaz 2018)

La seule chose ajoutée dans ce fichier est le script permettant d'utiliser l'API Google ainsi que la clé de celle-ci :

*<script src="https://maps.googleapis.com/maps/api/js?v=3&key=[KEY]&libraries=places"></script>*

### 6.1.8 Config.xml

Cordova se sert de ce fichier pour définir les paramètres de compilation du projet. Ce fichier est automatiquement généré en même temps que le projet. Il permet aussi aux développeurs de spécifier les « metadata » de l'application.

Figure 26 : Fichier config.xml


(Alex Perritaz 2018)

Dans le cas de celles-ci, on spécifie les paramètres concernant l'image et l'animation de chargement. (Chris Griffith 2018)

## 6.2 Backend

### 6.2.1 Interfaces

Les interfaces permettent de structurer et de vérifier l'utilisation attendue d'un objet auquel elle accède. Par exemple si un objet Article est créé et que l'on veut assigner une valeur à *name* de type *boolean* l'éditeur de code ne va pas afficher d'erreurs. C'est là qu'intervient l'interface, le fait d'avoir spécifié le type de données attendu, l'éditeur n'acceptera pas d'autres types pour cet attribut que celui défini. Le fait de créer une interface permet d'éviter les erreurs d'inattention lors du typage, gagner du temps sur le débogage et simplifier l'utilisation de ses propres objets. (richardshergold 2017)

Dans le cadre de ce projet, l'interface *Article* a été créé pour être utilisé avec le composant *ItemFrame* afin d'y afficher l'objet.

Figure 27 : Interface Article

```
export interface Article{
    id?                 : number;
    name                : string;
    description         : string;
    image               : Blob;
    distance            : number;
    price               : number;
    date                : Date;

    cat_id              : number;
    cat_icon            : Blob;
    cat_name            : string;

    sto_id              : number;
    unit                : string;
}
```

(Alex Perritaz 2018)

L'interface *Store* quant à lui a été créé pour afficher les informations du magasin sur la page *Store* de l'application.

Figure 28 : Interface Store

```
export interface Store{
    id?                 : number;
    name                : string;
    description         : string;
    phone               : string;
    logo                : Blob;
    open                : Time;
    close               : Time;

    city                : string;
    state               : string;
    postalcode          : string;
    country             : string;
}
```

(Alex Perritaz 2018)

### 6.2.2 Providers

Provider, Service ou Injectable désignent la même chose, ce sont les classes qui incluent @Injectable(). Ils permettent l'utilisation des fonctionnalités définies dans les différents composants de l'application. Ils aident à améliorer la lisibilité, la maintenabilité ainsi que la structure du code. (Morony 2017)

#### 6.2.2.1 Création

Pour créer un nouveau provider comme pour la plupart des éléments dans Ionic, il suffit de passer par le CLI (Command Line Interface) du Framework et de saisir la commande suivante : *$ ionic generate provider [<nom>] (Ionic Framework 2018c)*

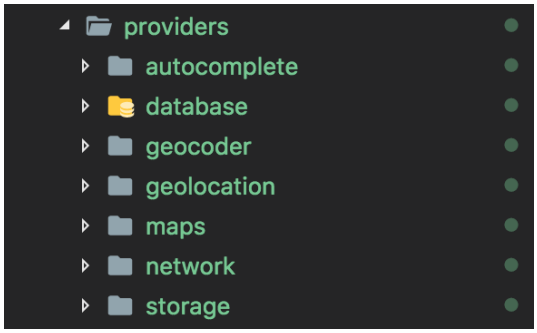
#### 6.2.2.2 Structure

La structure d'un provider est un simple document TypeScript servant à définir les attributs, propriétés et méthodes du service en question.

#### 6.2.2.3 Implémentation

Les providers suivants : autocomplete, geocoder, geolocation, maps et storage étant déjà présentés dans d'autres sections du document, ils ne seront pas repris dans ce point.

Figure 29 : Dossier contenant les différents providers




(Alex Perritaz 2018)

Le provider de base de données (database), permet de récupérer les articles en fonction des catégories sélectionnées, de la distance de recherche, de l'adresse saisie ou la position actuelle l'utilisateur. Elle permet aussi la récupération des informations d'un commerce en fonction de son identifiant. Elle sert entre-autres à la communication entre l'application et la base de données.

Le provider network permet d'obtenir les informations relatives à la connexion de l'appareil. Il observe l'état du réseau pour vérifier les changements liés à la connectivité de l'appareil ainsi que le type de connexion utilisé.

### 6.2.3 Stockage de données local

#### 6.2.3.1 Storage Ionic

Le système de stockage local de données Ionic permet d'enregistrer sur la mémoire de l'appareil les réglages et préférences utilisateur. Cela permet de sauvegarder les préférences de recherche ainsi que les commerces pour lesquels il porte intérêt sans pour autant avoir besoin de les stocker dans une base de données. Le fait de les avoir sur la mémoire locale, évite à l'utilisateur de devoir se créer un compte pour pouvoir avoir accès à ses informations sur une base de données. La RGPD (règlement général sur la protection des données) étant entré en vigueur le 25 mai 2018, il devient important de faire attention aux données conservées sur un serveur. C'est pourquoi dans cette application le choix a été fait de n'enregistrer les données que localement.

Le package permettant le stockage local de données est installée par défaut dans les versions de Ionic supérieures à 1.0. Il suffit donc d'importer le module IonicStorageModule depuis @ionic/storage au fichier *app.module.ts* et de l'importer dans le composant pour pouvoir en utiliser les fonctionnalités. Dans le cas du projet il a été utilisé dans le provider *storage*. (Ionic Framework 2018d)

### 6.2.4 Stockage de données distant

#### 6.2.4.1 MAMP

MAMP est un acronyme pour **M**acintosh **A**pache **M**ySQL **P**HP. Comme son nom l'indique, il s'agit d'un serveur web Apache qui répond aux requêtes du navigateur. La partie MySQL stocke les données et la partie PHP permet d'interagir avec ce dernier.

Le produit est décliné en plusieurs variantes, un par OS mais les plus communs sont :

- MAMP pour Macintosh
- XAMP pour Linux
- WAMP pour Windows

Ils permettent l'utilisation du Framework Laravel dans le cadre du projet ainsi que le stockage des données (partie base de données).

##### *6.2.4.1.1 PHPMyAdmin*

Le gestionnaire de base de données PHPMyAdmin est inclus par défaut dans MAMP. On récupère, modifie et supprime ces données grâce à des requêtes SQL reçues du Framework Laravel.

#### 6.2.4.2 Laravel

##### *6.2.4.2.1 Mise en place*

Pour mettre en place le Framework Laravel, il faut d'abord s'assurer d'avoir un serveur de données utilisant PHP 7.1.3 ou plus récent. Dans le cas de ce projet, il s'agit du serveur local MAMP précédemment mentionné.

Ensuite il faut installer Composer, un outil de gestion de dépendances en PHP. Il permet d'installer et de mettre à jour automatiquement les librairies nécessaires au projet. Pour installer Composer on peut utiliser le gestionnaire de paquets Homebrew précédement utilisé pour installer Ionic et Node. Saisir dans le terminal (invite de commandes pour Windows) les commandes suivantes :

```
brew homebrew/homebrew-php
brew install composer
```

Pour vérifier que composer a bien été installé exécuter la commande suivante :

```
composer --version
```

Si tout est installé comme il le faut il est censé retourner la version de Composer :

```
Composer version 1.5.5 2017-12-01 14:42:57
```

(Marcin Nabiałek 2018)

Une fois Composer installé, ouvrir le terminal et saisir l'instruction suivante pour créer un nouveau projet Laravel :

```
composer create-project --prefer-dist laravel/laravel <nom>
```

#### *6.2.4.2.2 Structure*

La Structure de Laravel peut paraître intimidante aux premiers abords, mais il est en réalité plutôt simple à comprendre lorsqu'on prend un peu de temps pour se familiariser avec.

Les composants principaux de la structure de Laravel sont :

- Un fichier .**env** qui permet de spécifier les informations concernant la base de données qui sera liée à Laravel.
- De **Models** qui représentent une table de la base de données en tant qu'objet pour en simplifier l'utilisation dans Laravel.
- De fichiers **Migrations** permettant de générer les tables de la base de données et des **Seeds** qui rempliront celle-ci une fois créée.
- De **Controllers** qui contiennent des méthodes appelées par les **Routes** définies dans le fichier *api.php*.

Afin de comprendre comment ces fichiers fonctionnent ils vont être expliqués un peu plus en détail dans les sections ci-dessous.

**Fichier .env**

Le fichier **.env** permet de spécifier les informations de la base de données à laquelle Laravel communique. Les champs modifiés pour la configuration du projet sont les suivants :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=8889
DB_DATABASE=bachelor_01
DB_USERNAME=root
DB_PASSWORD=root
```

### Models

Les models sont des représentations de la base de données en tant qu'objet, ils permettent de simplifier l'utilisation de ces tables dans Laravel. Ils permettent de récupérer les données des tables ainsi que d'y en ajouter des nouvelles.

Exemple d'un Model d'article :

```
class Article extends Model
{
    protected $table = 'article';
    protected $fillable = ['art_name','art_descr','art_img','art_price'];
}
```

### Migrations

Les migrations permettent de décrire en PHP la structure de la base de données. Lorsqu'il y a une modification à faire sur la base de données, il suffit de le faire dans les fichiers de migrations et de lancer la migration, ce qui aura pour effet de la modifier de la même manière que l'environnement de développement. (Grafikart.fr 2015a)

Un exemple du contenu d'un fichier de migration :

```
Schema::create('article', function (Blueprint $table) {
    $table->increments('art_id');
    $table->string('art_name',35);
    $table->text('art_descr');
    $table->binary('art_img');
    $table->decimal('art_price',5,2);

    // Foreign Keys - Store
    $table->integer('fk_sto_id')->unsigned();
    $table->foreign('fk_sto_id')->references('sto_id')->on('store');
    // Foreign Keys - Category
    $table->integer('fk_cat_id')->unsigned();
    $table->foreign('fk_cat_id')->references('cat_id')->on('category');
    // Foreign Keys - Measure unit
    $table->integer('fk_unit_id')->unsigned();
    $table->foreign('fk_unit_id')->references('unit_id')->on('measure_unit');
    $table->timestamps();
});
```

Lorsque l'on veut mettre à jour un champ ou des éléments de la base de données il suffit de saisir dans l'invite de commande :

```
php artisan migrate:refresh
```

### Seeds

Les seeds permettent de remplir toutes les tables de la base de données automatiquement lors du développement de l'application pour éviter de devoir les remplir à nouveau soi-même après des tests effectués sur celle-ci. Permettant ainsi de repartir sur une base propre.

Un exemple de seed générant des données dans la table article :

```
public function run()
{
    for($i = 1; $i <= 30; $i++) {
        DB::table('article')->insert([
            'art_name' => "art n°$i",
            'art_descr' => "description article n°$i",
            'art_img' => '',
            'art_price' => rand(0,10),

            'fk_sto_id' => $i,
            'fk_cat_id' => rand(1,6),
            'fk_unit_id' => rand(1,4),
            'created_at' => '2018-09-01 '. rand(0,23) .':00:00'
        ]);
    }
}
```

Lorsque l'on veut repeupler la base de données il suffit de saisir dans l'invite de commande:

```
php artisan migrate:refresh --seed
```

### Controllers

Lorsqu'une requête http est recue par Laravel celle-ci est assignée à une route qui exécutera le code à l'intérieur. On pourrait mettre toute la logique du code dans les routes mais ce n'est évidemment pas une bonne solution. Il est donc plus judicieux d'organiser la partie logique de notre code dans les Controllers. (Grafikart.fr 2015b)

Un exemple de méthode dans un Controller :

```
public function getByMostRecent(Request $request) {

        $myLat = $request->input("myLat");
        $myLng = $request->input("myLng");
        $dist = $request->input("dist");
        $order = $request->input("order");
        $excludedCategories = $request->input("cat");
        $parsed = implode(",",$excludedCategories);

         $query = ...

         $results = \DB::select($query, [1]);
        return $results;
}
```

#### 6.2.4.3 Architecture REST

##### *6.2.4.3.1 Routes*

Les routes sont définies dans le fihcier */routes/api.php*, elles permettent de faire correspondre un morceau de code à différents URL de notre application. (Grafikart.fr 2015c)

Par exemple lorsque dans le code de l'application cette méthode appelle l'URL suivant :

```
path = "http://localhost:8888/MyService/public/api/category?";

header = {
  'Content-Type': 'application/json',
  'X-Requested-With': 'XMLHttpRequest',
  'Accept': 'application/json'
}

Data = {
   'param1': 'premier',
   'param2': 'second',
}

this.http.get(path, data , this.header).then(data => {...})
```

La chemin (path) envoyé par le code va donc envoyer un http.get à cette URL. Le point d'entrée dans Laravel la redirige ensuite dans le fichier *api.php* qui va faire correspondre cette URL à la fonction getCategories du contrôleur de la catégorie CategoryController.

```
Route::get('/category', "CategoryController@getCategories");
```

#### *6.2.4.3.2 Routes utilisées*

Toutes les routes utilisées commencent par http://localhost:8888/MyService/public/api/

Lorsque l'on veut faire appel à une des routes par une méthode this.http.<verbe> il faut y passer en paramètre le header suivant :

```
header = {
  'Content-Type': 'application/json',
  'X-Requested-With': 'XMLHttpRequest',
  'Accept': 'application/json'
}
```

Tableau 13 : Routes utilisées

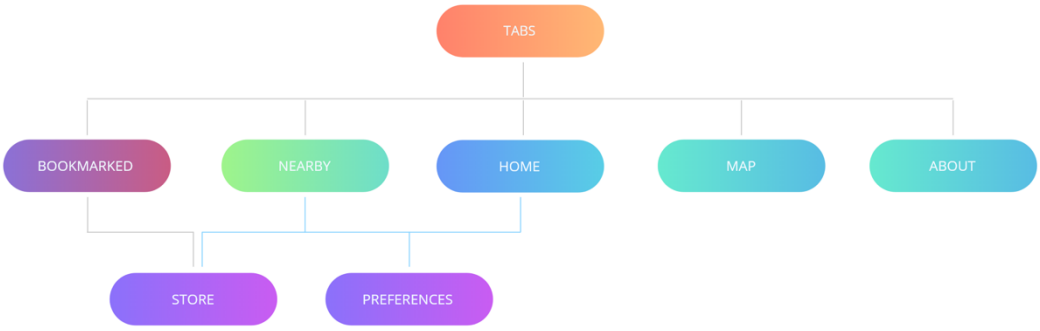
| N° | Verbe | URL | Middleware | Paramètres | Résultat | Description |
|---|---|---|---|---|---|---|
| 1 | GET | /category | - | - | 200 | Récupère les différentes catégories d'aliments |
| 2 | GET | /article | - | {<br>Lat<br>Long<br>Distance<br>ExCategories[]<br>Order<br>} | 200 | Récupère tous les articles dans un certain rayon en excluant certaines catégories d'aliment et en triant par date ou proximité |
| 3 | GET | /articleByStore | - | {<br>ID<br>Art_ID<br>} | 200 | Récupère les articles appart au commerce correspondant à un certain ID |
| 4 | GET | /store | - | {<br>ID<br>} | 200 | Récupère les données concernant un commerce en fonction de son ID |

# 7. Interfaces de l’application

Une fois toute la partie technique effectuée, il ne reste que le visuel à implémenter. Pour cela on va représenter la navigation entre les pages des applications et leur utilité.

## 7.1 Hiérarchie des pages

Figure 30 : Navigation au sein des pages




(Alex Perritaz 2018)

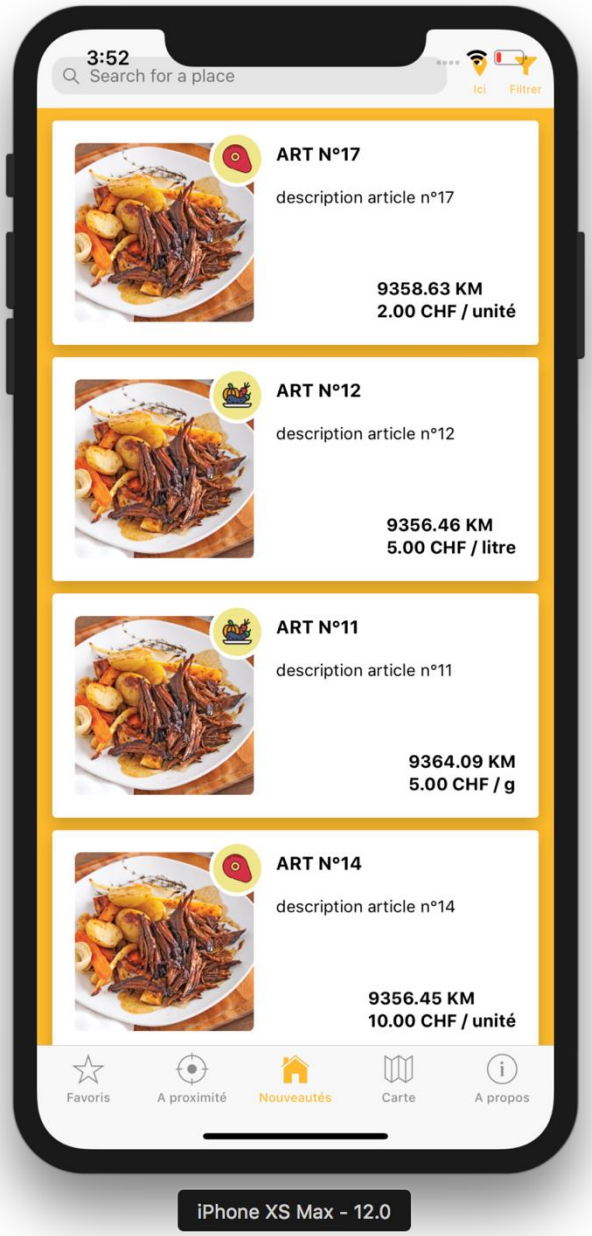
La navigation entre les différentes pages de l’application a été rendue aussi simple que possible pour rendre l’accès à chacune de ses fonctionnalités pour une utilisation rapide et de pouvoir avoir accès aux données souhaitées en un clin d’œil. On peut alterner entre chacune des pages avec une simple pression sur la section désirée.

## 7.2 Pages

### 7.2.1 Accueil

La page accueil est la page qui est affichée par défaut une fois que l'application est lancée. Dans la partie supérieure de l'écran, le composant Ionic Ion-Navbar est remplacé par le celui créé *Searchbar* qui implémente le service d'auto-complétion d'adresses mentionné plus tôt.

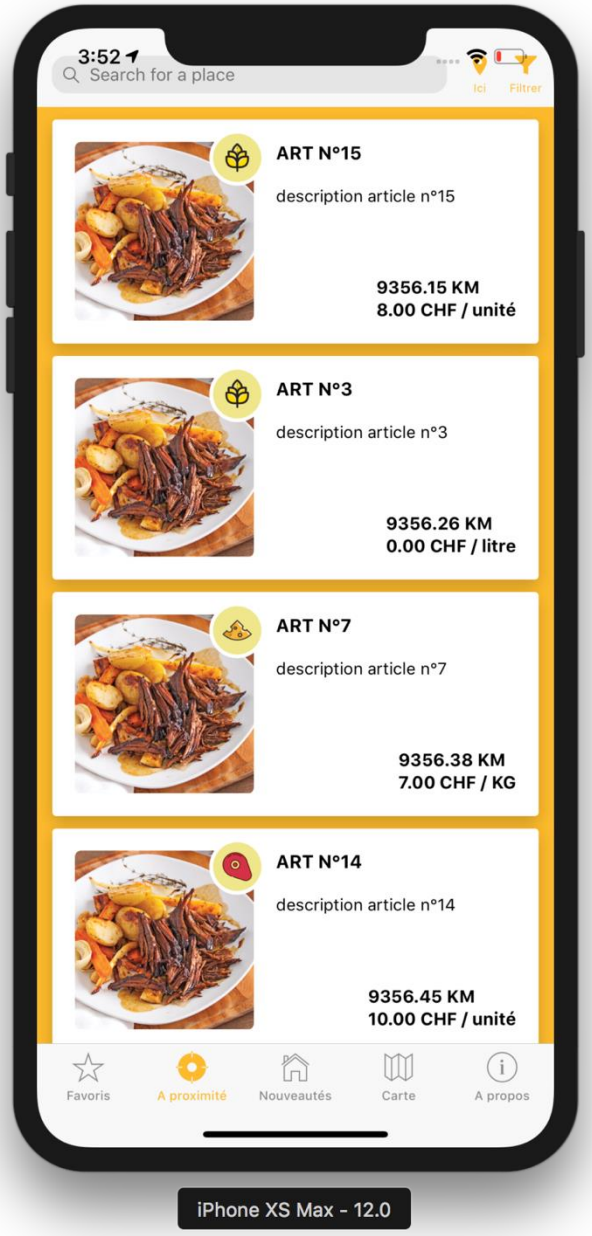
Figure 31 : Capture d'écran - Page Home




(Alex Perritaz 2018)

La page « A proximité » affiche est similaire à la page d'accueil « Nouveautés » mais affiche les offres les plus proches de nous, dans notre rayon de recherche.

Figure 32 : Capture d'écran - Page Proximity

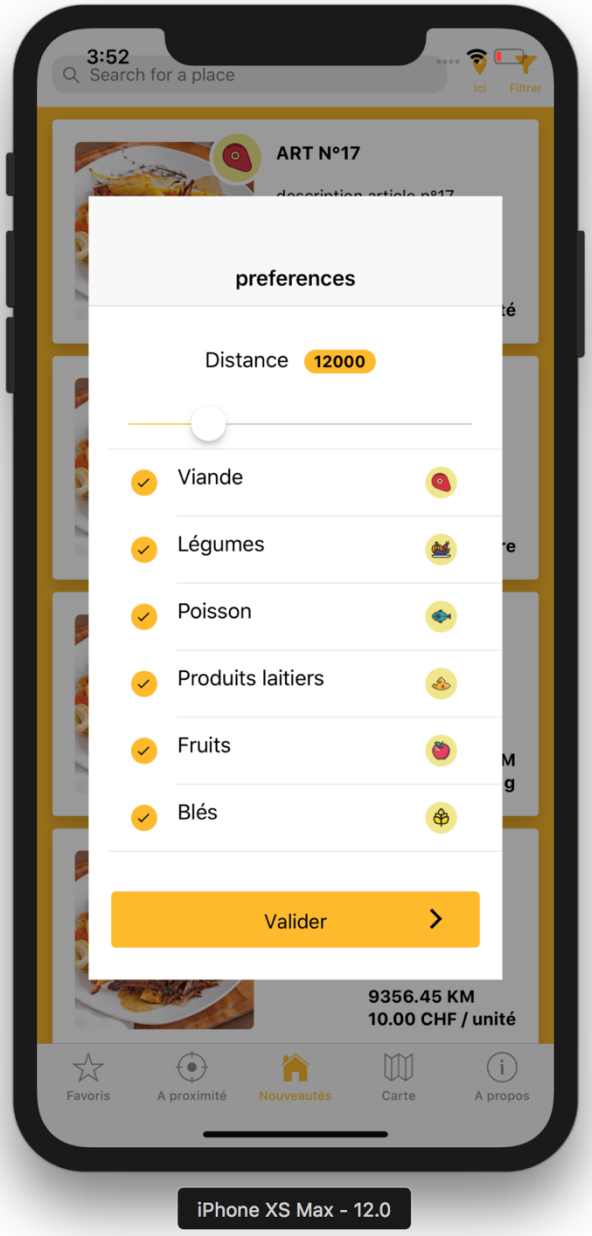



(Alex Perritaz 2018)

Les éléments affichés sont les composants *Itemframe*, ils correspondent aux articles les plus récents se trouvant dans notre rayon de recherche et filtré par type de catégorie sélectionné.

Lorsque le bouton filtrer est cliqué, une fenêtre modale s'ouvre et permet de définir notre rayon de recherche, ainsi que les catégories que l'on souhaite ou ne souhaite pas afficher. Une fois les paramètres définis, sauvegarde la configuration dans la mémoire locale du téléphone et actualise la liste des articles.

Figure 33 : Capture d'écran - Page Preferences




(Alex Perritaz 2018)

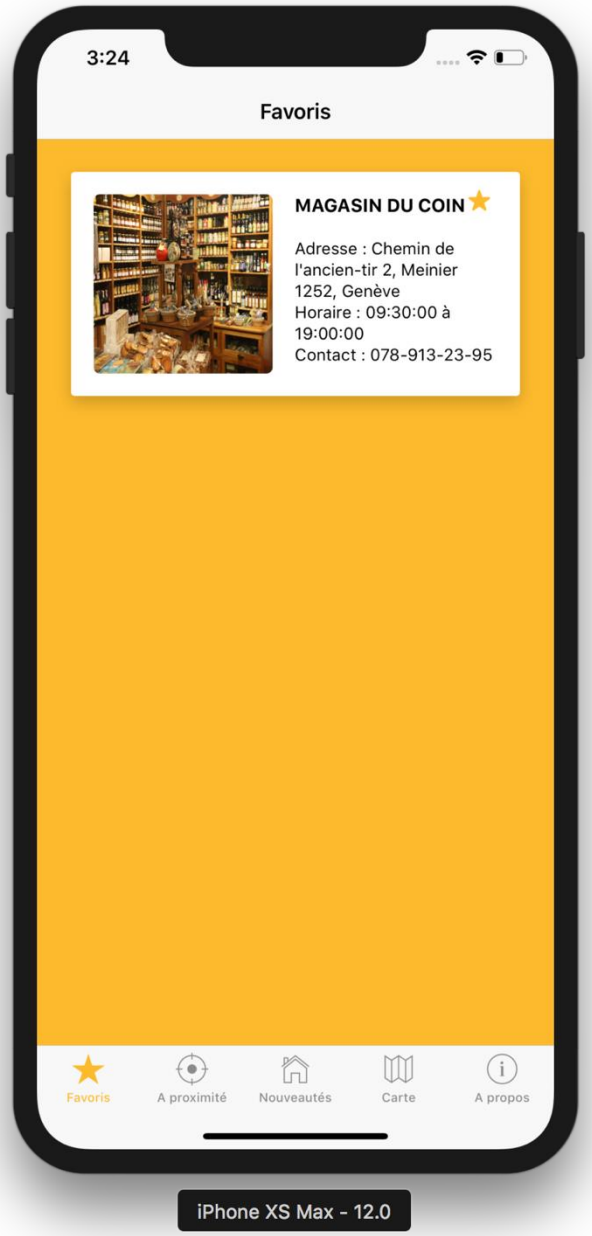
Lorsqu'un de ces éléments est cliqué cela nous renvoi à la page de la boutique qui propose le produit en question. La page de la boutique affiche les informations tels que le nom, l'adresse, les heures d'ouvertures et le numéro de téléphone ainsi que les autres articles proposés par ce même vendeur.

Figure 34 : Capture d'écran - Page Store


(Alex Perritaz 2018)

Lorsqu'une boutique nous plaît, il est possible de l'ajouter aux favoris. Ces favoris sont ensuite accessibles sur l'onglet « Favoris » permettant d'avoir accès rapidement à leur page et d'y afficher les offres qu'elles proposent actuellement.

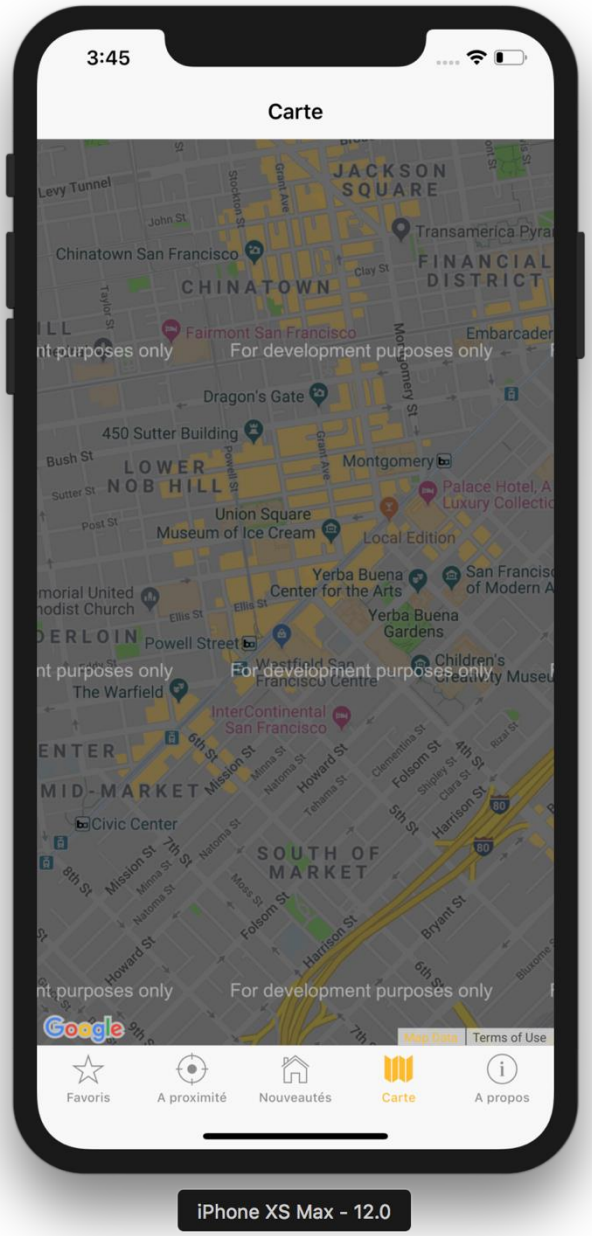
Figure 35 : Capture d'écran - Page Bookmarked




(Alex Perritaz 2018)

Une des fonctionnalités malheuresement pas encore réalisable est celle de Google Maps, qui pourrait permettre l'affichage des différents magasins qui auraient des offres à proximité.

Figure 36 : Capture d'écran - Page Maps




(Alex Perritaz 2018)

Finalement, l'onglet « A Propos » qui servirait d'expliquer les raisons pour lesquelles avoir développé une telle application et dans quel but. Pour moi il s'agit de la réalisation du travail de bachelor mais il est aussi question d'aider à promouvoir les marchés locaux et de participer à la réduction du gaspillage alimentaire.

Figure 37 : Capture d'écran - Page About


(Alex Perritaz 2018)

Le développement de chacun de ces pages ne s'est pas toujours déroulé comme prévu, comme pour la page Maps, elle n'a pu être implémenté à cause de problèmes survenus lors du développement de l'application. Ces problèmes vont être traités dans la section qui va suivre.

# 8. Problèmes rencontrés

Tableau 14 : Méthodes Asynchrones

| **Promises / Asynchrone – Synchrone** |
| --- |
| Pour comprendre ce problème il faut tout d'abord comprendre la différence entre l'exécution d'une tâche synchrone et d'une tâche asynchrone.<br><br>Lorsqu'une tâche est exécutée de manière asynchrone on n'attend pas la fin de celle-ci avant de passer à la suivante. Par défaut, c'est ce qui se passe en TypeScript.<br><br>Figure 38 : Tâches asynchrones<br><br>1 Fil : /<----------Tâche 1---------->/<br>/<----Tâche 2--->/<br>/<----Tâche 3---->/<br><br>(Nuriddin Kasimov 2017)<br><br>Lorsqu'une tâche synchrone est exécutée on attend la fin de cette tâche pour passer à la suivante. (Nuriddin Kasimov 2017)<br><br>Figure 39 : Tâches synchrones<br><br>1 Fil : /<--Tâche 1-->/ \|<br>\| /<--Tâche 2-->/ \|<br>\| /<-------Tâche 3-------->/<br><br>(Nuriddin Kasimov 2017)<br><br>Si une méthode T2 dépend du résultat de la méthode qui précède T1, et que celle-ci débute avant la récupération du résultat de T1, il faut s'attendre soit à une réponse erronée soit à ce qu'elle retourne une exception. |
| **Solution** |
| Pour éviter les problèmes liés aux méthodes asynchrones, il faut faire appel aux promises.<br><br>Par exemple, si l'on veut enchainer les méthodes T1, T2 et T3 de manière synchrone, il faut les déclarer séparément et les appeler de la manière suivante dans T1. (Oleksii Rudenko 2015)<br><br>`promise.then(T2()).then(T3())` |

Tableau 15 : Laravel - Header manquant

| **Laravel - Access Control Object Origin – http header** |
| --- |
| L’erreur survient lorsqu’on essaie d’effectuer une requête http depuis l’application sur le Framework Laravel. Il requiert que l’on entre un certain header pour pouvoir avoir accès aux données. |
| **Solution** |
| Pour envoyer la requête correctement il faut donc ajouter le header suivant aux paramètres de la méthode en question.<br>`header = {`<br>`'Content-Type': 'application/json',`<br>`'X-Requested-With': 'XMLHttpRequest',`<br>`'Accept': 'application/json'`<br>`}`<br>`this.http.get(path, data, this.header).then(data => {…});` |

Tableau 16 : Laravel - Encodage

| **Laravel – Problème d'encodage** |
| --- |
| Lors de la création de la base de données il faut spécifier l'encodage désiré, dans ce cas il s'agit de l'encodage utf8_general_ci. Mais lors de la migration de *Laravel*, il est par défaut assigné à utf8_unicode et pose donc un problème. |
| **Solution** |
| La solution qui fonctionne est de modifier dans le fichier *config/databases.php* dans la section suivante par le type d'encodage voulu. (boptom 2017)<br>`'mysql' => ['charset' => 'utf8mb4', 'collation' => 'utf8mb4_unicode_ci'];` |

Tableau 17 : Autocomplete - Restriction à une région

| **City limitation – Autocomplete** |
| --- |
| Lorsque l'utilisateur commence à saisir dans la *Searchbar*, il est préférable d'avoir les adresses les plus proches de celui-ci qui soit affichées et non une adresse à l'autre côté du globe. Il est donc préférable de limiter les champs de recherche selon moi au canton ou à la région. Peu de personnes font 4 heures de trajet pour un article en promotion. |
| **Solution** |
| Il n'est possible pour l'instant que de restreindre le champ de recherche pour le service d'autocomplete de Google au pays souhaité. Si cela est vraiment nécessaire, il est possible de définir soi-même les limites de la région voulue et de les limiter à cette région uniquement mais ce n'est pas la solution choisie pour cette application. (xomena 2017) |

Tableau 18 : Google Maps - Erreur

| **Google Maps – Production purposes only** |
| --- |
| Depuis le 11 Juin Google a modifié ses tarifs pour l'utilisation de son API Maps le rendant payant. L'affichage de la carte google est assombrie et ses fonctionnalités réduites à l'affichage uniquement rendant l'API peu pratique d'utilisation. |
| **Solution** |
| Il faut entrer une méthode de paiement valide afin de bénéficier d'un équivalent de $200 par mois d'utilisation offert par Google ou d'utiliser une autre API que celle de Google. (Marco 2018) |

Tableau 19 : Autocomplete - Erreur

| **Autocomplete Google API** |
| --- |
| Il arrive parfois que l'API Google d'autocomplete cesse de fonctionner pour des raisons qui me sont inconnues alors que tout peut bien fonctionner durant plusieurs jours. Il décide tout d'un coup de me retourner des erreurs 403 (Forbidden Error) pouvant être liés à une cinquantaine d'autres codes d'erreurs. Andrew Powell-Morse explique sur son post les différentes raisons pouvant être à l'origine de ce problème et qu'il n'est pas peu fréquent de tomber dessus. (Andrew Powell-Morse 2017)<br><br>Parfois l'API décide que j'ai aussi atteint les limites (gratuites) de 100'000 requêtes par jour alors que la journée vient de commencer pour des raisons que j'ignore encore malgré les recherches. |
| **Solution** |
| Je n'ai pas trouvé de solution au problème. Il suffit pour mon cas quelques heures ou que la journée se termine pour que cela passe et fonctionne à nouveau. |

# 9. Perspectives d'avenir

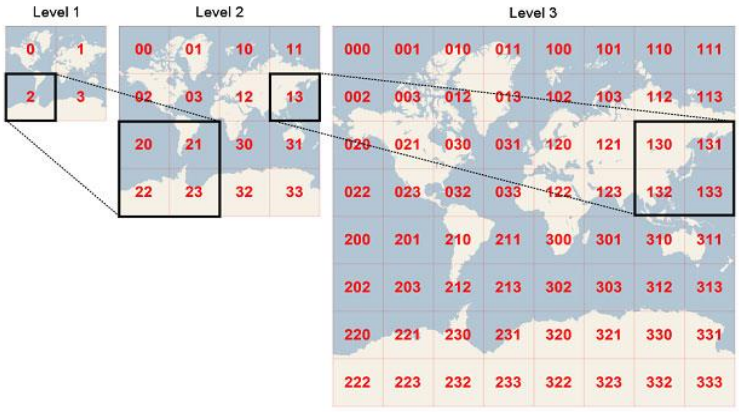
## 9.1 Améliorations envisageables

### 9.1.1 De Haversine à GeoHash

L'inconvénient avec la formule de Haversine est qu'on doit stocker la latitude et la longitude dans deux attributs différents. De par ce fait, on est obligé à créer une table pour stocker ces deux valeurs avec un index pour les lier à une adresse. Alors que la méthode de GeoHash convertit ces deux adresses en une seule ce qui permet donc la suppression de cette table et donc de la stocker directement dans la table du commerce concerné. Le calcul de Haversine étant très couteux au niveau du temps et des ressources, GeoHash est la solution idéale si beaucoup de données sont à traiter (Stevenson 2013)

Le principe de GeoHash consiste à découper en régions chaque partie du globe en parts égales identifiée par une chaine de nombres et chacune de ces régions en sous régions en y ajoutant à la suite de la chaîne de la zone correspondante comme illustré dans l'image ci-dessous. Pour retrouver les éléments à proximité on choisirait la taille du GeoHash adapté à la distance de recherche et effectuer les recherches uniquement dans cette zone.

Figure 40 : Illustration du fonctionnement de GeoHash




(Blum 2018)

La précision de cette méthode augmente en fonction du nombre de chiffres dont la région est composée comme constatée dans l'image. La distance correspondante au nombre de chiffres contenu dans le GeoHash est définie de la manière suivante.(radouxju 2014)

| # | km |
|---|---|
| 1 | ± 2500 |
| 2 | ± 630 |
| 3 | ± 78 |

| 4 | ± 20 |
|---|---|
| 5 | ± 2.4 |
| 6 | ± 0.61 |
| 7 | ± 0.076 |

| 8 | ± 0.019 |
|---|---|
| 9 | ± 0.0024 |
| 10 | ± 0.00060 |
| 11 | ± 0.000074 |

### 9.1.2 Maps

Pour résoudre le problème du fait que l'API de Google Maps soit devenue payante, on peut décider d'inscrire ses informations bancaires chez Google pour bénéficier de l'équivalent de $200 par mois d'utilisation gratuite. Soit comme le suggère Pawel Dulak sur ce forum de passer par une autre API comme OpenStreetMap, mais on y perdrait les fonctionnalités comme le GoogleDirections permettant d'établir un itinéraire jusqu'à l'endroit souhaité ou encore le GoogleStreetView permettant de se déplacer dans les rues comme si l'on y était, car ces deux services sont liés à Google.(Paweł Dulak 2018)

On peut aussi y ajouter un cluster de points, indiquant tous les commerces enregistrés dans notre base de données, permettant à l'utilisateur d'avoir une représentation visuelle d'où ils se situent sur la carte grâce à la démarche proposée par Josh Morony en utilisant l'API de Google Maps. (Morony 2016)

Cependant je n'ai pas réussi à trouver cette fonctionnalité dans Ionic pour OpenStreetMap encore.

## 9.2 Fonctionnalités supplémentaires

### 9.2.1 Application vendeur

Une application vendeur composé d'un écran de **Login**, **Inscription**, **Accueil**, **Ajout**. Il est aussi minimaliste que possible pour permettre une gestion simple et rapide de ses articles.

#### 9.2.1.1 Page : Login

Un écran de **login** permet :

- Se connecter grâce à ses identifiants → redirige à la page d'**accueil**
- S'inscrire → redirige à la page d'**inscription**

#### 9.2.1.2 Page : Inscription

Un écran d'**inscription** permettant de définir les informations suivantes concernant le commerce :

- Nom
- Identifiant
- Mot de passe
- Logo
- Adresse
- Téléphone
- Heure ouverture
- Heure fermeture

#### 9.2.1.3 Page : Accueil

La page d'**accueil** permettant :

- D'afficher les informations du commerce
- D'afficher une liste des produits actuellement en promotion
    - Affichant la quantité disponible
    - Permet d'incrémenter/réduire la quantité de produit disponible
    - Possibilité de modification du produit proposé
    - Possibilité de suppression du produit proposé
- Un bouton permettant l'ajout d'un nouveau produit
    - Redirige à la page d'**ajout de nouveau produit**

#### 9.2.1.4 Page : Ajouter

La page d'**ajout de nouveau produit** :

- Ajouter une image (de la bibliothèque ou de la prendre en photo directement)
- Ajouter le nom du produit
- Définir la catégorie du produit
- Ajouter une description du produit
- Ajouter le prix du produit
- Ajouter l'unité de mesure pour le prix donné

### 9.2.2 Réservation d'un produit

Lors de la consultation des articles en vente, ajouter la possibilité d'effectuer une réservation et prépayer pour venir la chercher plus tard dans la journée. Cette fonctionnalité serait basée sur des micro transactions faites dans l'application.

# Conclusion

Développer une application avec quelques fonctionnalités peut paraître comme quelque chose de facile, mais c'est surtout lorsque l'on doit assembler le tout et les faire fonctionner ensemble que les choses se compliquent. Il faut avant de se lancer tête baissée à coder notre application, savoir quels sont les différents outils mis à notre disposition et qu'est-ce qu'elles offrent de plus par rapport à un autre, et surtout lesquels nous conviennent le mieux.

Une fois les outils en main il faut apprendre à les utiliser et connaître leurs capacités ainsi que leurs limites, essayer de voir ce qui fonctionne et ce qui ne fonctionne pas, mais surtout de comprendre pourquoi. Il faut ensuite commencer à établir les différentes fonctionnalités de notre application et les réaliser.

Après l'implémentation des fonctionnalités principales, il faut encore être capable de les relier entre elles pour faire fonctionner le tout ensemble. C'est lorsqu'on met en place tous ces éléments, qu'on réalise qu'il ne suffit pas d'être en possession des pièces du puzzle, mais faut-il encore savoir les assembler.

C'est alors qu'une fois les pièces ensemble que l'on peut admirer fièrement son travail. Je ne cesse de penser déjà aux futures modifications que je pourrais apporter à cette application.

Enfin, j'espère que ce projet pourra continuer à être développé et être utilisée au sein d'une association qui développe le commerce de proximité et par la suite, si le succès escompté est au rendez-vous la développer ailleurs en Suisse, voire dans le monde.

# Bibliographie

ALEX PERRITAZ, 2018. 24 September 2018.

ALEXANDER RUBIN, 2018. Geo Distance Search with MySQL | Database Index | Longitude. *Scribd* [online]. 14 September 2018. [Viewed 14 September 2018]. Available from: https://www.scribd.com/presentation/2569355/Geo-Distance-Search-with-MySQL

ANDREW POWELL-MORSE, 2017. 403 Forbidden Error: What It Is and How to Fix It. *Airbrake Blog* [online]. 12 October 2017. [Viewed 19 September 2018]. Available from: https://airbrake.io/blog/http-errors/403-forbidden-error

BLUM, Stephen L., 2018. *Geo Hashing Chat by Proximity to connect two or more users to a group chat.: stephenlb/geohash-chat-by-proximity* [online]. JavaScript. [Viewed 24 September 2018]. Available from: https://github.com/stephenlb/geohash-chat-by-proximity

BOPTOM, 2017. Defining column collation in migrations. *Laracasts* [online]. 1 January 2017. [Viewed 19 September 2018]. Available from: https://laracasts.com/discuss/channels/general-discussion/defining-column-collation-in-migrations?page=0

CHRIS GRIFFITH, 2018. Safari, the world's most comprehensive tech & business learning platform. *Safari* [online]. 18 September 2018. [Viewed 18 September 2018]. Available from: https://www.safaribooksonline.com/library/view/mobile-app-development/9781491937778/app02.html

CLOCKWISE, 2018. React Native vs. Ionic. The battle of giants in the cross-platform development field. [online]. 27 April 2018. [Viewed 25 September 2018]. Available from: https://clockwise.software/blog/react-native-vs-ionic-cross-platform-app-development-frameworks/

DENIS CORBOZ, 2015. Gaspillage alimentaire. . 19 February 2015. P. 4.

FRANCE NATURE ENVIRONNEMENT, 2013. Du gaspillage alimentaire à tous les étages. . 2013. P. 43.

GENÈVE TERROIR, 2018. Recherche | Genève Terroir - Le portail du terroir genevois. [online]. 18 September 2018. [Viewed 23 September 2018]. Available from:

https://www.geneveterroir.ch/fr/search

GOOGLE, 2018a. Place Autocomplete | Places API. *Google Developers* [online]. 19 June 2018. [Viewed 13 September 2018]. Available from: https://developers.google.com/places/web-service/autocomplete

GOOGLE, 2018b. Overview | Maps JavaScript API. *Google Developers* [online]. 9 December 2018. [Viewed 13 September 2018]. Available from: https://developers.google.com/maps/documentation/javascript/tutorial

GRAFIKART.FR, 2015a. *Laravel 5.0 : Migration & Eloquent* [online]. 2015. [Viewed 24 September 2018]. Available from: https://www.grafikart.fr/formations/laravel/eloquent

GRAFIKART.FR, 2015b. *Laravel 5.0 : Controller* [online]. 2015. [Viewed 24 September 2018]. Available from: https://www.grafikart.fr/formations/laravel/controller

GRAFIKART.FR, 2015c. *Laravel 5.0 : Routes* [online]. 2015. [Viewed 24 September 2018]. Available from: https://www.grafikart.fr/formations/laravel/routes

HUGO HOUYEZ, 2017. Qu'est-ce qu'une api REST ou restful ? | SUPINFO, École Supérieure d'Informatique. [online]. 24 September 2017. [Viewed 24 September 2018]. Available from: https://www.supinfo.com/articles/single/5642-qu-est-ce-qu-une-api-rest-restful

IONIC FRAMEWORK, 2017. Ionic Application Structure. *IonicThemes* [online]. 23 September 2017. [Viewed 15 September 2018]. Available from: https://ionicthemes.com/tutorials/about/ionic-application-structure

IONIC FRAMEWORK, 2018a. Ionic Framework. *Ionic Framework - Geocoder* [online]. 2018. [Viewed 12 September 2018]. Available from: https://ionicframework.com/docs/native/native-geocoder/

IONIC FRAMEWORK, 2018b. Ionic Framework. *Ionic Framework - Geolocation* [online]. 2018. [Viewed 12 September 2018]. Available from: https://ionicframework.com/docs/native/geolocation/

IONIC FRAMEWORK, 2018c. Ionic Framework - Ressources. *Ionic Framework* [online]. 15

September 2018. [Viewed 15 September 2018]. Available from: https://ionicframework.com/docs/cli/cordova/resources/

IONIC FRAMEWORK, 2018d. Ionic Framework - Storage. *Ionic Framework* [online]. 20 September 2018. [Viewed 20 September 2018]. Available from: https://ionicframework.com/docs/storage/

JOSHUA MORONY, 2017. *Setting up Custom Components in Ionic* [online]. 22 August 2017. [Viewed 19 September 2018]. Available from: https://www.youtube.com/watch?v=z3fuSMNQmY4

LONGITUDESTORE, 2018. How big is a degree of Latitude Longitude. [online]. 13 April 2018. [Viewed 14 September 2018]. Available from: http://www.longitudestore.com/how-big-is-one-gps-degree.html

MARCIN NABIAŁEK, 2018. How to install composer via homebrew. *Ask Different* [online]. 26 June 2018. [Viewed 24 September 2018]. Available from: https://apple.stackexchange.com/questions/309490/how-to-install-composer-via-homebrew

MARCO, 2018. Is Google map mobile SDK no longer free? *Stack Overflow* [online]. 5 July 2018. [Viewed 13 September 2018]. Available from: https://stackoverflow.com/questions/50207616/is-google-map-mobile-sdk-no-longer-free

MICHAEL MADDOX, 2010. sql - Is Lazy Loading really bad? *Stack Overflow* [online]. 28 January 2010. [Viewed 15 September 2018]. Available from: https://stackoverflow.com/questions/2155363/is-lazy-loading-really-bad

MIKE, 2017. Ionic and Lazy Loading Pt 1. *The Official Ionic Blog* [online]. 28 April 2017. [Viewed 15 September 2018]. Available from: https://blog.ionicframework.com/ionic-and-lazy-loading-pt-1/

MORONY, Josh, 2015. How to Create a Directive in Ionic 2 & 3 – Parallax Header. *joshmorony - Learn Ionic & Build Mobile Apps with Web Tech* [online]. 3 December 2015. [Viewed 19 September 2018]. Available from: https://www.joshmorony.com/how-to-create-a-directive-in-ionic-2-parallax-header/

MORONY, Josh, 2016. Create Marker Clusters with Google Maps in Ionic 2. *joshmorony - Learn Ionic & Build Mobile Apps with Web Tech* [online]. 19 December 2016. [Viewed 24 September 2018]. Available from: https://www.joshmorony.com/create-marker-clusters-

with-google-maps-in-ionic-2/

MORONY, Josh, 2017. When to Use Providers / Services / Injectables in Ionic. *joshmorony - Learn Ionic & Build Mobile Apps with Web Tech* [online]. 22 June 2017. [Viewed 19 September 2018]. Available from: https://www.joshmorony.com/when-to-use-providersservicesinjectables-in-ionic/

MURRAY BOURNE, 2018. 3-D Earth Geometry. [online]. 17 March 2018. [Viewed 13 September 2018]. Available from: https://www.intmath.com/vectors/3d-earth-geometry.php

@NATGEOFRANCE, 2018. La Terre est-elle vraiment ronde ? *La Terre est-elle vraiment ronde ? | National Geographic* [online]. 16 March 2018. [Viewed 14 September 2018]. Available from: http://www.nationalgeographic.fr/espace/la-terre-est-elle-vraiment-ronde

NURIDDIN KASIMOV, 2017. Quelle est la différence entre une exécution synchrone et asynchrone ? - Quora. [online]. 18 April 2017. [Viewed 20 September 2018]. Available from: https://fr.quora.com/Quelle-est-la-diff%C3%A9rence-entre-une-ex%C3%A9cution-synchrone-et-asynchrone

OFAG, Office fédéral de l'agriculture, 2016. Food Waste. [online]. 27 July 2016. [Viewed 23 September 2018]. Available from: https://www.blw.admin.ch/blw/fr/home/politik/food-waste.html

OLEKSII RUDENKO, 2015. Best Practices for Using Promises in JS. *60devs* [online]. 21 October 2015. [Viewed 19 September 2018]. Available from: https://60devs.com/best-practices-for-using-promises-in-js.html

ORGANISATION DES NATIONS UNIES POUR L'ALIMENTATION ET L'AGRICULTURE (ed.), 2012. *Pertes et gaspillages alimentaires dans le monde ampleur, causes et prévention: étude menée pour le Congrès international Save food! à Interpack 2011, Düsseldorf, Allemagne*. Rome: Organisation des Nations Unies pour l'Alimentation et l'Agriculture. ISBN 978-92-5-207205-8.

PAWEŁ DULAK, 2018. Google Maps shows "For development purposes only." *Stack Overflow* [online]. 16 June 2018. [Viewed 24 September 2018]. Available from: https://stackoverflow.com/questions/50977913/google-maps-shows-for-development-purposes-only

POLYGEO, 2016. coordinate system - What is the approximate error of the Pythagorean Theorem vs.

Haversine Formula in measuring distances on a sphere at various scales? *Geographic Information Systems Stack Exchange* [online]. 20 September 2016. [Viewed 13 September 2018]. Available from: https://gis.stackexchange.com/questions/58653/what-is-the-approximate-error-of-the-pythagorean-theorem-vs-haversine-formula-i

RADOUXJU, 2014. What is the precision of a Geohash. *Geographic Information Systems Stack Exchange* [online]. 29 September 2014. [Viewed 24 September 2018]. Available from: https://gis.stackexchange.com/questions/115280/what-is-the-precision-of-a-geohash

RICHARDSHERGOLD, 2017. Typescript Interfaces. *Ionically Speaking* [online]. 20 January 2017. [Viewed 19 September 2018]. Available from: https://ionicallyspeaking.com/2017/01/20/typescript-interfaces/

SPEC INDIA, 2018. React Native vs. Ionic Comparison [Detailed] : A Step-by-Step Evaluation. *SPEC INDIA* [online]. 27 August 2018. [Viewed 25 September 2018]. Available from: https://www.spec-india.com/blog/react-native-vs-ionic-explained-a-step-by-step-evaluation/

STACK OVERFLOW, 2018. Stack Overflow Developer Survey 2018. *Stack Overflow* [online]. 24 September 2018. [Viewed 24 September 2018]. Available from: https://stackoverflow.com/insights/survey/2017/?utm_source=so-owned&utm_medium=social&utm_campaign=dev-survey-2018&utm_content=social-share

STACKSHARE, 2018. Node.js vs Laravel 2018 Comparison. *StackShare* [online]. 24 September 2018. [Viewed 24 September 2018]. Available from: https://stackshare.io/stackups/laravel-vs-nodejs

STEVAN, Caroline, 2016. Gaspillage alimentaire: la Suisse généreuse avec ses poubelles. *Le Temps* [online]. 21 September 2016. [Viewed 23 September 2018]. Available from: https://www.letemps.ch/societe/gaspillage-alimentaire-suisse-genereuse-poubelles

STEVENSON, James, 2013. Intro to Geohashing. *Cobrain* [online]. 18 September 2013. [Viewed 24 September 2018]. Available from: https://cobrainco.wordpress.com/2013/09/18/intro-to-geohashing/

SWATI DHINGRA, 2018. REST vs. SOAP: Choosing the best web service. *SearchMicroservices* [online]. 24 September 2018. [Viewed 23 September 2018]. Available from: https://searchmicroservices.techtarget.com/tip/REST-vs-SOAP-Choosing-the-best-web-service

TONINATO, Aurélie, 2018. Ils combattent le gaspillage grâce à une application. *TDG* [online]. 22 January 2018. [Viewed 23 September 2018]. Available from: https://www.tdg.ch/geneve/actu-genevoise/combattent-gaspillage-grce-application/story/30001508

UNION MARAÎCHÈRE DE GENÈVE, 2018. Le marché de légumes à prix d'amis | Union Maraîchère de Genève. [online]. 18 September 2018. [Viewed 23 September 2018]. Available from: http://www.umg.ch/content/le-march-de-l-gumes-prix-damis

W3SCHOOLS, 2016. JSON vs XML. [online]. 19 December 2016. [Viewed 13 September 2018]. Available from: https://www.w3schools.com/js/js_json_xml.asp

WIKIPEDIA, 2018. Framework. *Wikipédia* [online]. [Viewed 23 September 2018]. Available from: https://fr.wikipedia.org/w/index.php?title=Framework&oldid=152114423

XOMENA, 2017. jquery - how to filter address according country and city in google maps autocomplete address api. *Stack Overflow* [online]. 16 March 2017. [Viewed 19 September 2018]. Available from: https://stackoverflow.com/questions/42810524/how-to-filter-address-according-country-and-city-in-google-maps-autocomplete-add

# Annexe 1 : Quel avenir pour l’agriculture contractuelle de proximité à Genève ?

Voir document : TIPAlexPerritaz.pdf