



RAPPORT DE STAGE

cyril STERN

Développement de l'application mobile Welp

décembre 2015 - 7 janvier 2015



27 rue du chemin vert 75011 paris

Table des matières

PRESENTATION	7
1.1 LE STAGIAIRE	8
1.2 L'ENTREPRISE	8
1.2.1 LES MOYENS HUMAINS:	9
1.2.2 Chiffres Welp 2015 :	10
1.2.3 LA CONCURRENCE	12
1.2.4 - Abstract	13
1.2.5 - remerciements.....	13
ETUDE DE L'ART	14
1.1 Préambule	15
1.2 - La démarche générale	16
1.3 Le site web Welp	17
1.4 Le font end	17
1.4.1 Interface graphique du web-site.....	17
1.4.2 Le menu.....	17
1.4.3 Les cartes.....	18
CAHIER DES CHARGES	19
1.1 - Présentation	20
1.2 - Technologies imposées :.....	21
ANALYSE	28
1.1 Présentation de la méthodologie UML	29
1.2 - Les Diagrammes de Cas d'Utilisation.....	29
1.2.1 - Représentation graphique.....	29
DCU Front end	30
1.2.2 - Fiche de description textuelle d'un cas d'utilisation ici le login	31
1.2.3 - Diagramme d'état de navigations	32
1.2.4 - Les diagrammes de Séquence Système	35
1.2.5 - Diagramme d'activité	39
1.3 - Les Diagrammes de classe	40
1.5 La couche modele	41

1.4 - Le MDC (Merise)	42
1.5 - Le MLD (Merise)	42
1.5.1 Pourquoi ce modèle ?	44
1.5.2 La base de Donnée	44
La Conception.....	46
1.1 - Méthode de travail Agile	47
1.2 - Diagramme de séquence détaillée.....	49
1.3 - Diagramme de classes participatives.....	50
LE DEVELOPPEMENT.....	51
1.1 - Les categories de codes	52
1.1.1 Front-end	52
1.1.2 Back-end.....	52
1.2 exemple de code	54
Déploiement	58
1.1 Les solutions virtuelles	59
1.1.1 Gulp-Ionic-Chrome	59
1.2 La solution de déploiement mobile (cas android)	61
1.3 Déploiement sur le store	61
SECURITE	62
1.1 La sécurité de notre application	63
CONCLUSION	64
1.1 Mon retour d'expérience	65
BIBLIOGRAPHIE.....	66

Index lexical

Abréviations et définitions utilisées

Abréviations	définitions
API	Application Programming Interface
CSS	Cascading Style Sheet
CLI	Commande Line Interface
DOM	Document Object Model
IDE	Integrated Development Environnement
IHM	Interface Homme Machine
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfert Protocol
HTTPS	Hyper Text Transfert Protocol Secure
JS	JavaScript
JSON	JavaScript Objet Notation
REST	Representationnal State Transfert
SOP	Single One Page
TLS	Transport Layer Security
XSS	Cross-Site Scripting

Tableau 1 :

Index des illustrations

Illustration 1: Equipe Welp	11
Illustration 2: Chiffres Welp	12
Illustration 3: Concurrence	15
Illustration 4: Demarche générale	20
Illustration 5: Catégorie regroupant les entités associée à l'entité user	22
Illustration 6: Catégorie regroupant les entité associé à l'entité need	22
Illustration 7: Home-Page du site Welp	24
Illustration 8: Barre de Menu du site Welp	24
Illustration 9: Exemple de design de carte du site Welp	25
Illustration 10: logo ionic	32
Illustration 11: logo atom	32
Illustration 12: logo gulp	32
Illustration 13: logo ionic lab	33
Illustration 14: présentation de ionic lab avec outil de développement du navigateur chrome.....	33
Illustration 15: arborescence non déplié du projet	33
Illustration 16: logo Gulp.....	34
Illustration 18: logo coffeescript.....	35
Illustration 19: logo angular	36
Illustration 20: logo symfony 2	37
Illustration 21: logo doctrine 2	37
Illustration 22: logo php.....	37
Illustration 23 : logo git	38
Illustration 24 : logo GitLab	38
Illustration 25 : exemple d'un push de version en utilisant le plug in ungit.....	38
Illustration 26: logo markdown.....	39
Illustration 27 : code markdown	39
Illustration 28 : rendu html du code markdown	40
Illustration 29 : splashscreen	47
Illustration 30 : l'icone de l'application	47
Illustration 31 : login page	48
Illustration 32 : la home page	48
Illustration 33 : menu membre welp	48
Illustration 34 : le menu visiteur.....	48

Illustration 35: diagramme de séquence de l'authentification du user	49
Illustration 36 : propagation de variable entre controller en Angular Js.....	51
Illustration 37 : welpy game	52
Illustration 38: diagramme de sequence du jeu welpy	53
Illustration 39: diagramme d'activité entre needer et welper	54
Illustration 40 : diagramme de classe Need.....	55
Illustration 41: diagramme de classe User	56
Illustration 42: interface trello qui a permis la gestion du projet..	62
Illustration 43 : diagramme de gant du planning prévisionnel de l'application welp	62
Illustration 44: Diagramme de séquence détaillé de l'utilisation de champs de recherche dans le side menu droit en y inscrivant une adresse.....	64

PRESENTATION

«*RTFM* »

Proverbe Informaticien

«C'est toujours l'impatience de gagner qui fait perdre»

Louis XIV (cité dans "L'immortel" de FOG)

«J'écoute et j'oublie. Je lis et je retiens. Je fais et j'apprends»

Proverbe chinois

1.1 LE STAGIAIRE

Suite à des études en cinéma et une carrière riche dans la réalisation de clips musicaux, je décide suite à l'effondrement du marché du disque de réorienter ma carrière. Je deviens commercial pour un groupe de distributions de produits de home-cinéma. Je gravis les échelons en passant responsable d'un pôle commercial de vente en ligne et internet.

En 2014, Je passe aussi une certification intégrateur sur control4, qui me permet de me rapprocher d'un univers qui m'attire de plus en plus, la conception informatique. En constante relation avec les développeurs du site quant à l'ergonomie du site, j'en comprends tous les enjeux. Ce monde me fascine et je sens qu'il me permettra de mettre en adéquation mon sens pragmatique de la réalisation, du business et de la créativité.

Grâce à une opportunité je décide de reprendre mes études et contacte M2i pour leur soumettre mon projet. Nous décidons que la certification de concepteur développeur serait parfaite pour mon profil. Au cours de cette formation je rentre en contact avec M. Titouan BENOIT le CTO de la société Welp, dans le cadre de ma recherche de stage pour clore cette formation diplômante.

Lors de cet entretien, il m'offre l'opportunité de développer l'application mobile de la société Welp, sur des technologies que je n'avais pas étudiées lors de ma formation.

En accord avec mon tuteur M. Pascal BUGUET, je décide d'accepter la proposition de stage, et intègre la société pour une période de 2 mois à compter du 7 décembre.

1.2 L'ENTREPRISE

Welp est une société basée à Paris avec un siège social dans le 5ème arrondissement et des bureaux situés dans la pépinière 27, au 27 rue du chemin vert paris 11^e.

A la tête Marie TREPOZ, déjà créatrice de mille et une liste qui imagine un site et une application qui permettraient de créer du lien entre ceux qui ont besoin d'une aide ponctuelle et gratuite et ceux qui sont prêts à aider de temps en temps, près de chez eux et sans s'engager. Welp voit le jour le 16 mars 2015 avec près de 500 inscrits dès le premier mois.

1.2.1 LES MOYENS HUMAINS:

L'équipe de Welp se compose de 6 ETPS :

- **Marie TREPOZ** - Présidente de Welp et cofondatrice

DÉPARTEMENT INNOVATION ET TECHNOLOGIE

Mission : assurer le bon fonctionnement au quotidien de la plateforme et des applications Smartphone, et mettre en place toutes les innovations nécessaires au positionnement de Welp comme la plateforme référence du bénévolat 2.0.

• **Titouan BENOIT** - CTO (Directeur Technique). Responsable de toute la partie technique chez Welp (site internet, hébergement, infogérance, gestion du team de développement, relation avec la partie marketing,).

• **Fares DOGHRI** - Stagiaire développeur actuellement en 3ème année à l'école EPITECH, en stage en alternance chez Welp où il écrit les spécifications des futures applications Smartphone (iOS et Android) avant de participer à leur développement.

• **Cyril STERN** - Stagiaire développeur en formation « concepteur/Développeur » à l'école M2I depuis août 2015, il travaille sur les améliorations du site welp.fr, la mise en place d'API et le redéveloppement des applications smartphone.

DEPARTEMENT MARKETING/ COMMERCIAL & SERVICE CLIENT

Mission : veiller à la bonne utilisation du site et au confort client, négocier les partenariats avec les villes, entreprises et associations partenaires, organiser les événements et lancer toutes les campagnes de communication de Welp.

• **Christophe RENARD** - Stagiaire Marketing / commercial & service client

• **Valentine DEHONT** - Stagiaire Marketing / commercial & service client Valentine s'occupe du service client et du suivi des associations au sein de Welp.

• **Celine EUZEN** - Responsable relations presse freelance. Elle travaille à présent à la renommée de Welp en freelance.



Figure 1 Equipe Welp

Welp c'est d'abord la volonté de mettre en relation des particuliers autour d'une aide ponctuelle et gratuite et ceux qui sont prêts à aider de temps en temps, près de chez eux et sans s'engager. Leur idée est de recréer localement l'entraide informelle qui existait autrefois dans les villages entre les générations.

1.2.2 Chiffres Welp 2015 :

Welp aujourd'hui, c'est :

- + **5 000 inscrits**, 2000 annonces (dont 40% ont reçu une proposition d'aide), **5 200 fans Facebook**, + 50 000 vues sur notre chaîne You tube
- **Un 1er partenariat** signé avec la ville de Sèvres le 2/07/2015
- **20 partenariats en cours de discussion** avec des mairies (Boulogne-Billancourt, Chaville, Ville d'Avray, Meudon, Issy Les Moulineaux, Chatou, Poissy, Elancourt, Mairie de Paris et quelques mairies d'arrondissement, Levallois-Perret, Asnières, Reims, Guilherand, Saint Quentin en Aisne, Nice...)



Figure 2 Chiffre Welp

- **15 partenariats en cours de discussions avec des entreprises (+500 employés) : Danone, la MAIF, Fondation Carrefour, la Poste, L'Oréal, la Macif, Unibail-Rodamco, Intéria, Allianz, michelin, Générali, La Poste...**

Et plus de 400 associations inscrites sur le site Welp.fr (dont Action contre la faim, la croix rouge, le secours populaire, le secours catholique, habitat et humanise, j'accède, petit frère des pauvres, les amis de la maison verte, nos petits frères et sœurs, apprentis d'auteuil, banque alimentaire, action passeraile, espace les monis, autre monde, paralyse de France, camaleon, citizen momes, zebunet, genie sportif)

Tableau prévisionnel de développement.

Echéances	Objectifs qualitatifs	Objectifs quantitatifs	Moyens
Court terme (vous pouvez préciser l'année)	<i>Mettre en place toutes les innovations technologiques nécessaires à la création de nouvelles formules de partenariat et permettant de générer plus d'annonces et d'inscriptions.</i>	<i>8 000 inscrits à décembre 2015</i>	<i>Réalisations de dossiers de subventions et de prêts à taux 0 afin de pouvoir financer des embauches d'ETPs.</i>
Moyen terme	<i>Développement en flux tendu de sites de marque blanche au fur et à mesure de la signature des partenariats avec les villes, entreprises et associations sur toute la France.</i>	<i>50 000 inscrits et 189 958 euros de CA à décembre 2016</i>	<i>Embauche de 6 ETPS : 2 techniques, 2 commerciaux et 2 services clients pour la mise en place des innovations et la négociation des partenariats.</i>
Long terme	<i>Déployer Welp à l'étranger et comprendre les besoins d'entraide (quelles sont les annonces sur lesquelles les français ou espagnols... réagissent le plus ?, etc.)</i>	<i>500 000 inscrits à décembre 2017 et 499 836 euros de CA</i>	<i>Embauche de consultants/salariés de Welp dans les pays cibles afin de favoriser le déploiement de Welp localement.</i>

1.2.3 LA CONCURRENCE

Les principaux sites concurrents de Welp peuvent être répartis en 5 catégories différentes :

Les sites dits à engagement ciblant les particuliers à l'instar de 1mile.com, indigo.world, yakasaider.fr (à engagements car basés sur un principe d'échanges de services). Peu de chiffres disponibles mais aucun n'a percé à date.

Les sites sans engagement ciblant uniquement les associations comme mavillejetaide.fr, benenova.fr, tousbenevoles.org et francebenevolat.org (les 2 principaux et les plus anciens), generationenaction.com

Les sites de proximité où se mêlent services payants et gratuits : ma-residence.fr, mesbonnescopines.com, gensdeconfiance.fr...

Les sites se positionnant comme de l'entraide mais dont les services échangés sont en réalité payants (pas directement concurrents donc) : sefaireaider.com, stootie.com, hamak.fr

Et enfin les sites sans engagement ciblant les particuliers dont font partie Welp et Zewaow.

Zewaow est une application (pas développée sous forme de site) s'adressant à une population beaucoup plus restreinte puisqu'essentiellement mobile et sur le créneau de la rencontre (pas de dimension sociale à la différence de WELP).

Cartographie des concurrents de Welp :



Figure 3 Concurrence Welp

1.2.4 - Abstract

My project was a wonderful success and I am glad and proud to say that Welp is a very good place to make an internship, the working environment is very nice, developers are dynamic and young, directors are sympathetic and they put at your disposal very-performant hardware. I will keep a terrific reminder of this experience.

1.2.5 - remerciements

Je tiens tout d'abord à remercier mon encadrant Pascal BUGUET pour avoir su canaliser ma motivation, m'écouter et surtout pour m'avoir fait confiance. Je tiens à remercier la personne qui a aussi effectué mon encadrement durant ce stage, Titouan BENOIT que je remercie pour m'avoir accueilli et offert une opportunité unique en tant que stagiaire développer : 'la conception d'une application mobile de a-z'.

Un grand merci à Marie TREPOZ qui m'a permis d'intégrer l'équipe Welp.

Je n'oublie pas Christophe pour sa bonne humeur, son enthousiasme et les nombreux plats Picard que nous avons partagé. Merci à l'équipe Welp pour leur accueil convivial et leur aide durant ce stage.

Je tiens également à remercier l'équipe M2I et plus particulièrement Natasha qui a été la première interlocutrice dans ce projet et qui c'est battu pour que je puisse accéder à cette formation.

Un grand merci au DaftPunk qui ont accompagné mes longues heures de codage.

Merci à tous !

ETUDE DE L'ART

1.1 PREAMBULE

La démocratisation d'Internet et son adoption dans la majorité des foyers en fait un outil de communication puissant. Par conséquence, Internet est une excellente vitrine pour les entreprises et les marques associées. Ces dernières proposent des sites Web de plus en plus évolués devenant de véritables applications proposant de la valeur ajoutée (service de réservation ou d'achat en ligne, comparateurs de prix ou de trajets, stockage virtuel de données, ...) pour l'utilisateur et plus seulement des pages d'informations, le temps du web 1.0 est révolu.

Les applications ne sont plus conçues pour un usage prédéterminé, elles ne sont plus distribuées par un canal classique d'achat mais largement diffusées sur internet et enfin elles ne sont plus limitées uniquement aux postes de bureau. Dans une course à la communication, un nouvel outil est apparu sur le marché en 2007, renversant le surf le Smartphone c'est imposé comme élément déterminant avec le développement des app market, pour toute société désireuse de conquérir un panel d'utilisateur toujours plus large. Pour attirer les visiteurs, les applications mobiles doivent évoluer en permanence, savoir se renouveler pour proposer toujours plus de valeur ajoutée. La manière de concevoir ces applications et en particulier leurs Interfaces Homme Machine (IHM), devient une exigence première dans un monde où le zap est un fait institutionnalisé.

Aujourd'hui 55% des Français possèdent un smartphone pour 43% en 2014. Le nombre d'utilisateurs est en constante augmentation. Il est donc utile d'avoir une application fonctionnelle sur mobile.

/En France, les leaders des systèmes d'exploitation mobiles installés sont (+30 millions d'utilisateurs) :

Android (Google) : 63%

iOS (Apple): 21%

Windows. : 9 %

Le développeur va construire son service par composition de services existants et va donc devoir construire de nouveau une IHM pour le nouveau service afin de permettre l'utilisation combinée des différents services composés.

L'objectif du stage est de concevoir et développer une application opérationnelle répondant aux besoins et exigences d'aujourd'hui. Ce projet s'inscrit dans la démarche de la refonte totale d'une application existante mais dont le cahier des charges ne respectait pas les attentes de la société Welp. Le but étant de recréer l'expérience utilisateur qui est développé en interne depuis Août 2015 par le CTO.

Ce projet s'inscrit dans la démarche de la refonte totale d'une application existante mais dont le cahier des charges ne respectait pas les attentes de

la société Welp. Le but étant de recréer l'expérience utilisateur qui est développé en interne depuis Août 2015 par le CTO.

L'application mobile existante à ce jour présente des problèmes d'inconsistante fonctionnels de lenteur et de bug rémanent.

L'application a d'ailleurs été retirée des stores en attendant un retro-fit de celle-ci.

1.2 - LA DEMARCHE GENERALE

La démarche choisi correspond à celle sur une démarche préconisée par Pascal Roques dans son livre « UML2 - Modéliser une application web » publié chez Eyrolles.

C'est celle que nous suivons avec quelques aménagements, dû au choix des technologies et que nous développons une application mobile.

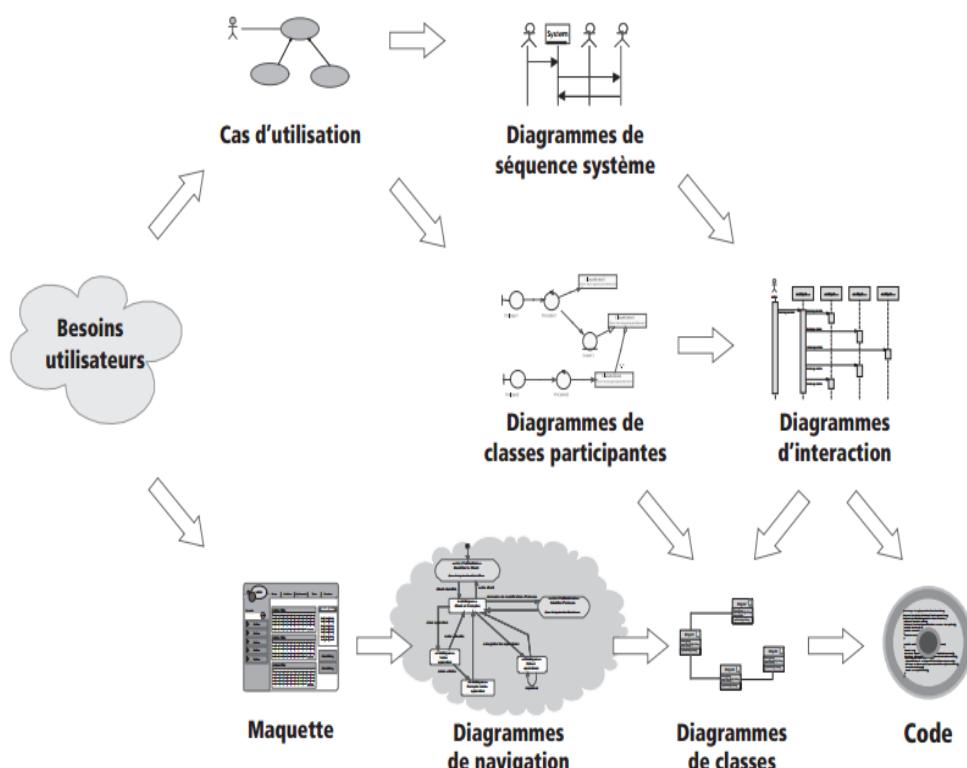


Figure 4 Démarche générale

1.3 LE SITE WEB WELP

Le site a été développé en PHP (*Hypertext Preprocessor*) avec le framework Symfony 2 reposant sur le design pattern MVC, dont la relation entité métier et base de données est entièrement gérée par un ORM Doctrine 2.

1.4 LE FONT END

Tout comme la base de donnée, il était nécessaire de bien comprendre l'IHM développer sur le front end du site web pour pouvoir développer l'application mobile. En voici quelques exemples qui sont détaillés dans l'annexe

1.4.1 Interface graphique du web-site



Figure 5 Home-Page du site Welp

1.4.2 Le menu



Figure 6 menu site Welp

1.4.3 Les cartes



Figure 7 Exemple de design de carte du site welp

Nous avons convenu, que notre priorité était de faire ressortir l'essence même de Welp. Nous voulons rendre l'IHM identique à du SOP (single one page) et épouser au mieux les conventions « matériel design » des différents OS mobile, dans un premier temps en se focalisant sur IOS et Android.

Nous avons pour les différents diagrammes d'activité, nommés aussi au sein de la société « workflow des Users Actions », surveillé à ce qu'ils soient parfaitement identiques entre l'application mobile et l'application web.

CAHIER DES CHARGES

1.1 - PRESENTATION

Lorsque je suis arrivée au sein de l'entreprise welp, le cahier des charges de l'application venait juste d'être entrepris par le second stagiaire, et plus précisément il rédigeait la partie explicative des technologies utilisées.

Je me suis dans un premier temps attelé à la conception des mockups, en imaginant aussi la navigation générale du site. Le travail aboutit à la création des diagrammes de navigation de séquence et du diagramme des cas d'utilisation.

Dès notre première release avec l'équipe welp, orchestrer par notre CTO, nous avons pu valider le diagramme de navigation et après quelques échanges, l'ensemble des IHM.

J'ai crée dans un premier temps les mockups avec Ionic creator qui semblaient être une interface intéressante pour faire l'interface homme machine. Néanmoins même si ce travail persiste sur le cahier des charges, j'ai très vite utilisé photoshop qui s'est révélé plus ergonomique et au final à reproduit un rendu proche de ce que j'avais en tête. J'ai de même utilisé powerpoint pour faire une démonstration virtuelle de l'application au CTO ainsi qu'à la présidente pour formaliser l'avancement du projet ainsi que la validation de la direction prise.

En annexe une version complète du diagramme des cas d'utilisations et le powerpoint de présentation de l'IHM.

En parallèle j'ai du apprendre les différents langages et framework qui nous allions utilisé pour le développement de l'application mobile et que nous n'avions vu ou simplement évoqué au cours de ma formation.

Dès la validation de la partie IHM nous avons commencé le développement de l'application mobile.

1.2 - TECHNOLOGIES IMPOSEES :

Pour le développement de l'application mobile Welp nous avons choisi de créer une application dite "hybride". Une application hybride est basée sur un ensemble de langages communs entre toutes les plateformes. Dans notre cas les langages issus du Web : HTML (*hyper text markup language*), CSS (*cascading style sheet*), JavaScript. La WebView représente un puissant élément commun au différent OS mobile présent sur le marché.

La webView rend possible le portage des applications sur l'ensemble des terminaux à partir d'un seul code. Ce qui permet aux sociétés de conception développement logiciel de n'avoir qu'un seul développeur et non pas un développeur par langage natif.

Avantages et inconvénients

Les avantages du développement d'applications hybrides sont :

- Utilisation des fonctionnalités natives.
- Multiplateforme : on a un seul code à développer et on peut réaliser une application sur les différents systèmes mobiles. (Android, iOS, Firefox OS)
- Maintenabilité du code source : il n'y a qu'un seul code à modifier en cas de mise à jour.

Les Inconvénients :

- Mémoire utilisée : +/- 50 MO en plus. (Négligeable)
- Le debugge peut être parfois difficile.
- Performance faible sur les anciens mobiles : iOS < 3gs et Android < 4

Sources :

<http://www.journaldunet.com/ebusiness/mobile/part-de-marche-des-os-mobiles-en-france.shtml>

<http://butterflyeffect.fr/blog/application-mobile-native-ou-hybride-il-faut-choisir/>

http://lentreprise.lexpress.fr/marketing-vente/ebusiness/le-m-commerce-en-2015-en-5-chiffres_1658637.html

Nous avons opté pour le Framework Ionic qui bien plus qu'un simple Framework traditionnel comme JqueryMobile, Onsen UI. Ionic est un Framework full-stack. Il incorpore Angular JS parfait pour faire du SOP et cordova qui fera la liaison entre le code natif nécessaire à faire fonctionner les différents périphériques du device et la webView. Ionic vient aussi avec Sass qui est un préprocesseur CSS.



Figure 8 Logo Ionic

Lors de ma formation, j'ai travaillé sur des IDE puissant comme Netbean ou Android Studio, mais qui se sont avérés inefficaces sur le projet du développement de l'application mobile.

J'ai pris le pas d'utiliser l'éditeur de code Atom. Très léger et étant open-source, il offre une bibliothèque de plug-in importante et qui augmente la productivité du développeur de façon significative.

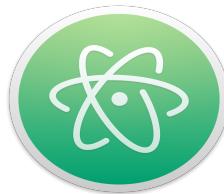


Figure 9 Logo Atom

Le défi, pour un stagiaire, sur un projet avec du code interprété était le test du code en lui même. Le code interprété ne compile pas comme du java. Et le debugge peut se montrer fastidieux.

J'ai utilisé Gulp qui est un automatiseur de taches, et bien que gourmand en ram, il dispose d'un correcteur syntaxique et il permet de compiler les ressources pour une utilisation en temps réel sur navigateur.

Pour la virtualisation de l'application nous nous sommes servi de l'outil Ionic-Labs. Elle permet la virtualisation en parallèle du rendu de l'application sur IOS et ANDROID.



Figure 10 Logo Gulp



Figure 11 Logo Ionic Lab

Je me suis aussi appuyé sur les outils de développement offerts par chrome, nécessaire au debugge d'exécution.

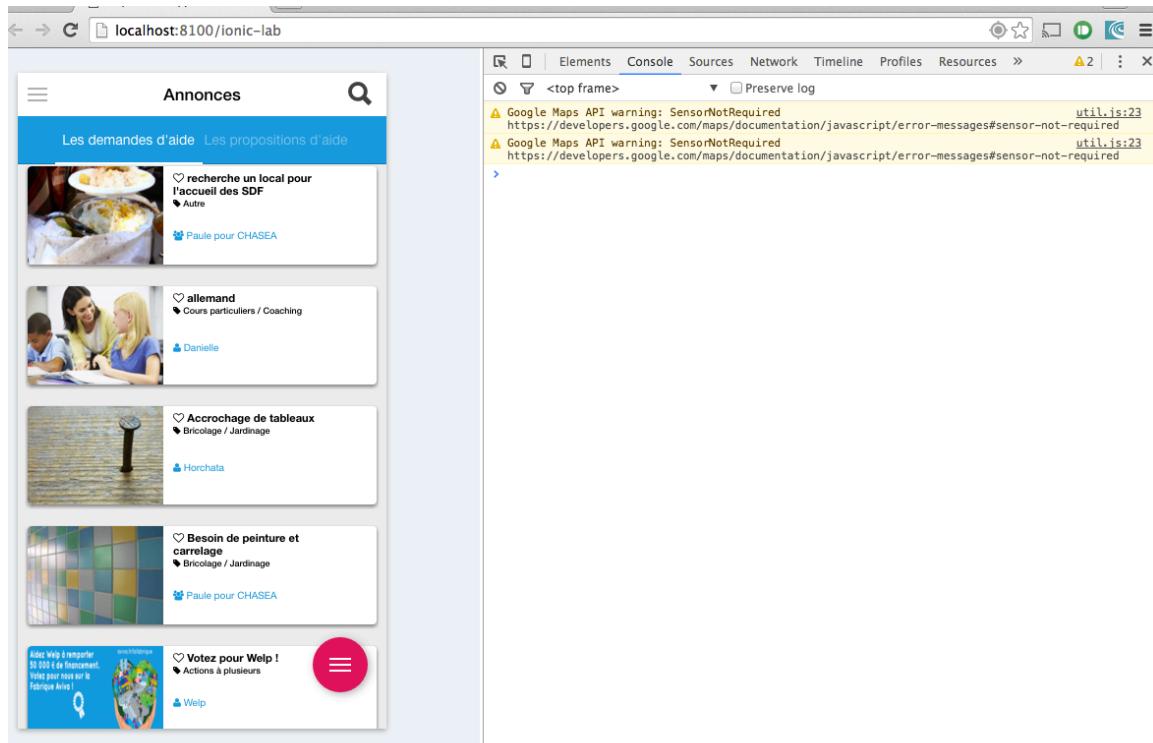


Figure 12 Présentation de Ionic Lab avec outil de développement du navigateur chrome

Nous avons pris soin de partir sur un scaffolding commun entre les développeurs afin de minimiser les erreurs de ‘merge’ sur le logiciel de versionning, mais aussi afin de respecter la structure de nécessaire à l’automatiseur Gulp. Le scaffolding se divise en components, ressources, script, et templates.

Pour faciliter l'intégration de dépendances au sein de notre projet, nous nous sommes appuyés sur deux gestionnaires de dépendances qui sont Bower et npm , en définissant des hooks pour améliorer l'intégration de ses dépendances au sein de la librairie des dépendances.

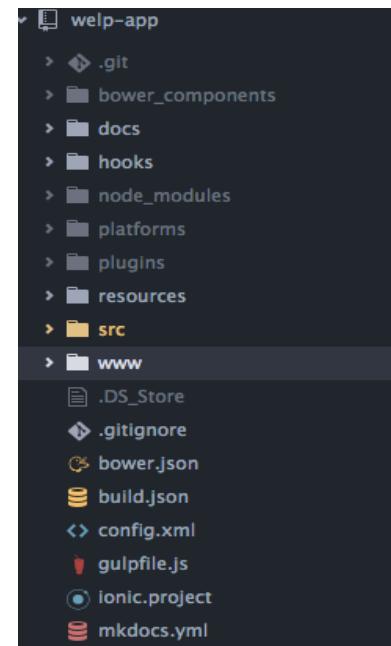


Figure 13 Arborescence non développer du projet



Figure 15 Logo Bower



Figure 14 Logo npm

Le point décisif lors de l'écriture d'un code, c'est sa compréhension au sein d'une équipe Dev et sa maintenabilité.

Dans ce sens nous avons codé la partie logique en CoffeeScript.

Le CoffeeScript est un langage de programmation type Ruby, et qui se compile en JavaScript. Il offre une meilleure lisibilité du code tout en augmentant la brièveté du code. C'est un code indenté.

Le CoffeeScript a permis à l'équipe de rebondir à chaque problème rencontré par l'un des développeurs.



Figure 16 Logo CoffeeScript



Figure 17 Logo Angular JS

En utilisant Ionic nous avons construit notre code avec le Framework Angular Js. Ce Framework a pour but d'aider le développeur à concevoir et organiser son code JavaScript comme une vraie application dédiée à l'UI, en implémentant des patterns dérivés du MVC (Modèle - Vue - Contrôleur) et en fournissant des boîtes outils riches pour la gestion des appels asynchrones, la mise à jour de l'interface et la gestion des données métiers.

Il se compose de quatre couches dominantes qui sont la vue, les contrôleurs (une vue un contrôleur) des services et des directives. Son architecture se base sur un gestionnaire de routes qui lui confère une navigation SOP. Comme je l'ai évoqué plus haut nous avons déplacé la couche modèle de l'applicatif. Et nous avons développé l'application sur un design pattern propre MVW ou plus exactement DataServices - Vue Controller.

La force du Framework réside dans le data-binding bi directionnel entre la vue et le contrôleur, ce qui facilite grandement le développement.

Sa grande force réside dans la data-binding entre la DOM (document objet model) et le Controller grâce au scope, de lier la partie logique et la vue.

La couche modèle est physiquement détachée de la couche métier sur l'application Welp. Pour la développer nous utiliserons du code PHP avec le Framework Symfony 2 en constituant des entités qui feront appeler à l'ORM Doctrine 2. Doctrine fera le lien entre nos entités et la base de données hébergées sur un serveur MySQL.



Figure 20 Logo Doctrine 2



Figure 18 Logo Symfony



Figure 19 Logo PHP

Nous avons versionné notre code en utilisant Git comme logiciel de versions décentralisé. Et gitLab pour gérer le dépôt de nos versions. Nous avons aussi utilisé le plug-in Ungit qui nous donne à accès à une interface graphique, à la place d'une utilisation de CLI.



Figure 22 Logo GitLab

GitLab



Figure 21 Logo Git

git

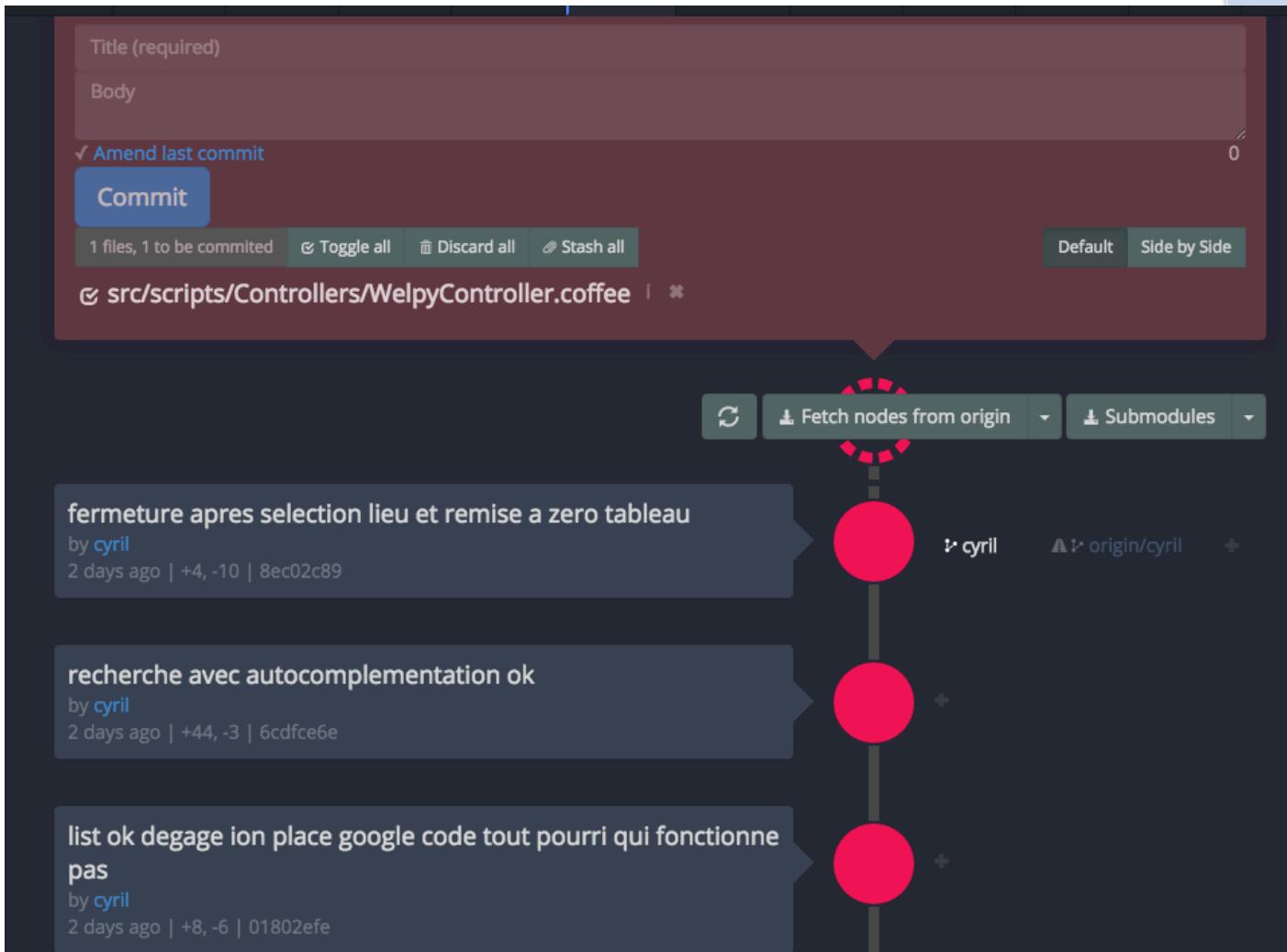


Figure 23 exemple de push de code avec le plug-in Ungit

Tout du long du développement nous avons rédigé la doc de l'application en MarkDoown qui est un langage de balisage et qui offre une syntaxe facile à lire et à écrire.

```
1 # Welp - application mobile
2
3 ## Installation
4
5 To install you will need access to the following projects:
6
7 * http://git.nbcorp.fr/welp/welp-website (backend with REST API)
8 * http://git.nbcorp.fr/welp/welp-app (front)
9
10 Clone these 2 repositories.
11
12 ```` bash
13 git clone git@git.nbcorp.fr:welp/welp-website.git
14 git clone git@git.nbcorp.fr:welp/welp-app.git
15 ````_
16
17 To install backend, follow instructions in README of `welp/welp-website` projects.
18
19 Front development requires these following packages:
20
21 * `npm`_
22 * `ionic`_
23 * `cordova`_
24
25 Assuming `npm` is installed,
26
27 ```` bash
28 sudo npm install -g ionic
29 sudo npm install -g cordova
30 ````_
31
32 ## Git workflow
```

Welp - application mobile

Installation

To install you will need access to the following projects:

- <http://git.nbcorp.fr/welp/welp-website> (backend with REST API)
- <http://git.nbcorp.fr/welp/welp-app> (front)

Clone these 2 repositories.

```
git clone git@git.nbcorp.fr:welp/welp-website.git
git clone git@git.nbcorp.fr:welp/welp-app.git
```

To install backend, follow instructions in README of [welp/welp-website](#) projects.

Front development requires these following packages:

- [npm](#)
- [ionic](#)
- [cordova](#)

Assuming [npm](#) is installed, run this to install ionic :

```
sudo npm install -g ionic
sudo npm install -g cordova
```

Git workflow

Figure 25 Rendu HTML MarkDown



Figure 24 Logo
MarkDown

ANALYSE

1.1 PRESENTATION DE LA METHODOLOGIE UML

Le processus de conception est une étape déterminante dans le portage d'informations entre le client et le développeur. Il va déterminer tout le fonctionnel du système amener à être développé. Lors de notre conception d'application mobile nous avons choisi d'utiliser UML (Unified Modeling Language) ainsi qu'une version plus graphique pour que les documents puissent être partagés au sein de l'entreprise, facilitant l'interaction des non-initiés dans le processus car ils représentent une source d'information importante. Je prends l'exemple d'un diagramme de séquence représentant le workflow d'un welp action, partagé avec la partie commerciale de l'entreprise en lien direct avec les utilisateurs.

Ce partage nous a permis d'opposer notre vision théorique et celle plus pratique des Users. Cet approche couplée à un nombre important de release, nous a permis d'édifier notre IHM rapidement, sans pour autant revenir dessus pour le corriger.

1.2 - LES DIAGRAMMES DE CAS D'UTILISATION

1.2.1 - Représentation graphique

Un **Diagramme de Cas d'Utilisation** représente les fonctionnalités (ou dit cas d'utilisation) nécessaires aux utilisateurs.

Pour notre application, nous devons respecter le DCU du site Welp. A la seule différence que nous n'implémenterons pas de back-office. Après avoir énuméré tous les cas d'utilisation et tous les acteurs, j'ai très rapidement soumis l'idée,

De donnée plus de cas d'utilisation à l'acteur Visiteur. L'acteur Visiteur n'avait à l'origine que les cas d'utilisation login et register. Pour augmenter la qualité de l'IHM, j'ai ajouter à l'acteur Visiteur les cas d'utilisation consulter les propositions d'aide, consulter les demandes d'aide, consulter les événements, accéder au CGU. Je voulais que le visiteur non inscrit puisse accéder aux demandes d'aide et aux propositions d'aide, sans contre-partie.

Le visiteur peut donc consulter les propositions et les demandes d'aide, mais aussi se tenir informer des événements Welp, se loger et s'inscrire.

Le nombre de cas d'utilisation, pour le membre inscrit, s'enrichit. Il peut interagir avec les annonces, welper accepter une aide, chater avec les membres de la communauté Welp. Mais il peut aussi créer des demandes d'aide et des propositions d'aides avec la gestion CRUD (create, read, update, delete) de ses annonces.

DCU Front end

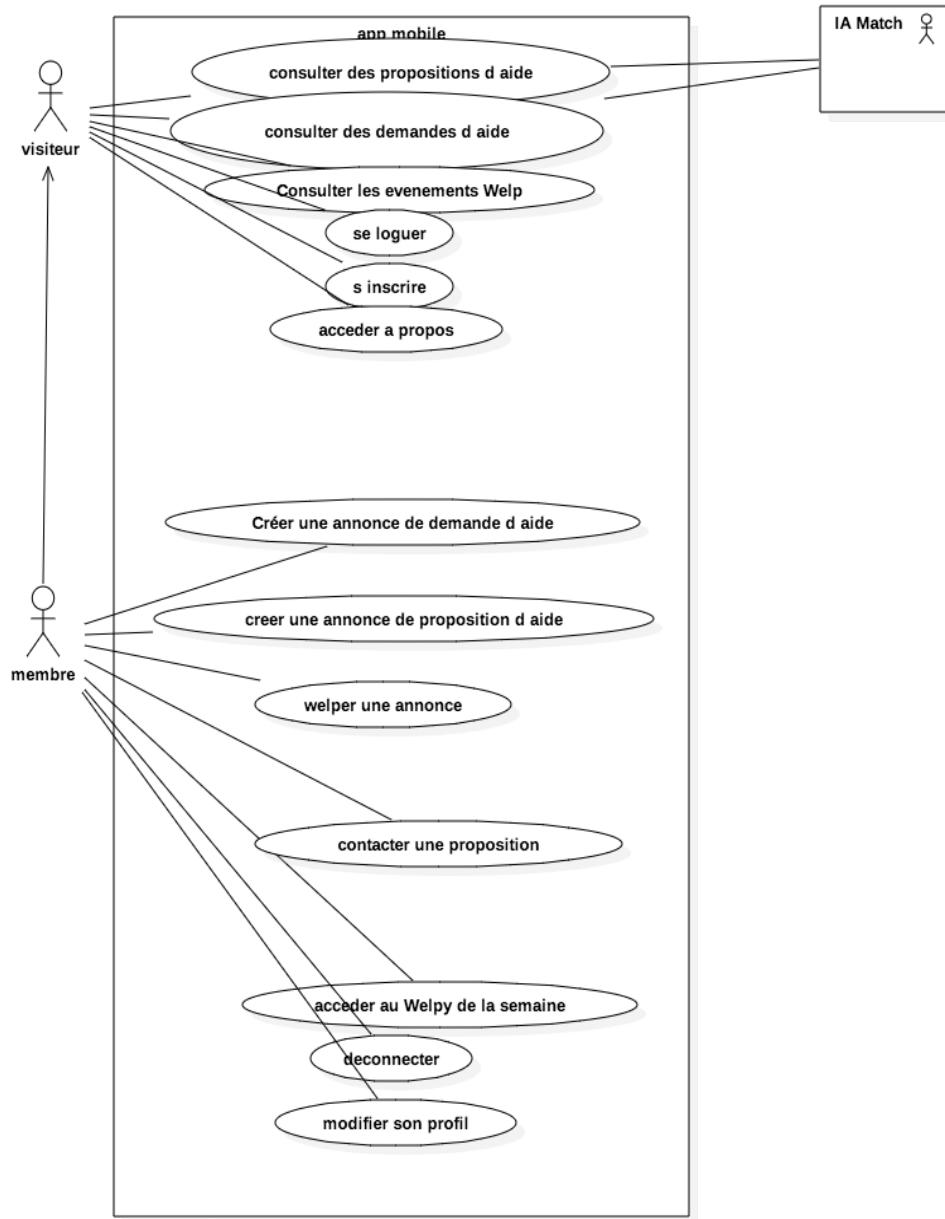
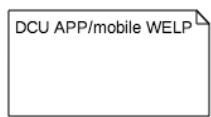


Figure 27 Diagramme des cas d'utilisation de la welp-app

1.2.2- Fiche de description textuelle d'un cas d'utilisation ici le login

Rubriques	Description
Identification Pré-conditions	Visiteur souhaitant s'identifier Remplir le champ input du username et password et submit
Scenario nominal	If username == localStorage.username && password == localStorage.password accès à l'application dans sa totalité.
Scenarii d'erreurs du cas nominal	If username!= localStorage.username && password!= localStorage.password appel au scénario alternatif.
Scenarii alternatifs	DataService call apirest /username/password if return true write localstorage username && password accès à l'application dans sa totalité.
Scenarii d'erreurs des cas alternatifs	If api return false alert vous n'êtes pas inscrits
Post-conditions	null
Exigences non fonctionnelles (Optionnel)	LocalStorage synchrone Dataservice asynchrone

1.2.3- Diagramme d'état de navigations

Premier problème de l'ancienne application : l'interface homme machine. Elle est « inadéquate avec les standards imposés par les différents matériel design » des différents os mobile. Je me suis en grande partie appuyé de la documentation Google pour établir l'architecture de navigation incluant aussi par ce fait les designs de chaque pages.

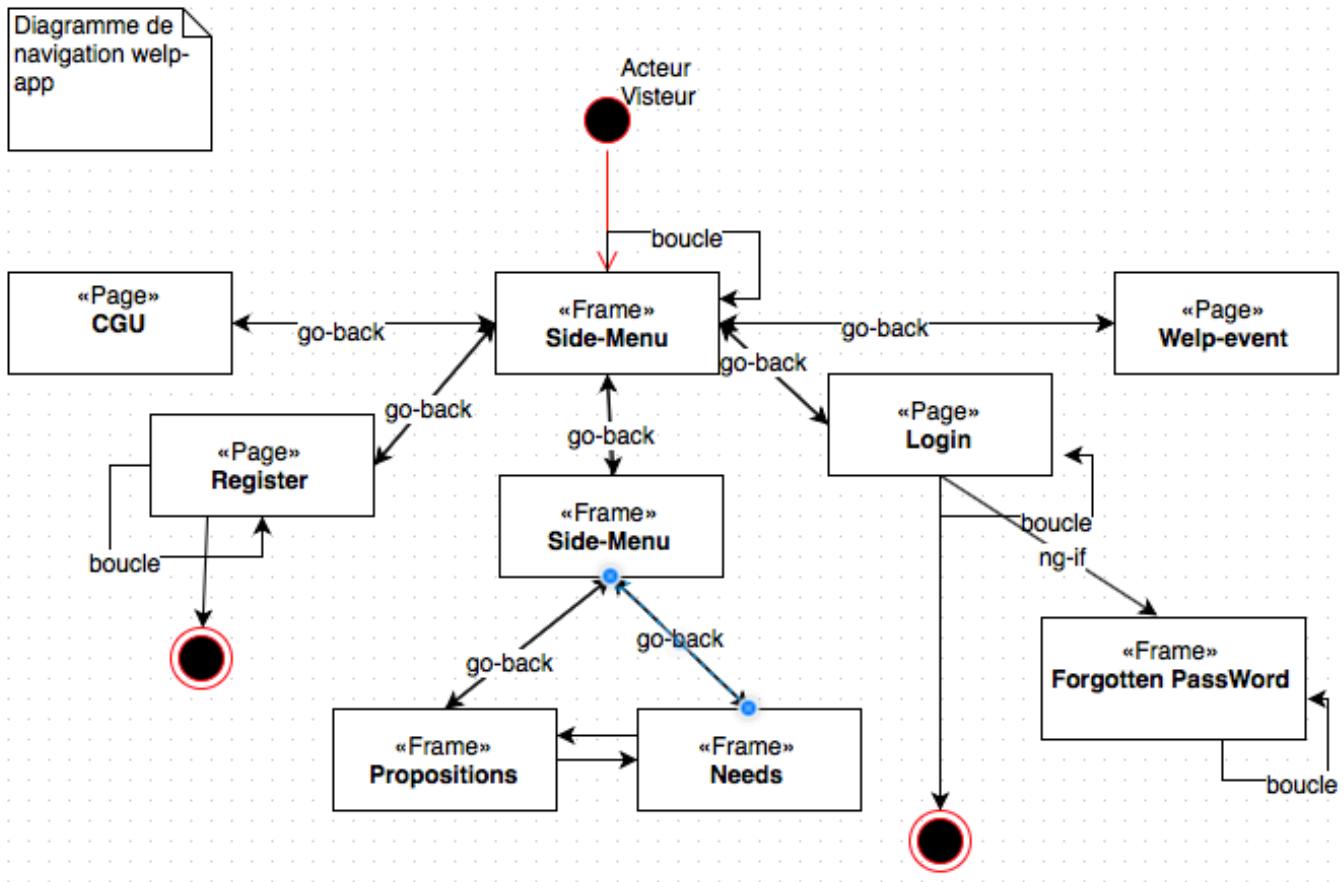


Figure 28 Diagramme de navigation état transitoire

(cf <https://www.google.com/design/spec/material-design/introduction.html>)

UML offre la possibilité de représenter graphiquement l'état de navigation dans l'interface homme-machine en produisant des diagrammes dynamiques qu'on appelle diagrammes de navigation. Le concepteur a le choix d'opter pour cette modélisation entre des diagrammes d'états transitions et des diagrammes d'activités. Puisque nous allons modéliser un comportement événementiel dans le cas d'espèce, nous optons pour les diagrammes d'états de navigation par acteur.

MockUp :

Voici quelques exemplaires de mockup que j'ai réalisé avec photoshop et qui, aussi, mon servit à expliquer le diagramme de navigations.



Figure 30 Icon Welp sur Iphone



Figure 29 SplashScreen Welp

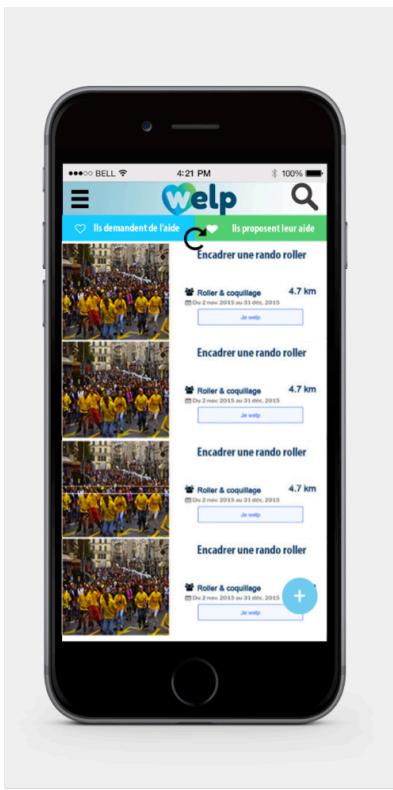


Figure 32 Page Annonce



Figure 33 Page Login

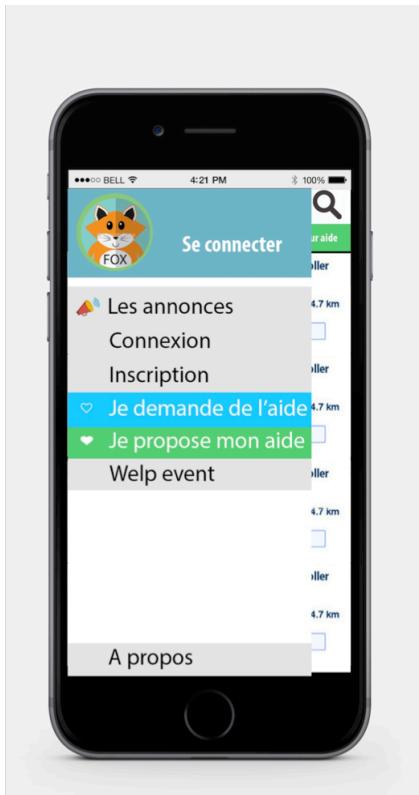


Figure 31 Frame Side Menu visiteur

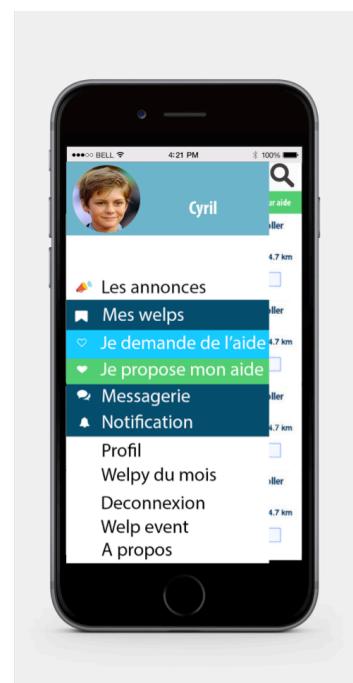


Figure 34 Frame Side Menu User

1.2.4- Les diagrammes de Séquence Système

L'urbanisation du système nous a permis de développer l'application en deux parties. La partie vue-controller toujours dépendante fonctionnellement et la partie service. D'abord rattaché à un seul controller et suivant les conseils du CTO, j'ai très rapidement factorisé ces services à la manière d'un service J2EE pour les rendre accessibles à l'ensemble des controllers et ainsi diminuer le nombre grandissant de fichier. Nous avons élaboré conjointement avec le deuxième stagiaire le squelette de l'IHM ainsi que le rooting Angular et des nominations des templates principaux.

Je me suis ensuite attelé dans un second temps à la réalisation de la partie authentification et inscription de l'application.

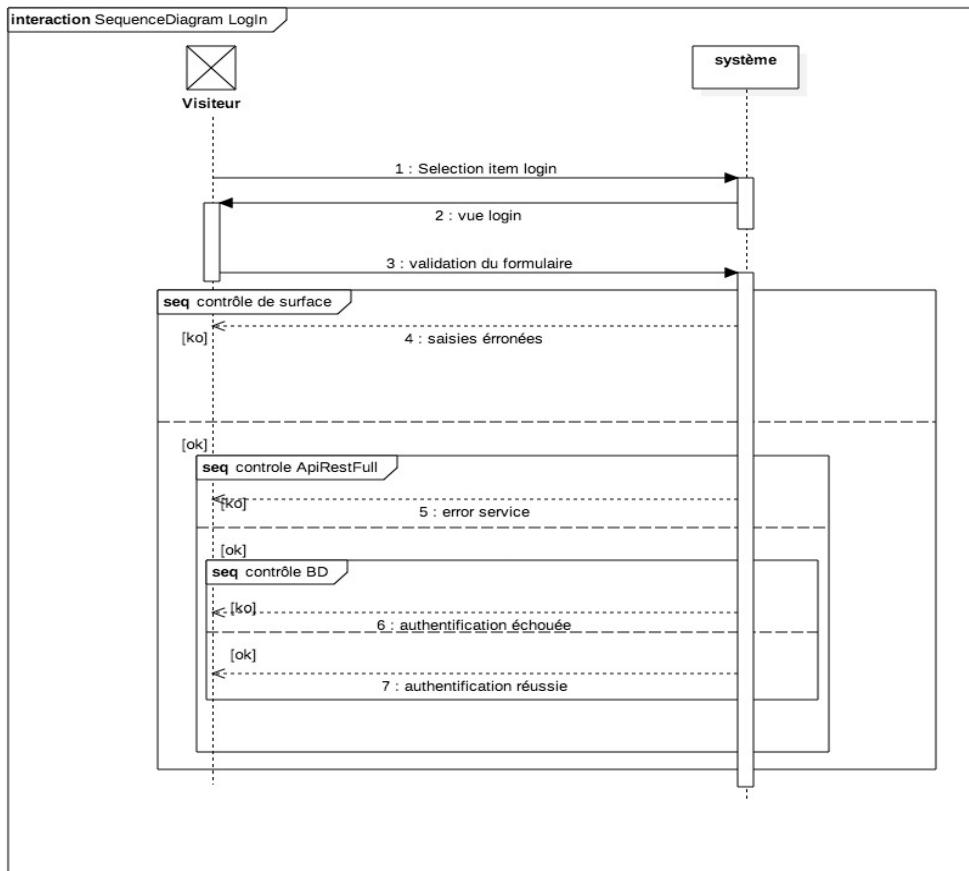


Figure 35 Diagramme de séquence Login

En suivant le diagramme que j'avais développé et que j'ai détaillé en annexe, Je me suis rendu compte de la nécessité que je devais, dans la mesure où l'API Rest ne serait développée que dans un second temps, utiliser le localStorage (qui de toute manière était nécessaire pour la persistance des données user dans le cycle de vie d'un application) et créer un persistance de donnée qui pouvait s'apparenter et se soustraire à l'apiRest.

J'ai donc choisi d'intégrer le plugin file de cordova pour pouvoir écrire un fichier Json des données de l'utilisateur au sein de l'application, puis de développer les services de CRUD (Create Read Update Delete) autour de ce fichier. Cet exercice a permis le test de nos codes. Il nous facilitera aussi la tache lors de la migration vers l'apiRest.

Nous aurons simplement à changer le jsonService par un apiRestService qui communiquera avec le DataService déjà développer.

Dans un deuxième temps nous avons réfléchi pour garder le jsonService comme cache service, afin d'optimiser les performances de notre application.

Ce service créé, et après avoir terminé les services de login et register, j'ai développé le crud des annonces propres au user.

La plus grande difficulté à été le rafraîchissement de la DOM après le saisie des données. Le data-binding d'angular effectuant entre controllers par le rootScope j'ai du utilisé la propagation à travers les scopes enfant à partir d'un événement et du rootScope. Cet exercice m'a permis de comprendre un point clé d'AngularJs et ses subtilités.

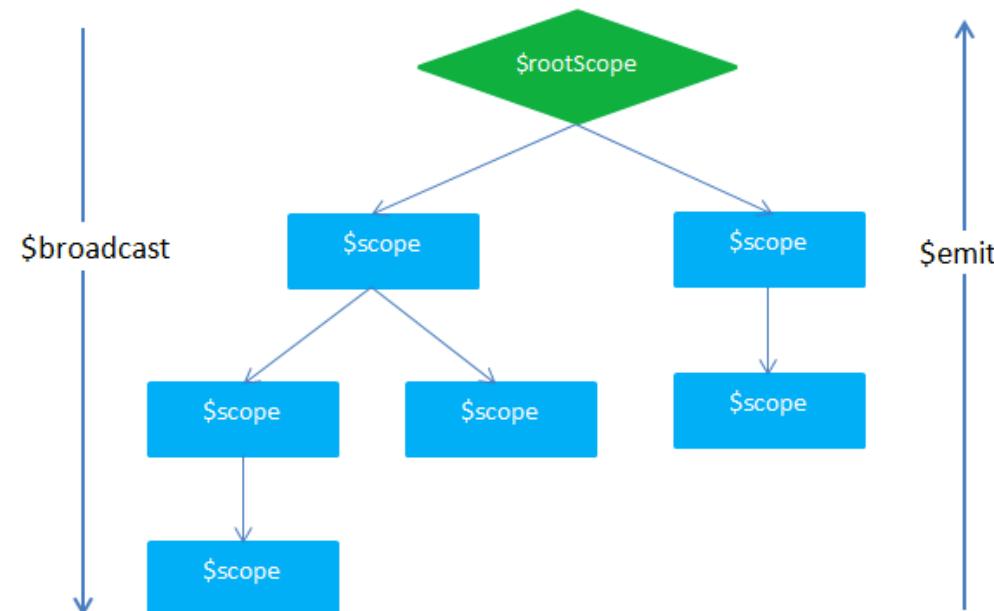


Figure 36 Héritage \$scope

Lors d'un release avec l'équipe, le CTO m'a exposé le business modèle de la société, et ainsi le besoin de concevoir et développer un module répondant à ce business modèle.

Le module consiste à faire choisir aux membres de la communauté Welp une association parmi un carrousel. Chaque sélection (un seul par membre chaque semaine/mois) permet à l'association sélectionnée de percevoir un euros de la part d un mécène démarché par Welp (Welp se rémunérant dix points de la transaction).

J'ai basé ma conception sur un jeu ludique. Le but a été de vraiment faire participer le user. Plus que l'action sur un simple submit avec un carrousel, j'ai orienté mon code sur trois principes fondamentaux du code Javascript, l'event - l'event-listener et le call-back.

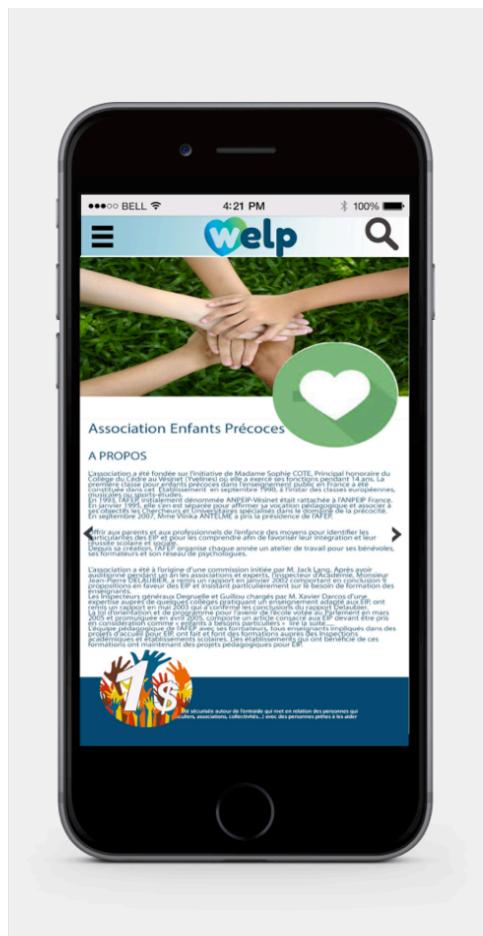
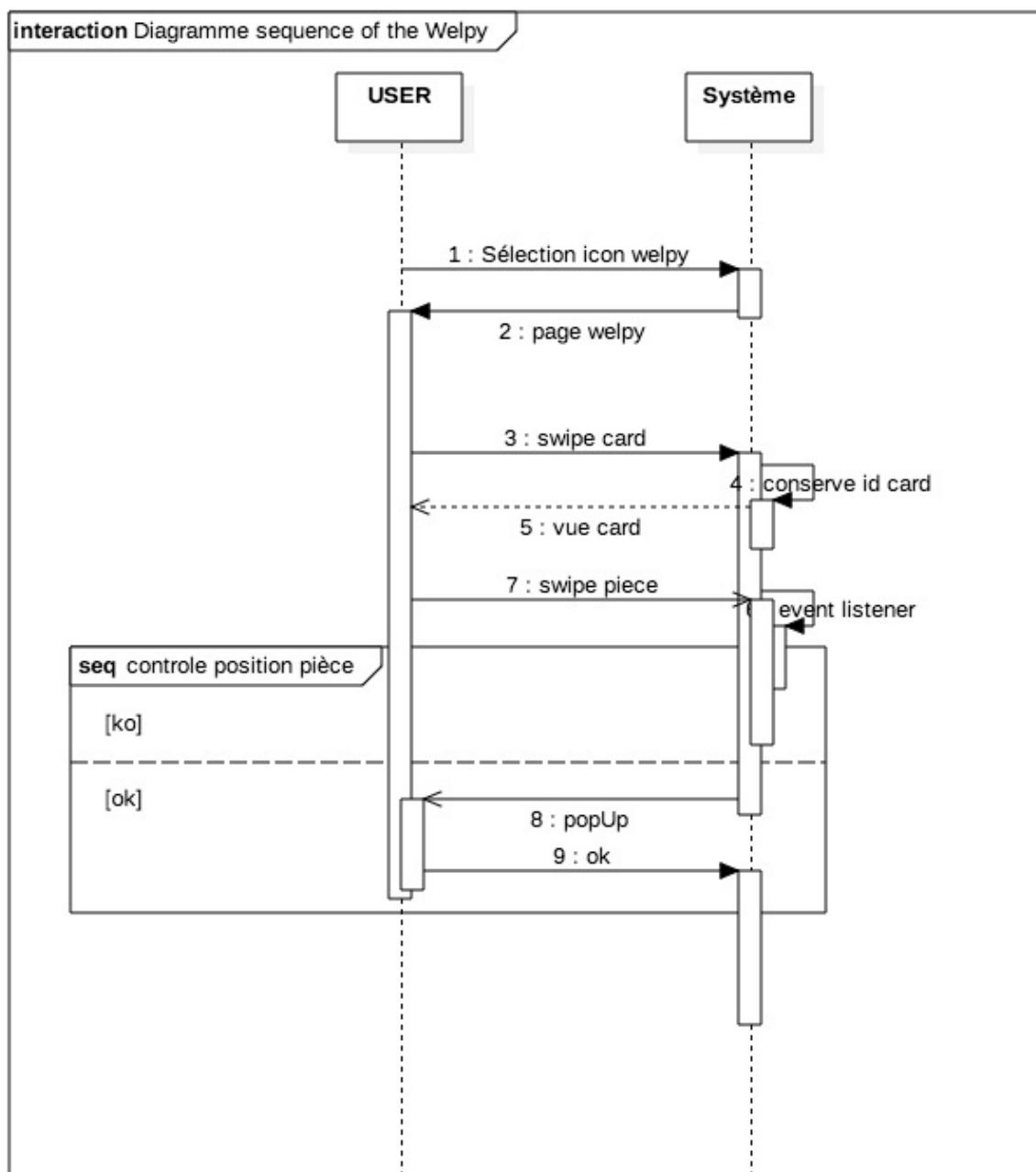


Figure 37 Welpy Game

Ces principes permettent de rendre le web complètement interactif, voire vivant. J'ai donc proposé que sur un carrousel d'association on ajoute un canvas ou une image représentant cet euros symbolique, et qu'elle réponde au drag du doigth sur le téléphone. L'action étant de la faire heatboxer avec un coeur qui grossit. Le drop finalisant à ce moment l'action du jeu, et donc le versement de l'euro à l'association.

L'appréhension du drag drop avec l'expérience du touch game était importante plus que celle du code. Mais les progrès de Cordova ramenant à 300 ms le temps de réponse d'une webView, nous à conforté dans notre choix. La sensation est semblablement identique à un code pure natif.

Figure 38 Diagramme de séquence jeu welpy



1.2.5- Diagramme d'activité

Un Diagramme d'Activité permet de représenter le déroulement d'un cas d'utilisation.

L'activité la plus difficile à mettre en œuvre, sur un plan de sa compréhension, a été le workflow d'une annonce welp. Il fait appelle à quasiment toutes les entités de la base de données ainsi qu'un wording extrément précis, un wording que j'ai assimilé rapidement pour échanger facilement pendant les releases.

Une annonce à un cycle de vie qui peut varier indéfiniment. Elle peut-être créée, consultée, welpée ou acceptée mise en favoris, désactivée supprimée si aucun welp ou acceptations, archivée. Ce workflow est possible grâce à l'interaction deux acteurs sur la même entité.

(Cf Diagramme d'activité d'une annonce en annexe)

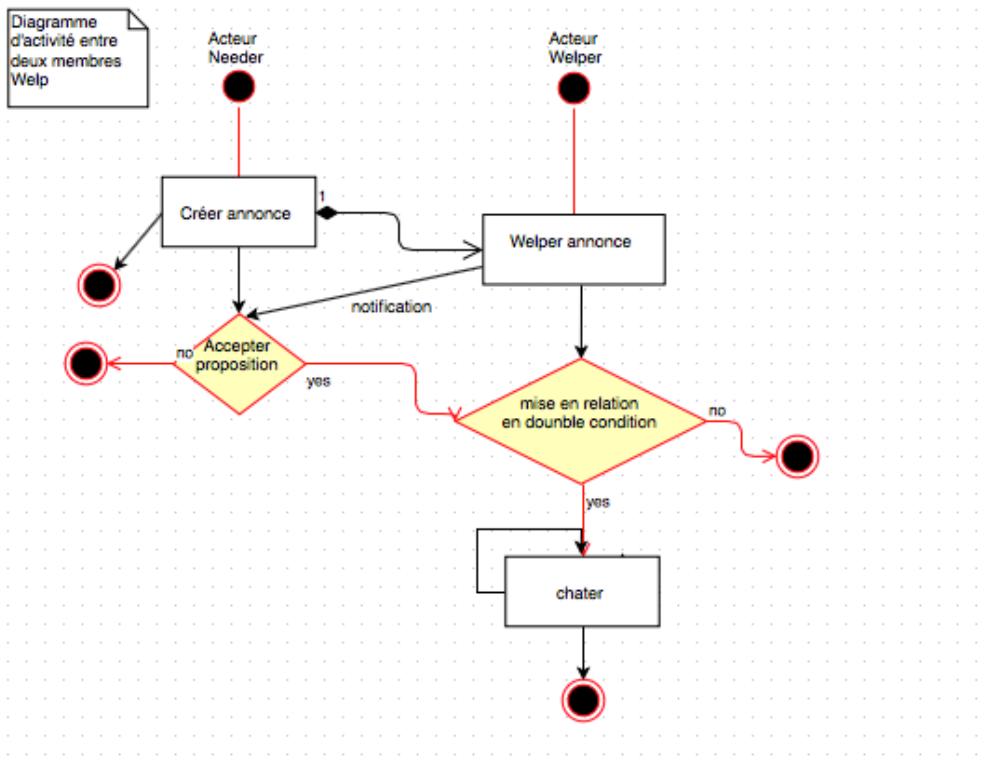


Figure 39 Diagramme d'activité interaction needer welper

Deux points de démarrage, ici, sont représentés, celui acteur needer et celui acteur welper. L'action de créer une annonce crée en héritage la possibilité pour l'acteur welper de sélectionner cette annonce. Les transitions amènent à deux conditions. L'acceptation du welp par le premier acteur, puis la double condition

de vouloir chater qui est une boucle infinie tans que l'un des acteurs ne termine l'action

1.3 - LES DIAGRAMMES DE CLASSE

Le diagramme de classe est une vue statique, car nous ne tenons pas en compte le facteur temporel dans le comportement du système. il modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Execivement utilisé en programmation object, nous avons du l'adapter au langage utilisé qui est plus basé sur des services Angular et du prototypage propre à Javascript.

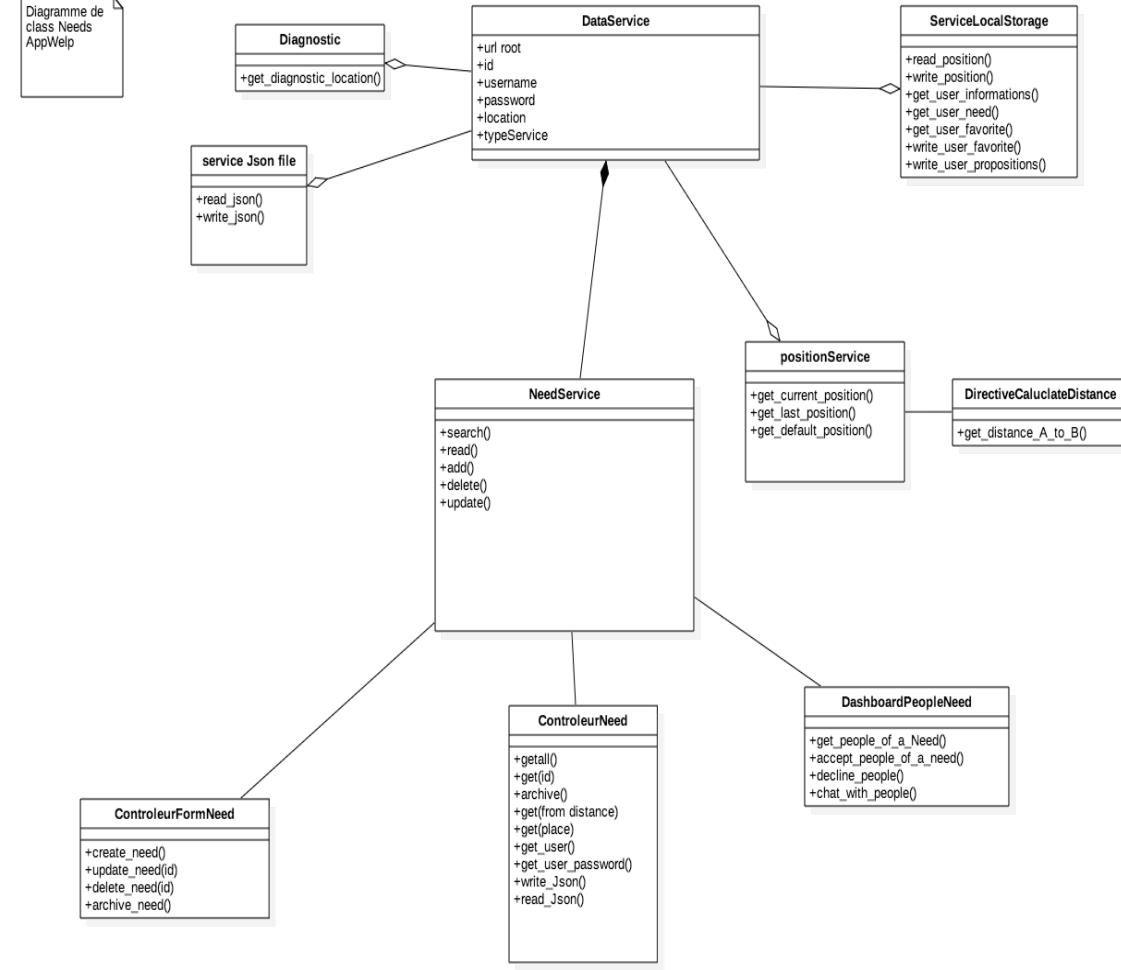


Figure 40 Diagramme de class Need

DiagrammeDeClass
Mobile Device User

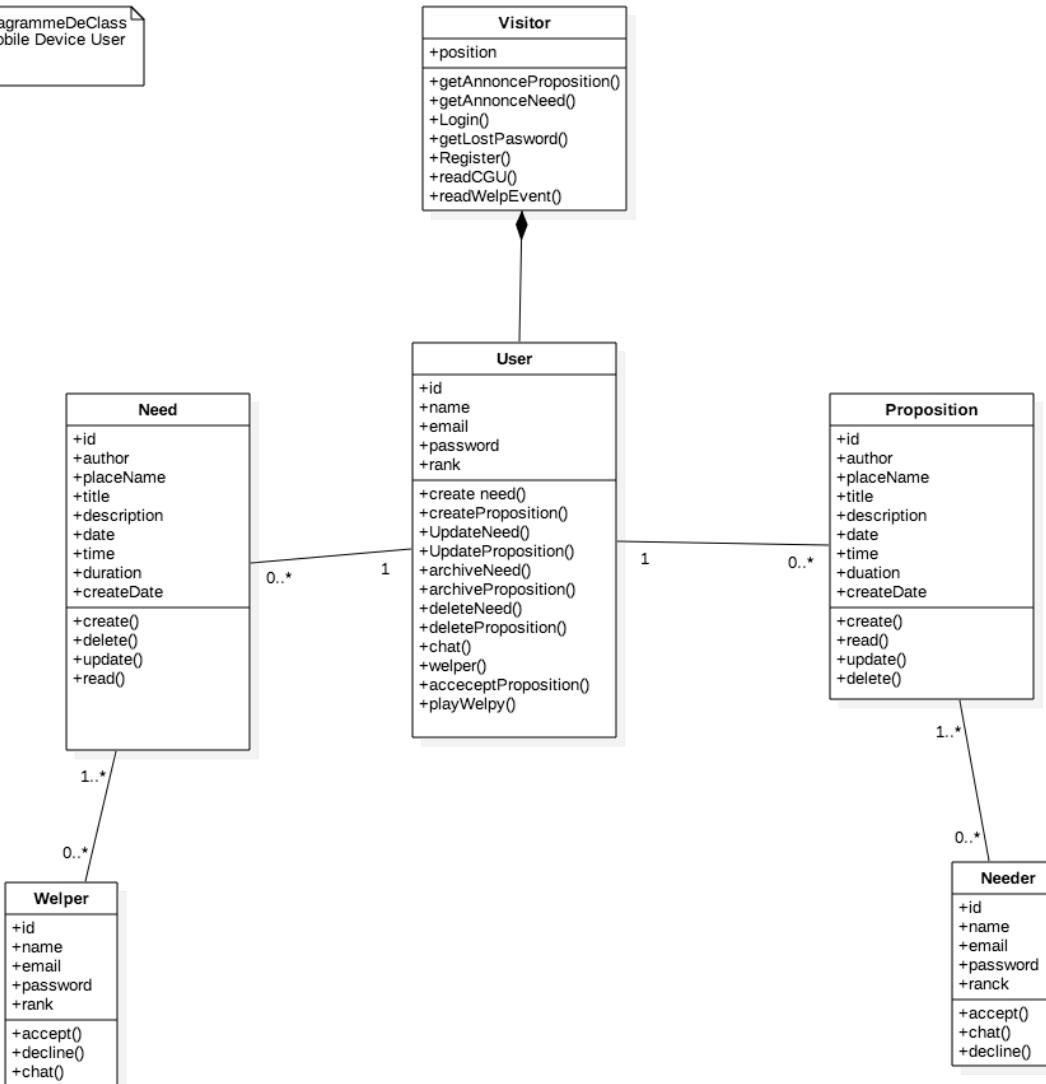


Figure 41 Diagramme de class User

1.5 LA COUCHE MODELE

C'est l'élément majeur de la relation entre l'application développée et l'existant. Contrairement au modèle traditionnel MVC vu et étudié lors de la formation de concepteur au sein de M2I, la couche modèle va être complètement déporté et donc ne fera pas parti de la partie logique embarquée.

La couche modèle va être développé en PHP et sera directement hébergé sur le serveur. Nous utiliserons Doctrine 2 comme ORM entre les classes développées et la base de données.

Enfin nous développerons une Api REST qui se chargera de faire le lien entre l'application mobile et le modèle.

1.4 - LE MDC (MERISE)

Le Modèle Conceptuel des Données - MCD (ou Modèle entité-association), permet de représenter la structure du système d'information, du point de vue des données, et définit également les dépendances ou relations entre ces différentes données.

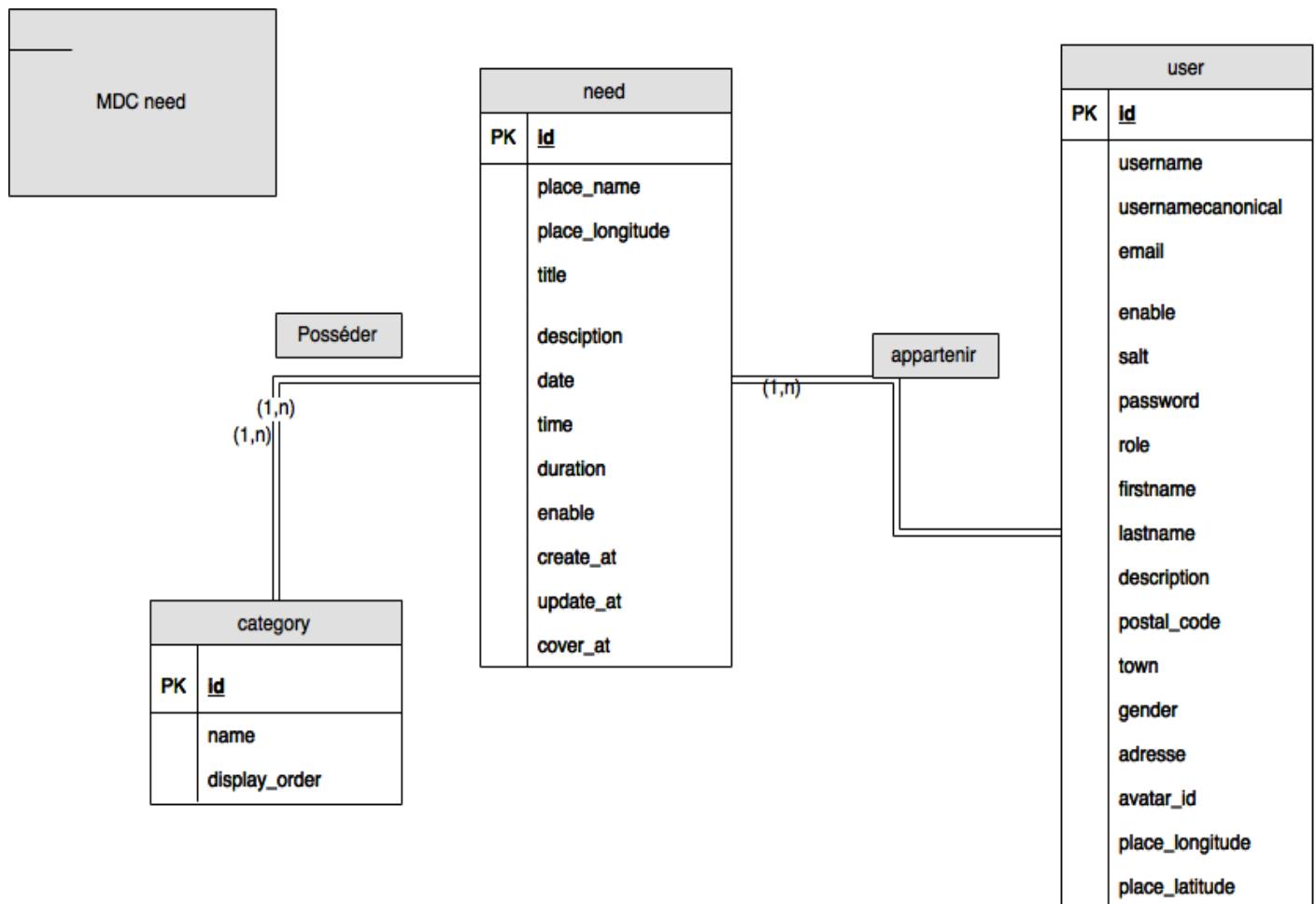


Figure 42 MDC need user

1.5 - LE MLD (MERISE)

Le MLD (modèle logique de donnée), représente la structure de la base de données.

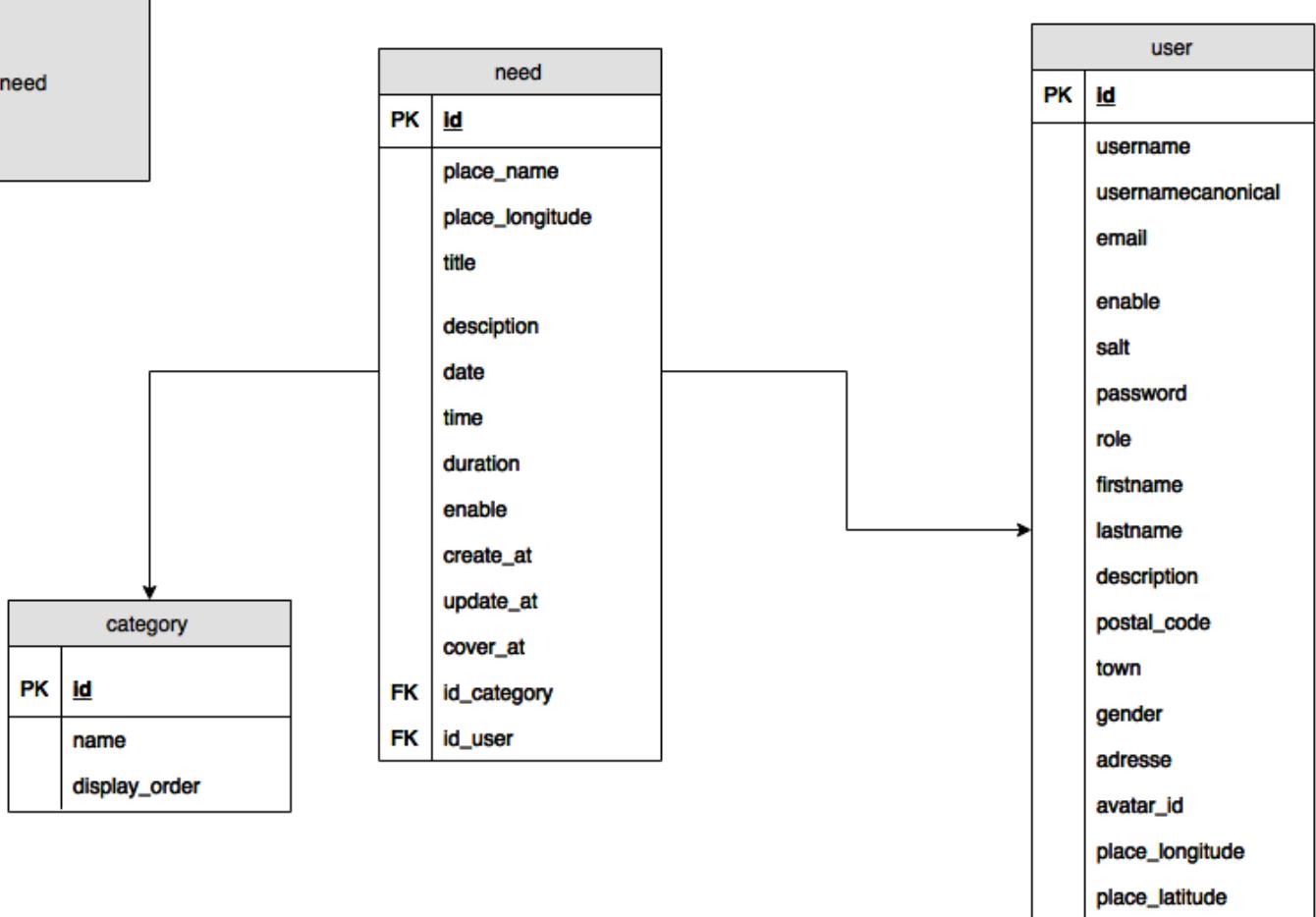


Figure 43 MDC need user

1.5.1 Pourquoi ce modèle ?

L'application est plus simple à entretenir, car elle désolidarise la partie client de la partie serveur. La nature hypermédia de l'application permet d'accéder aux états suivants de l'application par inspection de la ressource courante. Elle permet également de ne pas avoir à maintenir une connexion permanente entre le client et le serveur. Le serveur peut ainsi répondre à d'autres requêtes venant d'autres clients. Cela devient essentiel dans un système distribué.

Dans un contexte App:

L'utilisation du protocole HTTP permet de tirer parti de son enveloppe et ses en-têtes ; l'utilisation d'[URI](#) comme représentant d'une ressource permet d'avoir un système universel d'identification des éléments de l'application ;

Le développement de l'API REST se fera dans un deuxième temps. Nous avons choisi de développer dans une premier temps toute la partie client en développant les différents services qui viendront dialoguer avec l'API REST.

1.5.2 La base de Donnée

La base de données welp est un SGBD-R MySQL.

Durant la première partie de mon stage la priorité a été donnée au front-end et au back-end de l'application mobile. Néanmoins j'ai eu accès au diagramme EER généré par workBench.

Elle nous a aidé dans la conception des services tout comme dans la conception de l'application en général. J'ai pu en extraire dès le démarrage de mon stage que l'utilisateur de l'application welp se retrouve au centre de trois grosses entités qui sont les entités Needs et les Propositions et les

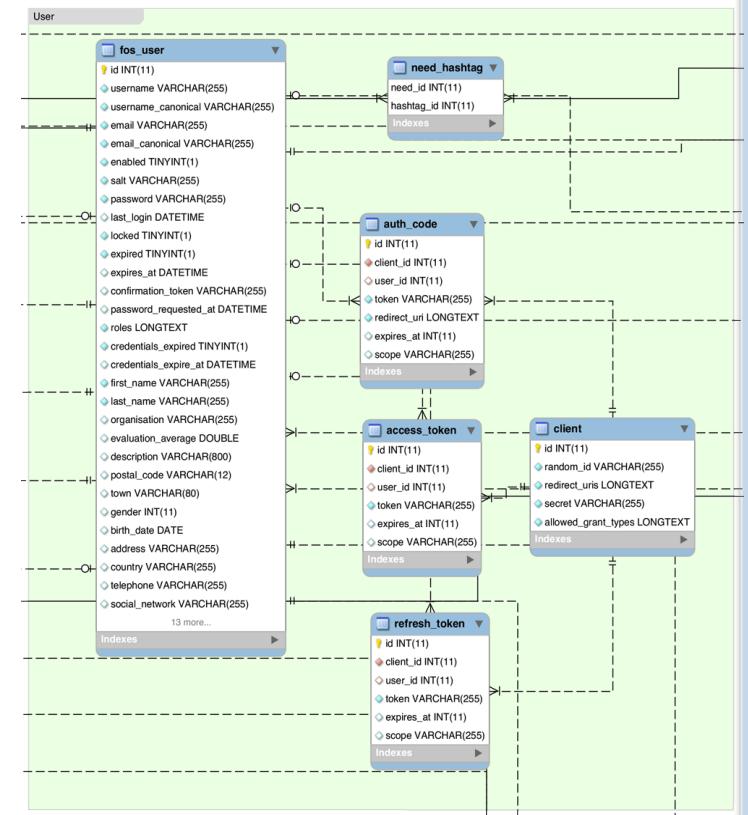


Figure 44 Catégorie regroupant les entités associées à l'entité user

Users, elles regroupent à elles seules la majorité des associations avec les 28 autres entités. J'ai classé l'ensemble de ces entités en six catégories qui sont :

- les Needs
- les Propositions
- les Users
- les associations
- les médias
- les interactions users

Je joins un pdf détaillée de la base de donnée en annexe.
J'ai ensuite étudié les liens de multiplicité entre les entités (cardinalité en meurise du mcd).

J'en ai déduit quelqu'un de ces exemples:

un user peut :

- créer plusieurs annonces de proposition
- plusieurs annonces de demande d'aide
- welper plusieurs annonces
- accepter plusieurs annonces
- noter plusieurs fois
- mais une annonce q'une seul fois

qu'une annonces a

- un seul et unique auteur associées
- une annonce peut avoir plusieurs note.

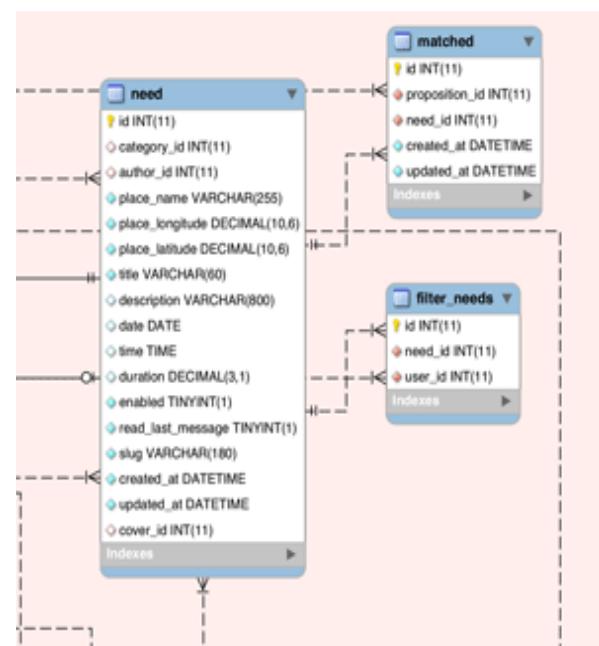


Figure 45 Catégorie regroupant les entités associées à l'entité need

LA CONCEPTION

1.1 - MÉTHODE DE TRAVAIL AGILE

Les méthodes agiles reposent sur une structure (cycle de développement) commune (itératrice, incrémentale et adaptative), quatre valeurs communes déclinées en douze principes communs desquels découlent une base de pratiques, soit communes, soit complémentaires.

Dans le cadre du développement de l'application Welp nous sommes partis sur des itérations d'une semaine. Chaque matinée comportait une réunion d'un quart d'heure avec le CTO, durant laquelle nous expliquions ce que nous développions dans la journée. Nous avons utilisé l'application web trello parfait pour mettre en place cette méthode de travail.

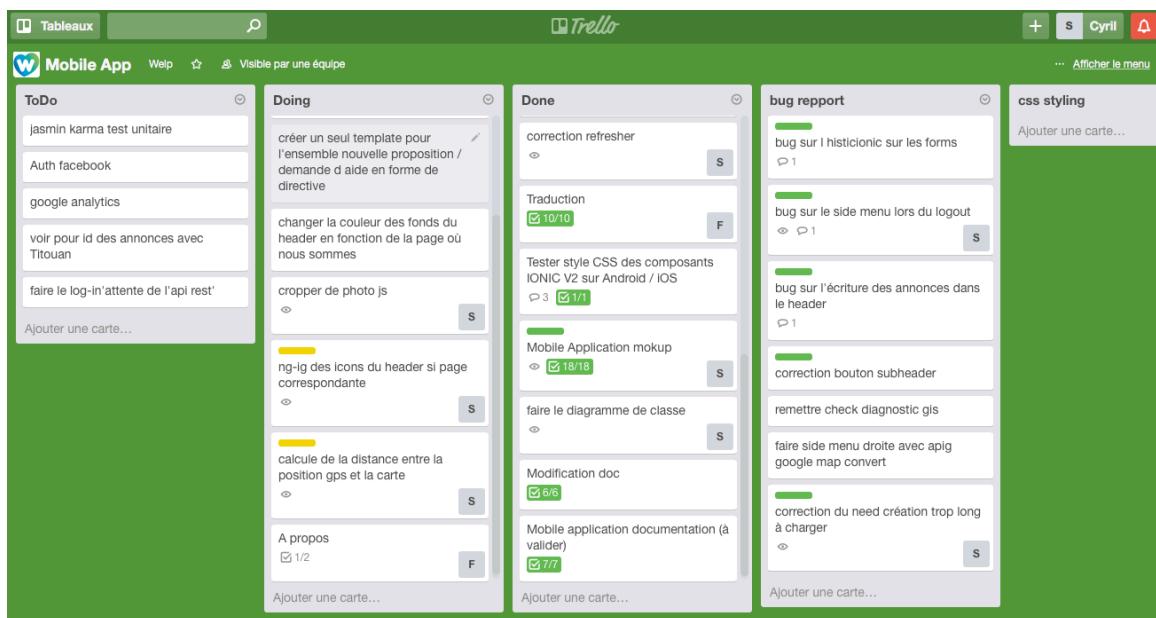


Figure 46 Interface trello pour la gestion du projet

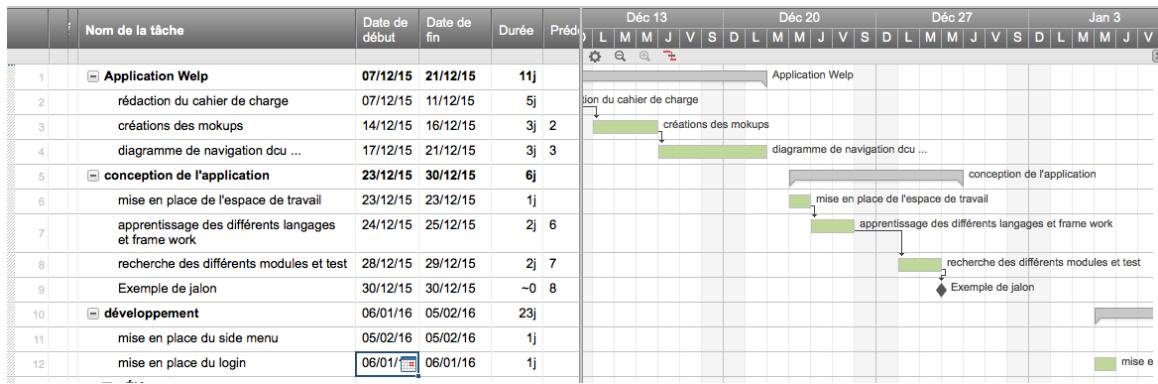


Figure 47 Diagramme de gant du planning prévisionnel de l'application Welp

Afin de maintenir une bonne orientation du projet j'ai réalisé un diagramme de Gant sur le site smarsheets.

Les tâches planifiées sont :

- La création du cahier des charges avec :
 - La recherche des technologies nécessaire au projet
 - Les plug-in nécessaires
- La réalisation des mokups
- La réalisation des différents diagrammes Uml nécessaire au projet
- La release des mokups
- Le développement avec
 - Mise en place de scaffolding
 - Codage des différentes IHM
 - Développement de tous les services
 - Mise en forme CSS de chaque templates
 - Test unitaire du code
- La release de l'avancement de l'application
- Cours magistral sur l' API REST
- Développement de l'API REST
- Migration du dataService
- Déploiement de l'application sur les stores

1.2 - DIAGRAMME DE SEQUENCE DETAILLEE

Le diagramme de séquence est un diagramme qui met en évidence les interactions de toutes les couches (métier) de la boîte noire. C'est un point d'entrée pour le développeur incontournable.

S

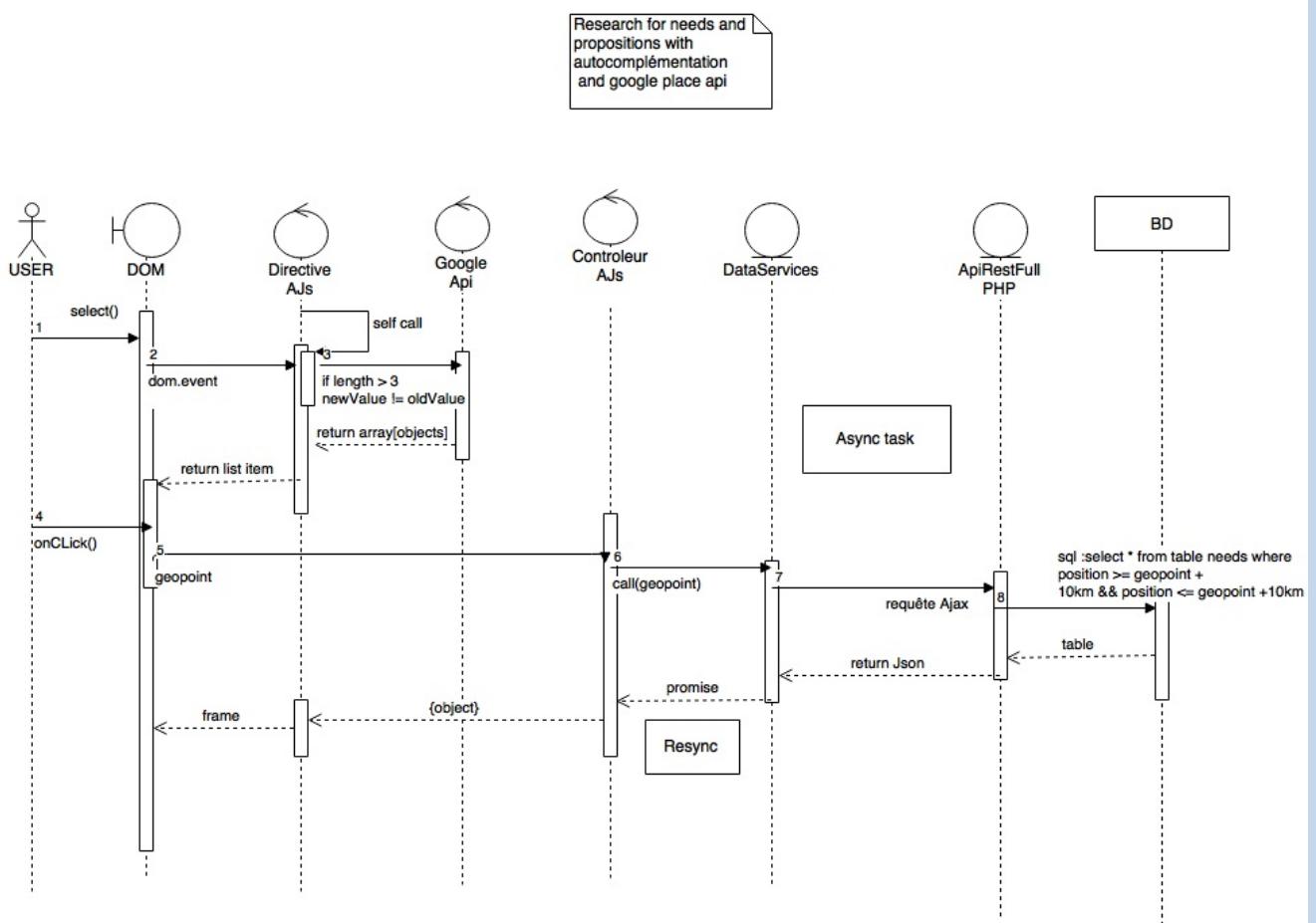


Figure 48 Diagramme de séquence détaillée du cas de l'utilisation recherche annonce suivant adresse

Ici le diagramme expliquant la boîte blanche implémenté dans la directive autocomplete que j ai développé. Elle permet de faire une recherche par adresse des différentes annonces. Elle incorpore aussi l'autocomplémentation de google. Une fois l'adresse sélectionnée la page des annonces se rafraîchit en affichant les annonces proches de la triangulation de l'adresse. C'est exemple servira dans le chapitre développement.

1.3 - DIAGRAMME DE CLASSES PARCIPATIVES

C'est un diagramme de classes où toutes les classes du Diagramme de séquence détaillé sont représentées

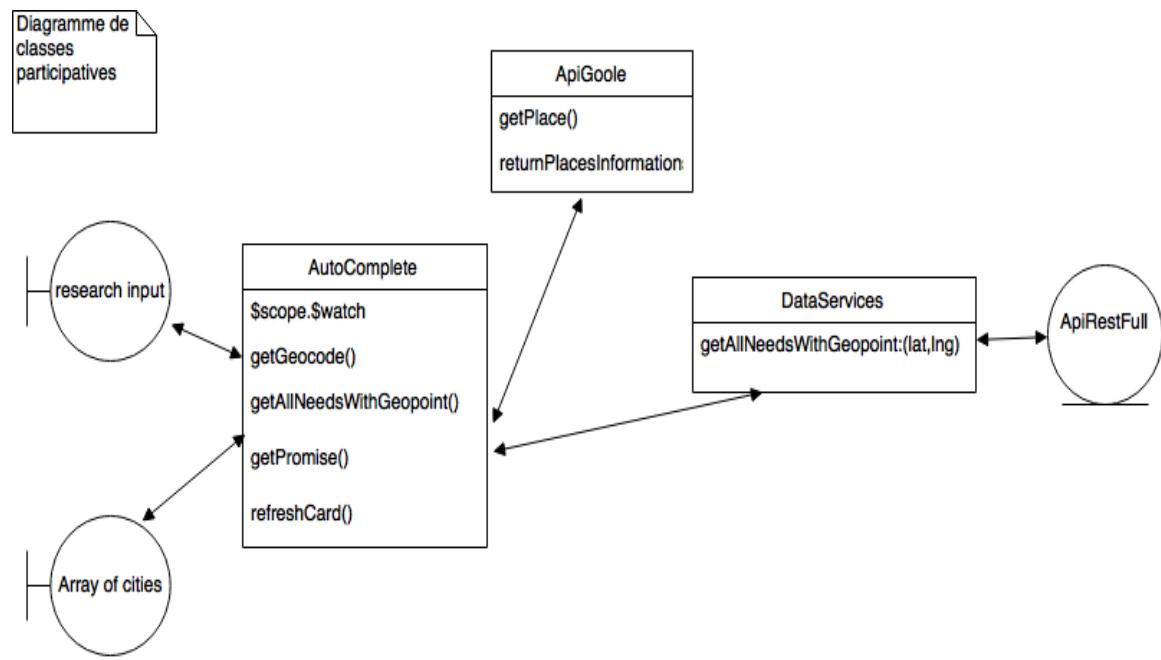


Figure 49 classes participatives du Autocomplète

LE DEVELOPPEMENT

1.1 - LES CATEGORIES DE CODES

1.1.1 Front-end

L'*Hypertext Markup Language*, généralement abrégé **HTML**, est le format de données conçu pour représenter les pages web. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques.



Figure 50 Logo
HTML

Les **feuilles de style en cascade**, généralement appelées **CSS** de l'anglais *Cascading Style Sheets*, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.



Figure 51 Logo
CSS

1.1.2 Back-end

Rappel

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs

permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe.

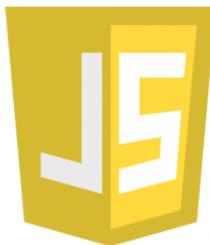


Figure 52 Logo
JavaScript

JavaScript

Ionic est un Framework open source pour développeur des applications mobiles dites hybrides, en se basant sur le moteur de rendu du navigateur embarqué (HTML/CSS/JavaScript). Il est fourni avec les Framework AngularJs et Cordova.

AngularJS est un Framework JavaScript libre et open-source développé par Google. Ce Framework a pour but d'aider le développeur à concevoir et organiser son code JavaScript comme une vraie application dédiée à l'UI, en implémentant des patterns dérivés du MVC (Modèle - Vue - Contrôleur) et en fournissant des boîtes outils riches pour la gestion des appels asynchrones, la mise à jour de l'interface et la gestion des données métiers.

Cordova ou plus anciennement Apache Callback ou PhoneGap, est un framework open-source développé par la Fondation Apache.

PHP: Hypertext Preprocessor3, plus connu sous son sigle **PHP** (acronyme récursif), est un langage de programmation libre4, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP3, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet comme C++. PHP a permis de créer un grand nombre de sites web célèbres, comme Facebook, YouTube, Wikipédia, etc.5 Il est considéré comme la base de la création des sites Internet dits dynamiques.

Ajax (acronyme d'*Asynchronous JavaScript and XML*) est un protocole de communication client serveur asynchrone, d'abord utilisé pour le web, il s'est étendu aux applications desktop et mobile.

1.2 EXEMPLE DE CODE

Pour aller dans la continuité du diagramme de séquence détaillé, un peu plus haut, je vous propose dans ce chapitre de mettre en avant le Framework Angular JS et plus particulièrement des directives.

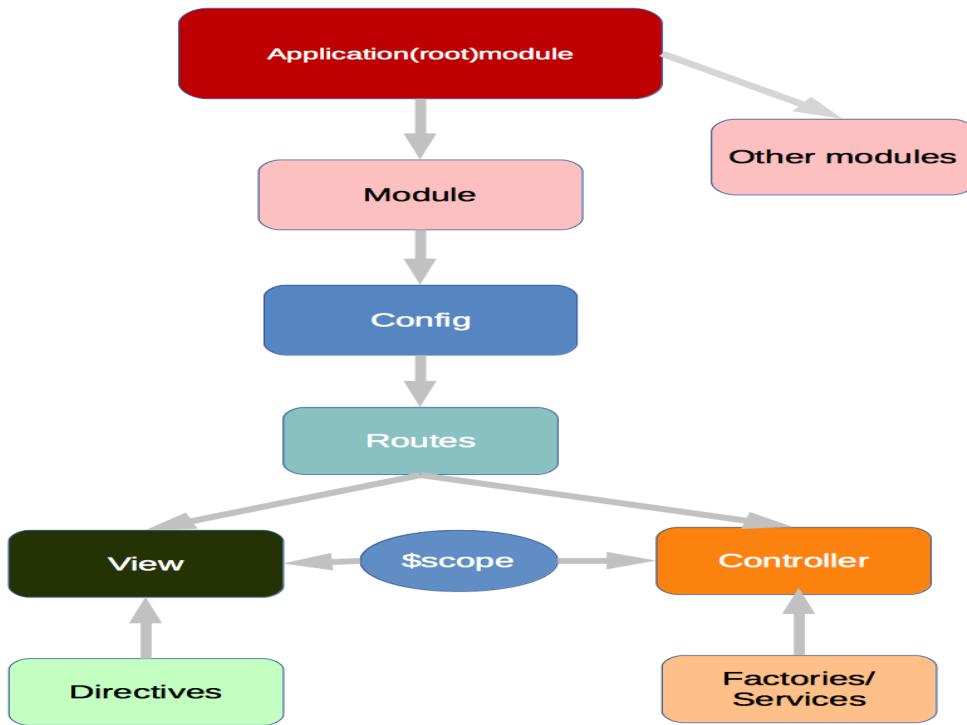


Figure 53 Couches Angular JS

Les directives permettent d'interagir avec la DOM, en injectant des comportants directement dans le code HTML. Angular JS possède déjà ses propres directives (ngModel, ngApp, ngBind...), mais il est possible de créer aussi nos propres directives.

C'est ce que j'ai fait dans le cas du développement de l' autocomplete directive.

Tout d'abord dans le code html j'insère une nouvelle balise (element) que j'ai nommé autocomplete. Cet élément sera le point d'entrée de l'injection de la nouvelle directive

```
....<!-- Right Menu -->
....<ion-side-menu side="right">
....  <ion-header-bar class="header-right bar-stable">
....    <h1 class="title">recherche avancée</h1>
....  </ion-header-bar>
....  <ion-content>
....    ....
....    <autocomplete></autocomplete>
....  </ion-content>
....</ion-side-menu>
....</ion-side-menus>
</ion-view>
```

Figure 54 Code HTML

Pour continuer je dois développer ma directive. Elle devra incorporer le template qui sera injecté dans la DOM et un contrôleur ou linker qui sera la partie logique de cette directive

```

angular.module('welp')
  .directive('autocomplete', [position, $timeout, needsService, $ionicLoading, $rootScope] =>
    {
      restrict: 'E'
      scope:
        latitude: "=?"
        longitude: "=?"

      template: "<input type='text' ng-model='barRecherche'><ion-list>
        <ion-item menu-close ng-repeat='location in locations' type='item-text-wrap' ng-click='selectLocation(location)'>
          {{location.formatted_address}}
        </ion-item></ion-list>"

      controller: ($scope, $timeout) =>
        searchEventTimeout = undefined
        geocoder = new google.maps.Geocoder()

        $scope.selectLocation = (location) =>
          latitude = location.geometry.location.lat()
          longitude = location.geometry.location.lng()
          needsService.getAllNeedsWithGeopoint(latitude, longitude).then((response) =>
            $rootScope.$broadcast 'refreshCard', response
            $ionicLoading.hide()
            $scope.locations = []
          )

        $scope.$watch 'barRecherche', (newValue, oldValue) =>
          if searchEventTimeout
            $timeout.cancel(searchEventTimeout)
            searchEventTimeout = $timeout(() =>
              # if newValue != oldValue
            ), 350)

```

Figure 55 Code de la directive

```

      # return
      if newValue != oldValue && newValue.length > 3

      geocoder.geocode { address: newValue }, (results, status) =>
        if status == google.maps.GeocoderStatus.OK
          $scope.$apply =>
            $scope.locations = results
        else
      ), 350)

```

Le template est injecté dans la DOM, avec une second directive ngRepeat qui est bindée directement au data result de la recherche sur l'API Google.

Avec la directive ngClick, on appelle la fonction getAllNeedsWithGeopoint() du service needService.

```
getAllNeedsWithGeopoint:(lat,lng) =>
  ...
  deferred = $q.defer();

  ...
  $ionicLoading.show template: 'chargement...'

  ...
  $http(
    ...
    headers: 'X-Requested-With': 'XMLHttpRequest'
    ...
    method: 'GET'

    ...
    url: 'https://www.welp.fr/map/search?app_search%5Blimit%5D=20&app_search%5Bneed%5D=true&app_sea

    ...
    #params: 'limit=10, sort_by=created:desc'

    ...
  )
  .success((response, status, headers, config) =>
    ...
    deferred.resolve(response)
  )
  .error =>
    ...
    deferred.reject message: 'impossible de se connecter'

  ...
}

return deferred.promise
```

Figure 56 Code du needService

Le service envoie une requête directement au serveur qui lui retourne un fichier JSON. Il le convertit en objet manipulable dans le code. Il ne reste plus qu'à binder dans la DOM.

DEPLOIEMENT

1.1 LES SOLUTIONS VIRTUELLES

1.1.1 Gulp-Ionic-Chrome

Le déploiement de l'application c'est faite principalement avec Gulp qui est un task manager. Il compile le projet, compile le CoffeScript en javascript, compile le saas, minifie le css, concatene et minifie le JS , injecte les sources qu'il aura compressés directement dans les fichiers src www

Cordova add plateform android/ ionic plateform add android

Cette CLI nous permets d'ajouter le sdk (ici android mais pourrait être aussi IOS) au projet. Le SDK de l'OS est prérequis.

Cordova build

Cette CLI nous permets de builder l'application.

Pour la virtualisation de l 'IHM nous avons utilisé ionic serve et ionic serve lab.

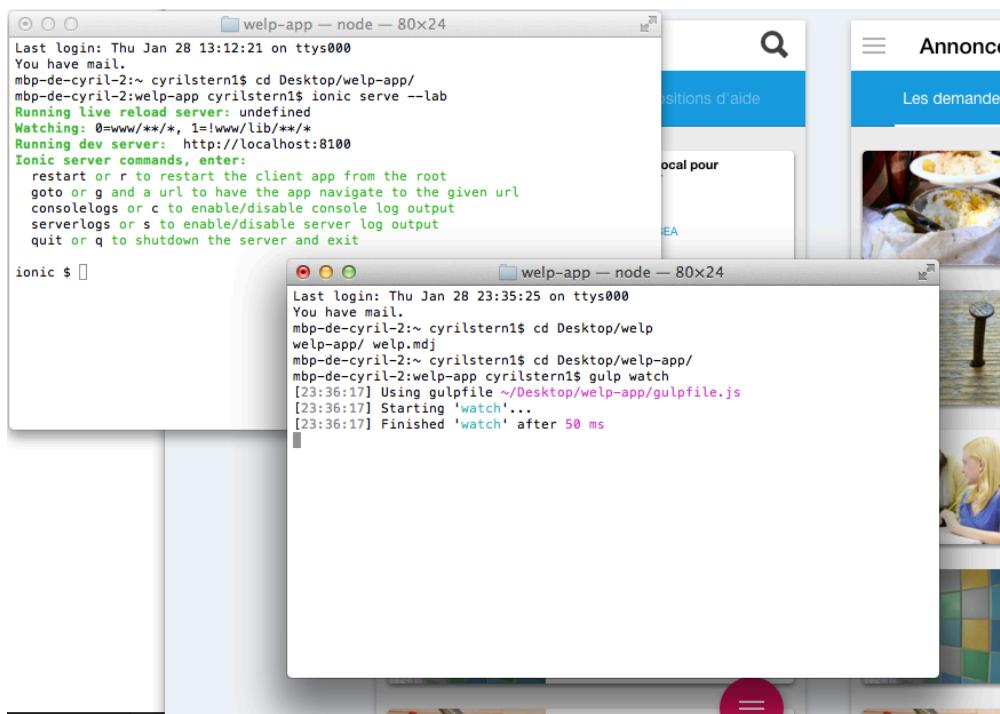
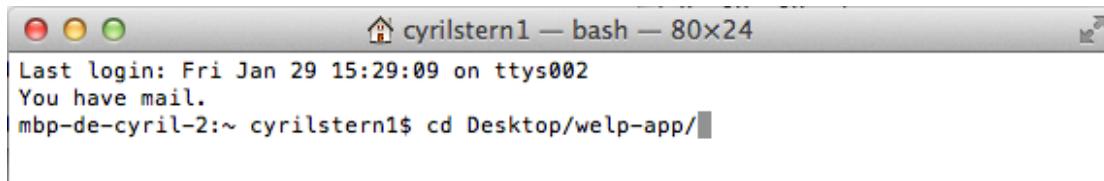


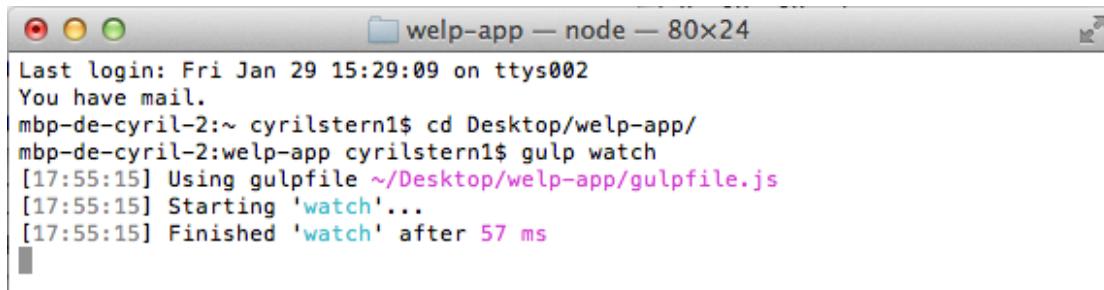
Figure 57 CLI de déploiement

Guide de déploiement sur device virtuel



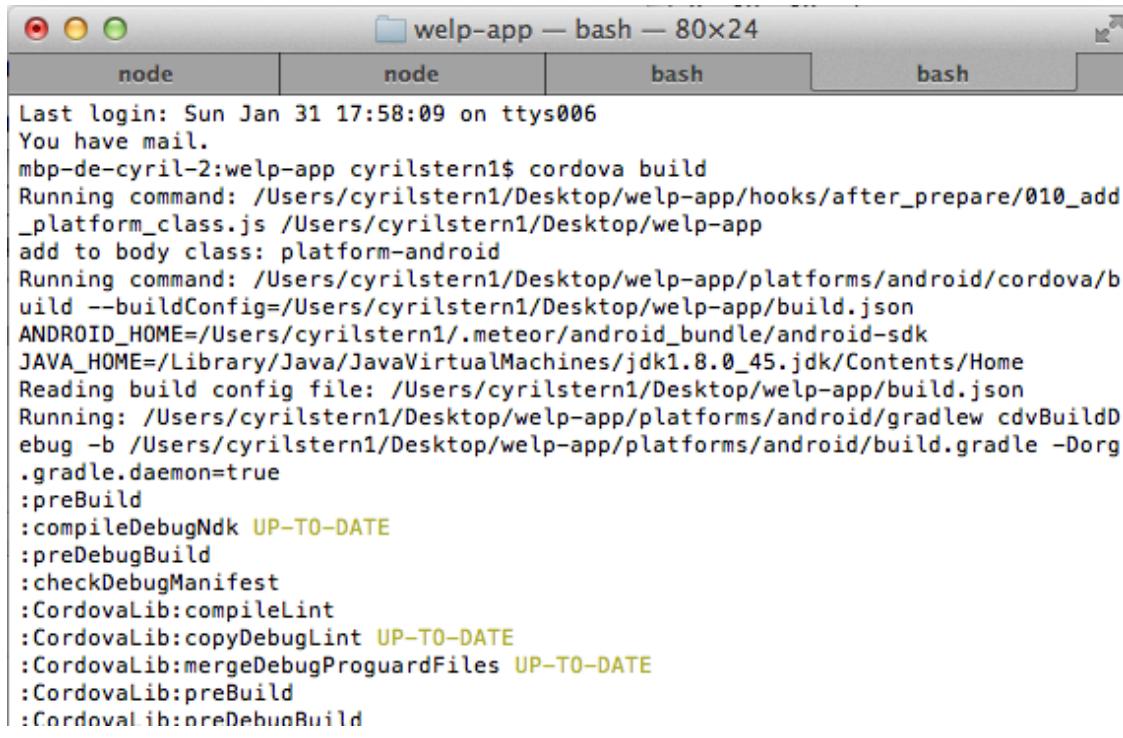
```
cyrilstern1 — bash — 80x24
Last login: Fri Jan 29 15:29:09 on ttys002
You have mail.
mbp-de-cyril-2:~ cyrilstern1$ cd Desktop/welp-app/
```

Figure 59 accès au fichier projet



```
welp-app — node — 80x24
Last login: Fri Jan 29 15:29:09 on ttys002
You have mail.
mbp-de-cyril-2:~ cyrilstern1$ cd Desktop/welp-app/
mbp-de-cyril-2:welp-app cyrilstern1$ gulp watch
[17:55:15] Using gulpfile ~/Desktop/welp-app/gulpfile.js
[17:55:15] Starting 'watch'...
[17:55:15] Finished 'watch' after 57 ms
```

Figure 60 lancement du task manager



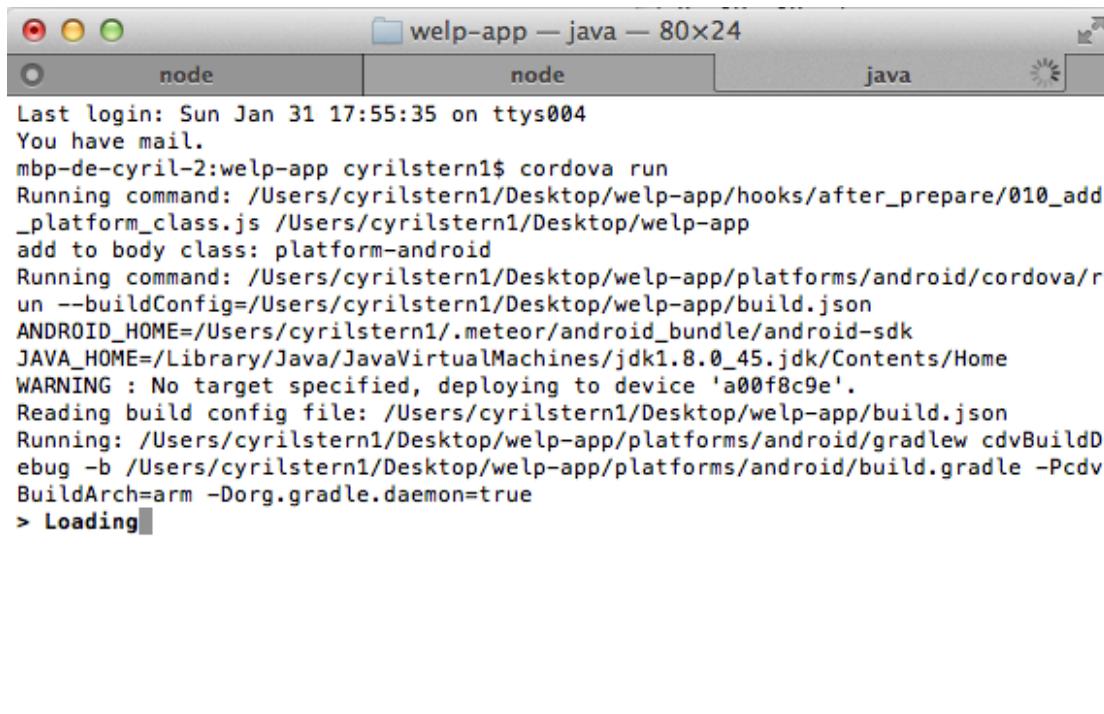
```
welp-app — bash — 80x24
node          node          bash          bash
Last login: Sun Jan 31 17:58:09 on ttys006
You have mail.
mbp-de-cyril-2:welp-app cyrilstern1$ cordova build
Running command: /Users/cyrilstern1/Desktop/welp-app/hooks/after_prepare/010_add_platform_class.js /Users/cyrilstern1/Desktop/welp-app
add to body class: platform-android
Running command: /Users/cyrilstern1/Desktop/welp-app/platforms/android/cordova/b
uild --buildConfig=/Users/cyrilstern1/Desktop/welp-app/build.json
ANDROID_HOME=/Users/cyrilstern1/.meteor/android_bundle/android-sdk
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
Reading build config file: /Users/cyrilstern1/Desktop/welp-app/build.json
Running: /Users/cyrilstern1/Desktop/welp-app/platforms/android/gradlew cdvBuildD
ebug -b /Users/cyrilstern1/Desktop/welp-app/platforms/android/build.gradle -Dorg
.gradle.daemon=true
:preBuild
:compileDebugNdk UP-TO-DATE
:preDebugBuild
:checkDebugManifest
:CordovaLib:compileLint
:CordovaLib:copyDebugLint UP-TO-DATE
:CordovaLib:mergeDebugProguardFiles UP-TO-DATE
:CordovaLib:preBuild
:CordovaLib:preDebugBuild
```

Figure 58 build de l'application

1.2 LA SOLUTION DE DEPLOIEMENT MOBILE (CAS ANDROID)

Pour déployer l'application sur un device Android nous nous sommes servis encore du tasks manager, cordova build et utiliser la CLI cordova run, qui s'est montrée moins capricieuse que ionic run android.

Guide de déploiement sur device



A screenshot of a Mac OS X terminal window titled "welp-app — java — 80x24". The window has three tabs: "node", "node", and "java". The "node" tab is active and contains the following command-line output:

```
Last login: Sun Jan 31 17:55:35 on ttys004
You have mail.
mbp-de-cyril-2:welp-app cyrilstern1$ cordova run
Running command: /Users/cyrilstern1/Desktop/welp-app/hooks/after_prepare/010_add_platform_class.js /Users/cyrilstern1/Desktop/welp-app
add to body class: platform-android
Running command: /Users/cyrilstern1/Desktop/welp-app/platforms/android/cordova/run --buildConfig=/Users/cyrilstern1/Desktop/welp-app/build.json
ANDROID_HOME=/Users/cyrilstern1/.meteor/android_bundle/android-sdk
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
WARNING : No target specified, deploying to device 'a00f8c9e'.
Reading build config file: /Users/cyrilstern1/Desktop/welp-app/build.json
Running: /Users/cyrilstern1/Desktop/welp-app/platforms/android/gradlew cdvBuildDebug -b /Users/cyrilstern1/Desktop/welp-app/platforms/android/build.gradle -PcdvBuildArch=arm -Dorg.gradle.daemon=true
BuildArch=arm -Dorg.gradle.daemon=true
> Loading
```

1.3 DEPLOIEMENT SUR LE STORE

Il n'est pas encore à l'ordre du jour puisque nous n'avons pas encore terminé l'application, nous ne disposons pas encore de compte développeur sur le google store où sur l'app store.

SECURITE

1.1 LA SECURITE DE NOTRE APPLICATION

La sécurité a été de mise lors du développement de l'application. Le framework Angular JS empêche les failles XSS et aussi les injections de code. Nous sommes sur une application mobile donc il est difficilement envisageable que l'User désactive le JavaScript, pour empêcher les contrôles de champs. Néanmoins la couche modèle «API REST» est une seconde protection sur les injections de code.

Diffusion des requêtes en HTTPS et TSL

Enfin la partie logique de l'API REST étant développée avec une identification oauth2 nous nous assurons de la parfaite étanchéité des ressources.

Ce protocole permet à des applications tierces d'obtenir un accès limité à un service disponible via HTTP par le biais d'une autorisation préalable du détenteur des ressources. L'accès est demandé par ce qu'on appelle «un client», et qui peut être un site internet ou une application mobile par exemple. Si les ressources n'appartiennent pas au client, alors ce dernier doit obtenir l'autorisation de l'utilisateur final, sinon il peut directement obtenir l'accès en s'authentifiant avec ses propres identifiants.

CONCLUSION

1.1 MON RETOUR D'EXPERIENCE

Ce stage m'a réellement été très bénéfique. Il m'a permis de revoir un grand nombre de choses que j'avais apprises durant la formation : UML, CSS, JAVASCRIPT, MySQL, SQL, la programmation orientée objet, le design pattern MVC, Il m'a appris énormément d'autre chose, un nouveau langage CoffeeScript, des nouveaux design pattern MVVM et MVW, les Frameworks, Angular JS Cordova, Ionic, le préproesseur SASS, un nouvel environnement de développement : Atom avec ungit gulp bower npm.

La gestion de projet avec trello.

Le développement d'une API REST., et surtout savoir lire une doc.

D'un point de vue personnel, il m'a aussi montré à quel point il fallait être rigoureux, surtout au niveau de l'analyse et de la conception.

Ce stage m'a aussi rassuré sur mes capacités à exercer ce métier et va me permettre une recherche d'activité avec plus d'assurance. Il m'a aussi donné envie d'approfondir mes connaissances dans des domaines que nous n'avons pas traités durant ces deux mois. J'ai apprécié Angular JS par exemple durant la formation, qui nous a permis de réaliser une superbe application.

En conclusion ce stage fut pour moi très réussie pour une formation très enrichissante. Cependant il ne s'agit-là que d'un début car je suis conscient du travail qu'il reste à faire pour devenir un bon concepteur développeur full-stack. Je pense même continuer en parallèle des études pour devenir développeur ingénieur orienté mobile.

BIBLIOGRAPHIE

<https://docs.angularjs.org/guide>
<http://ionicframework.com/docs/>
<https://cordova.apache.org/plugins/>
<https://www.grafikart.fr>
<http://symfony.com/>
<https://getcomposer.org/>
<https://github.com/FriendsOfSymfony/FOSRestBundle>
<https://nodejs.org/>
<http://bower.io/>
<https://openclassrooms.com/courses/redigez-en-markdown>
<https://www.getpostman.com/>
<http://www.riverbankcomputing.co.uk/software/pyqt/intro>
<https://trello.com/>
<https://slack.com/>
<https://cordova.apache.org/>
<https://developers.google.com/maps/documentation/geocoding/intro>
<http://coffeescript.org>
<https://www.google.com/design/spec/material-design/introduction.html>

ANNEXE

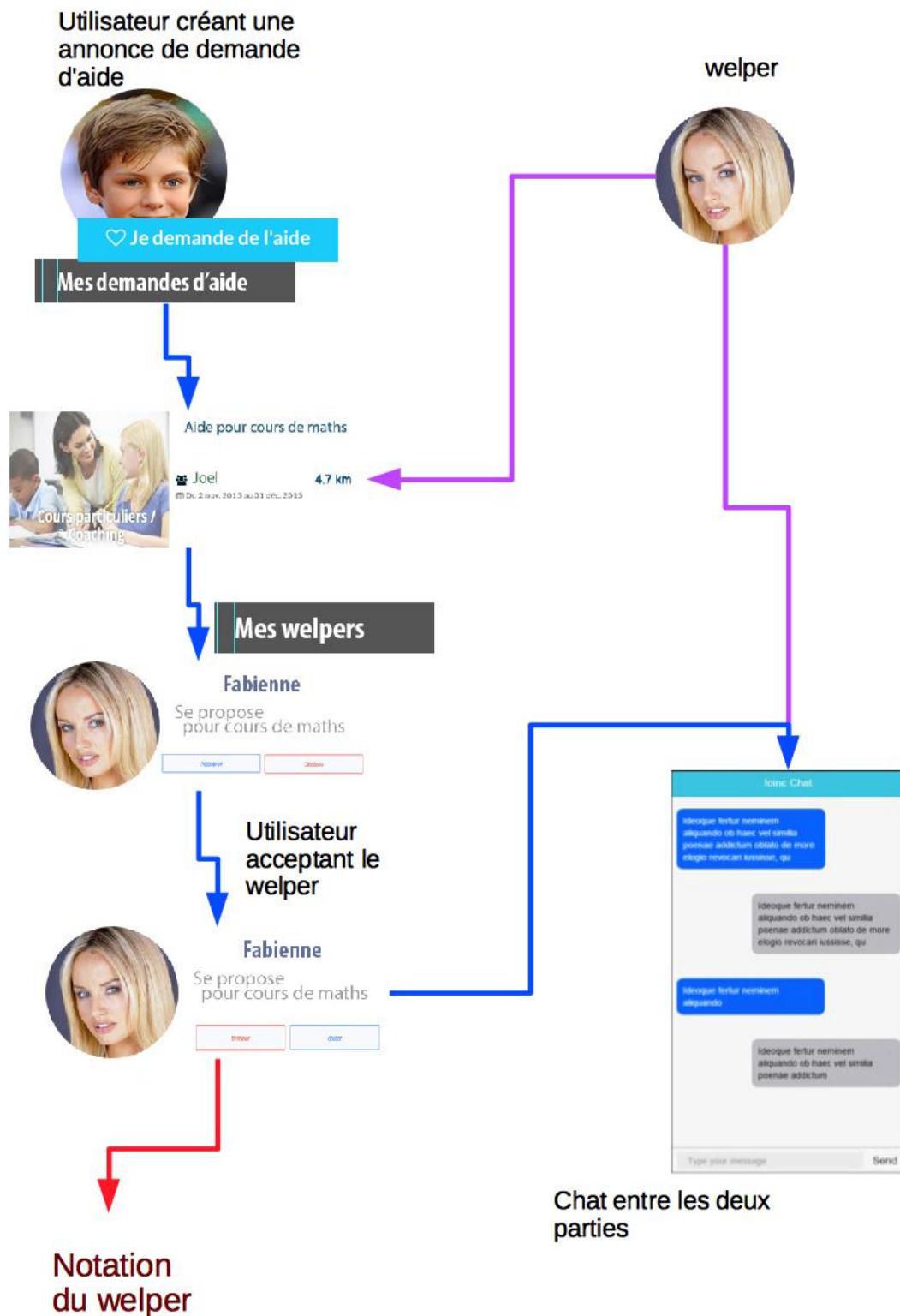


Figure 61 diagramme d'activité/navigation

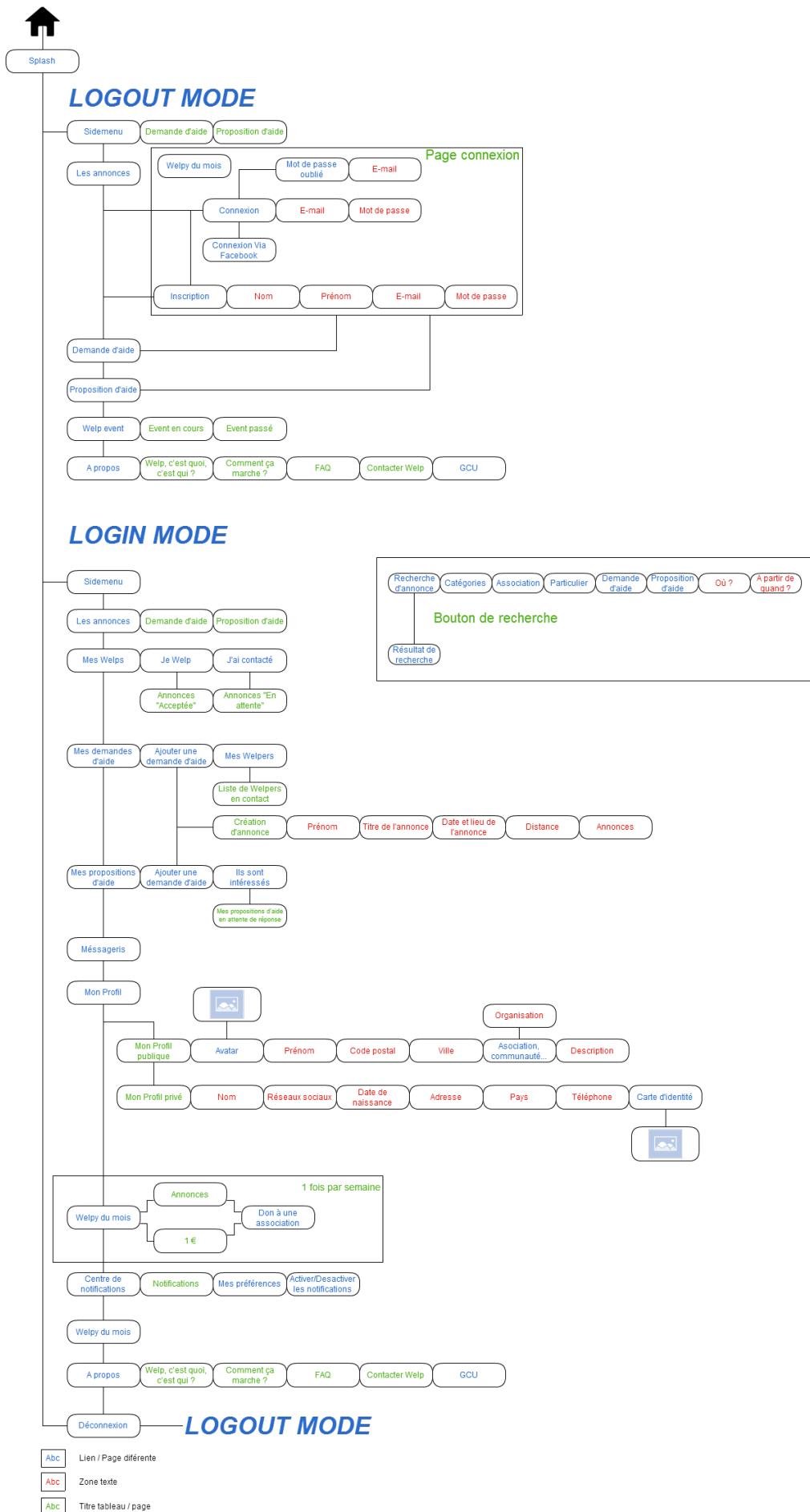


Figure 62 diagramme navigation/wording

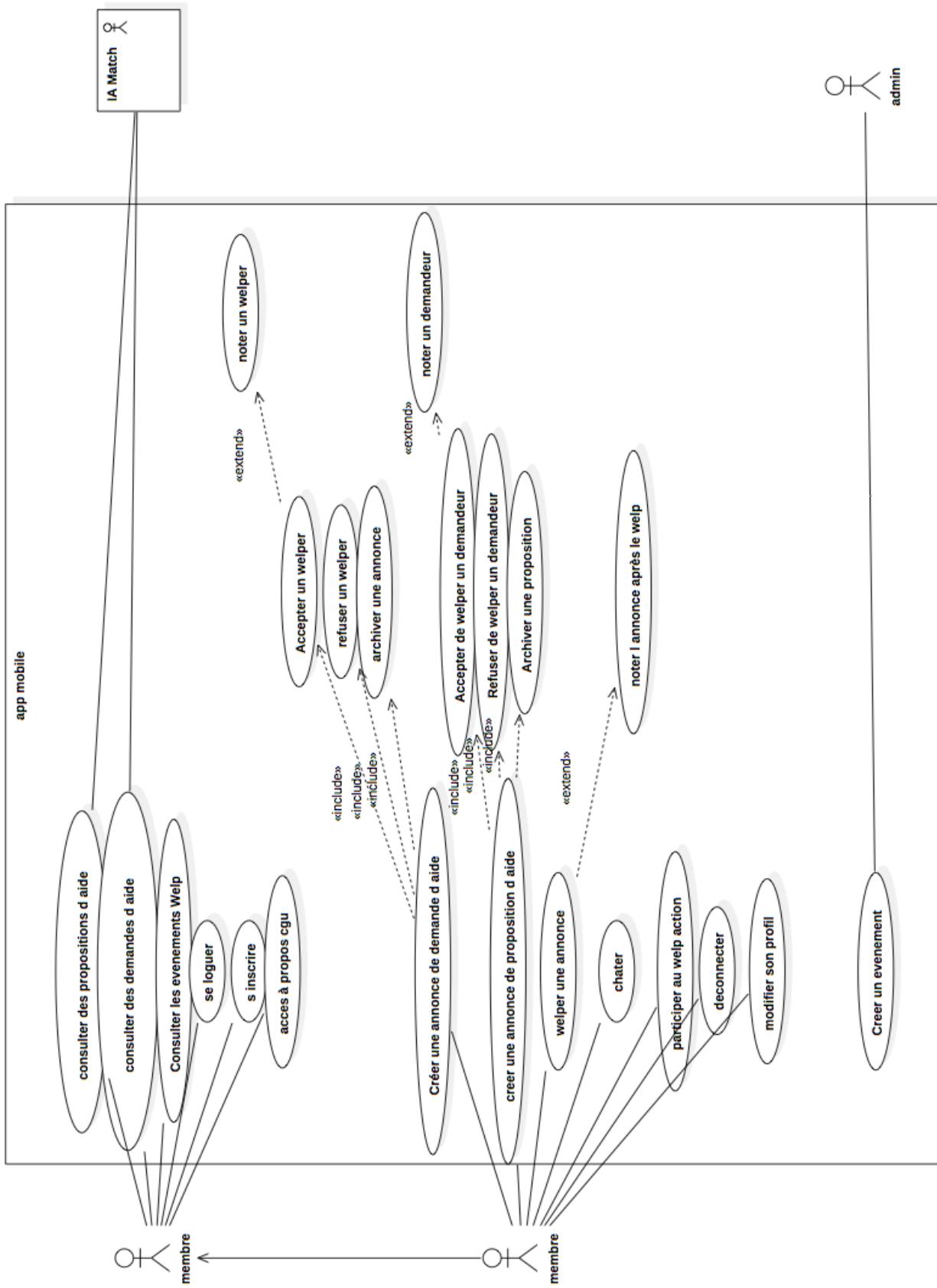


Figure 63 DCU complet welp-app

Ce site utilise des cookies afin d'améliorer l'expérience utilisateur. [Plus d'information](#) [Fermer](#)

welp

[Carte des annonces](#)

[Votez pour Welp !](#) [Carte des annonces](#)

[Inscription](#) [Connexion](#) [Aide ▾](#)

Rechercher quand je déplace la carte

Choisissez les tags

Particulier

Proposition d'aide

Demande d'aide

A partir de quand ?

Recherche avancée ▾

Résultat de votre recherche

20 annonces affichées sur 2108

 **Sylvie** propose "Promener un chien le week end"

Animaux
⌚ 1h ⚖ moins de 1km
📍 Houaria

 #dépannage

Besoin d'aide pour Installation de logiciels
📍 Administratif / Informatique
⌚ 1h ⚖ moins de 1km
📍 Houriia

Welp, la nouvelle façon de s'entraider

> Welp, c'est quoi, c'est qui ?

Welp est un réseau d'entraide qui met en relation des personnes qui ont et celles qui ont envie de donner.

BLOG [LES DERNIERS ARTICLES](#)

INNOVATIONS INDUSTRIELLES

COLLABORATION

Hors-ligne

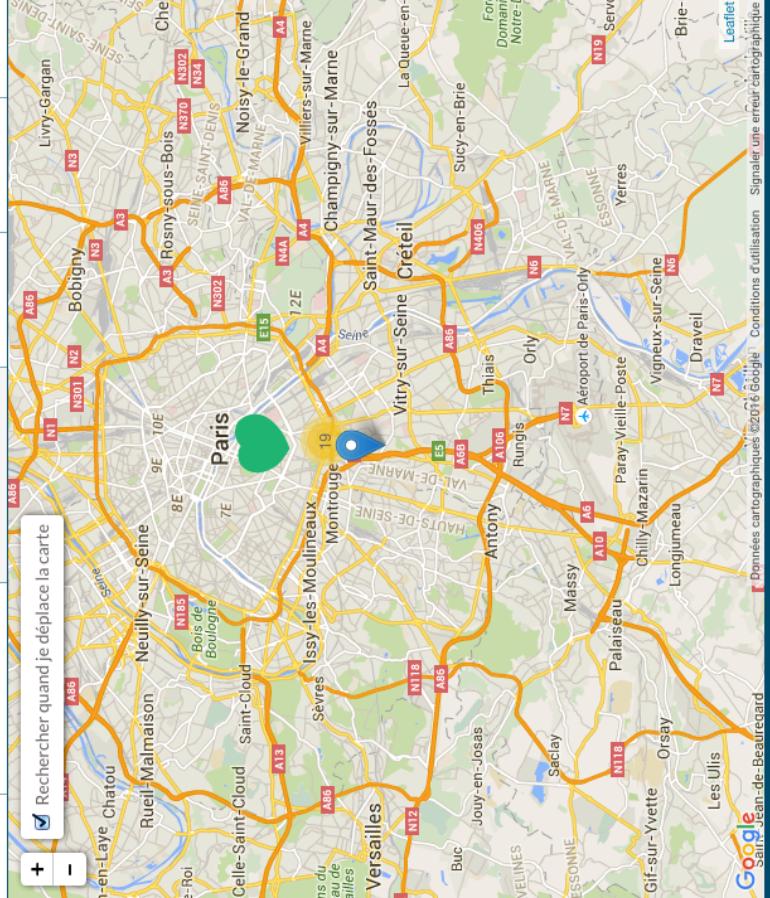


Figure 64 welp site map

ACTIONS POSSIBLES DE L'AUTEUR DE L'ANNONCE SUR LES WELPACTONS EN FONCTION DES ÉTATS DE L'ANNONCE ET DU STATUT DU WELPACTION

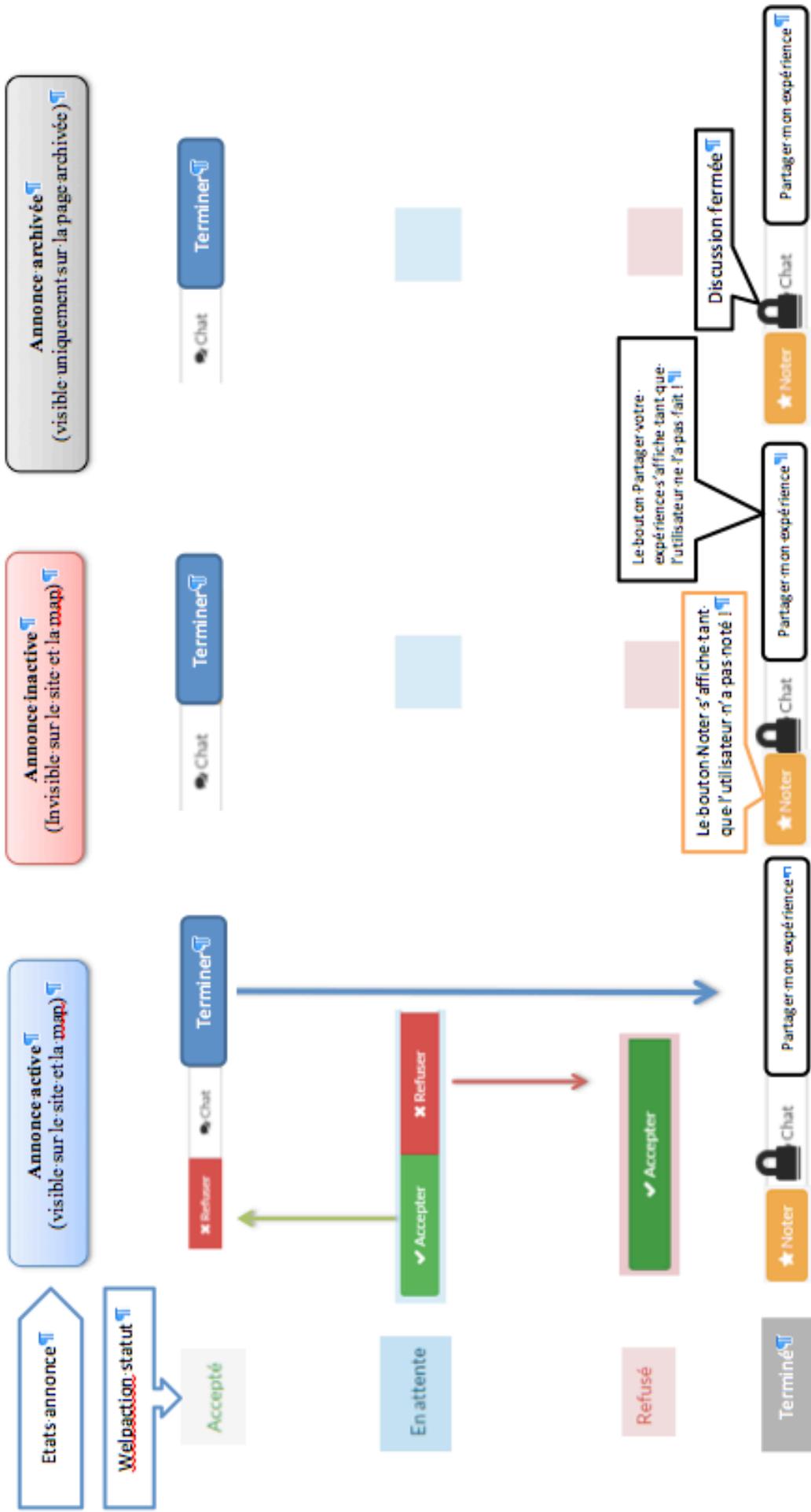


Figure 65 workflow annonce