h e g

Haute école de gestion
Genève

Développement d'une application WEB, client-serveur vs AngularJS.js

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Arnaud LABARBE

Conseiller au travail de Bachelor :

Rolf HAURI, Professeur HES

Carouge, le 12 octobre 2015

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre Bachelor en informatique de gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Carouge, le 12 octobre 2015

Arnaud LABARBE

Remerciements

Je tiens à remercier particulièrement Monsieur Rolf Hauri pour le suivi et le déroulement du travail, mais aussi pour tous les conseils et toute l'aide qu'il a pu m'apporter tout au long du projet.

Je remercie également ma famille pour son aide et ses conseils lors de la rédaction de ce travail mais aussi pour les corrections apportées.

Résumé

Ce travail a pour but de comparer deux manières de concevoir des applications WEB, l'une plus ancienne, à savoir l'architecture client-serveur, l'autre plus récente, l'utilisation de la librairie AngularJS. Au travers des différents chapitres, nous allons retracer et expliquer chaque façon de faire pour ensuite réaliser un prototype d'application de gestion que nous pourrons ensuite analyser.

Afin de réaliser au mieux cette comparaison, le projet sera découpé en quatre parties distinctes :

La première partie sera une comparaison du client-serveur et d'AngularJS.

La deuxième reviendra sur l'historique d'AngularJS ainsi que les concurrents existants.

La troisième partie consistera en l'apprentissage de la librairie en utilisant les tutoriels à disposition sur le site officiel du Framework ou encore ceux de W3Schools.

Enfin, en quatrième et dernière partie, je réaliserai un prototype d'application de gestion avec AngularJS que j'analyserai et auquel je donnerai un ressenti d'expérience personnel.

Table des matières

Liste des tableaux

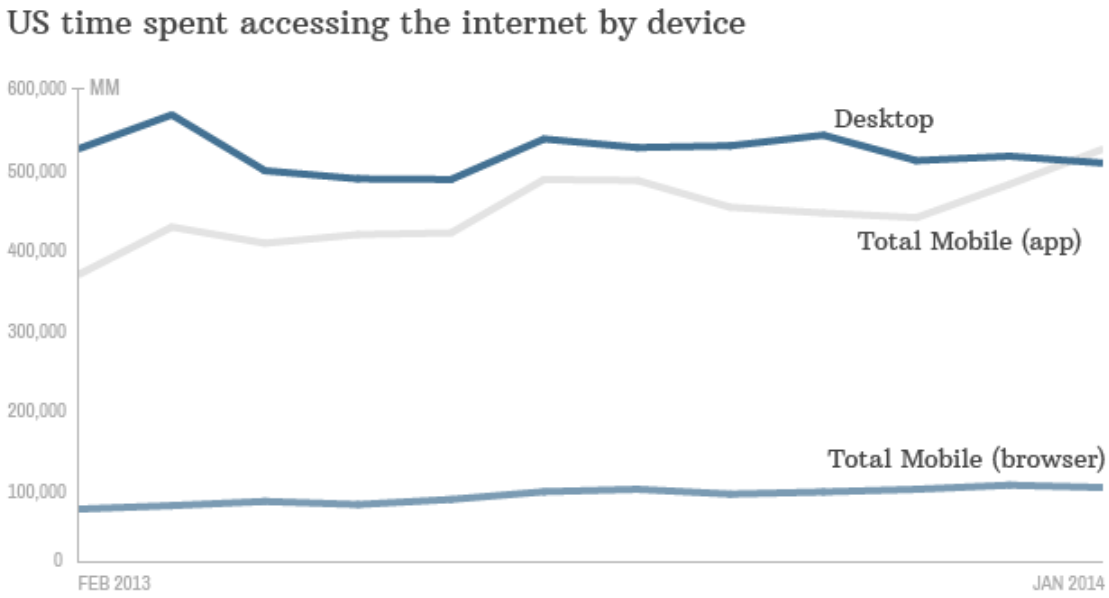
Liste des figures

1. Introduction

Notre façon d'utiliser internet aujourd'hui a complétement changé. En effet, nous sommes devenus beaucoup plus mobiles et internet également. Nous ne nous contentons plus d'être derrière un écran et un clavier pour utiliser le web, nous avons, pour la plupart, un smartphone ou une tablette capable de se connecter à internet où que ce soit.

En janvier 2014, aux Etats-Unis et pour la première fois depuis la création d'internet, l'usage du WEB sur téléphone mobile a dépassé celui sur PC.

Figure 1 : Temps passé sur internet par appareil



http://i2.cdn.turner.com/money/dam/assets/140227174442-mobile-time-spent-1024x576.png

Cette mobilité croissante entraîne des contraintes, puisque notre connexion mobile peut se révéler instable (perte de réseau, faible débit, etc.), mais elle a aussi forcé les développeurs à s'adapter et donc à trouver des solutions pour atténuer les problèmes liés à la mobilité.

En effet, l'architecture client-serveur qui était la plus répandue a dû, peu à peu, laisser place à des architectures ou Frameworks nous permettant de nous affranchir d'un serveur et de pouvoir centraliser les informations dans nos applications.

Ce projet a pour but de montrer l'évolution des besoins des applications WEB et les nouvelles solutions qui sont aujourd'hui développées pour palier à l'architecture classique du client-serveur, notamment grâce au Framework AngularJS, que nous allons utiliser et analyser.

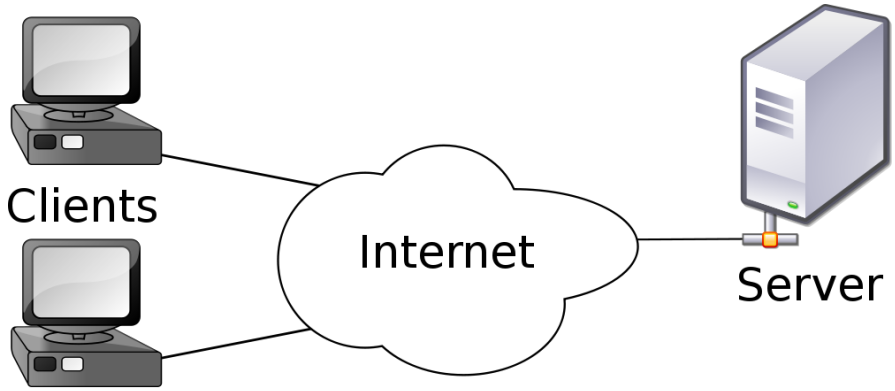
2. Architecture client-serveur vs Angluar.js, les nouveaux besoins des applications de gestion

Dans ce chapitre, je vais présenter l'architecture client-serveur sous les différentes formes possibles. J'évoquerai aussi l'aspect purement technique du Framework AngularJS, qui en fait sa singularité. Je parlerai également des besoins des entreprises qui permettent d'orienter le choix d'infrastructure pour une application. L'évolution des types de client (client léger, client riche) pour les applications web sera en outre abordée.

2.1 Architecture client-serveur

Tout d'abord qu'est-ce qu'une architecture client-serveur ? Cette architecture est basée sur 2 entités, le serveur d'un côté et le client de l'autre. Chacun a son rôle, le client envoie des requêtes au serveur, alors que le serveur attend les requêtes, les traite et y répond. Dans cette architecture, le serveur est dédié et a des capacités supérieures à celles du client, que ce soit en termes de puissance de calcul ou de connexion réseau. Un serveur est capable de répondre à un très grand nombre de requêtes simultanées.

Figure 2 : Modèle client-serveur (architecture à 2 niveaux)



http://imasters.expert/wp-content/uploads/2015/01/client-server.png

L'avènement de cette architecture a eu lieu au début des années 2000 avec l'essor d'internet et des applications WEB. En effet, c'est à cette époque qu'internet est arrivé dans les foyers et a commencé à gagner du terrain.

L'application pour laquelle cette architecture est la plus utilisée est la base de données. En effet, ce modèle permet de centraliser un grand volume d'informations, de les traiter rapidement grâce à la puissance du serveur et de répondre simultanément à plusieurs requêtes, le tout dans un laps de temps très court.

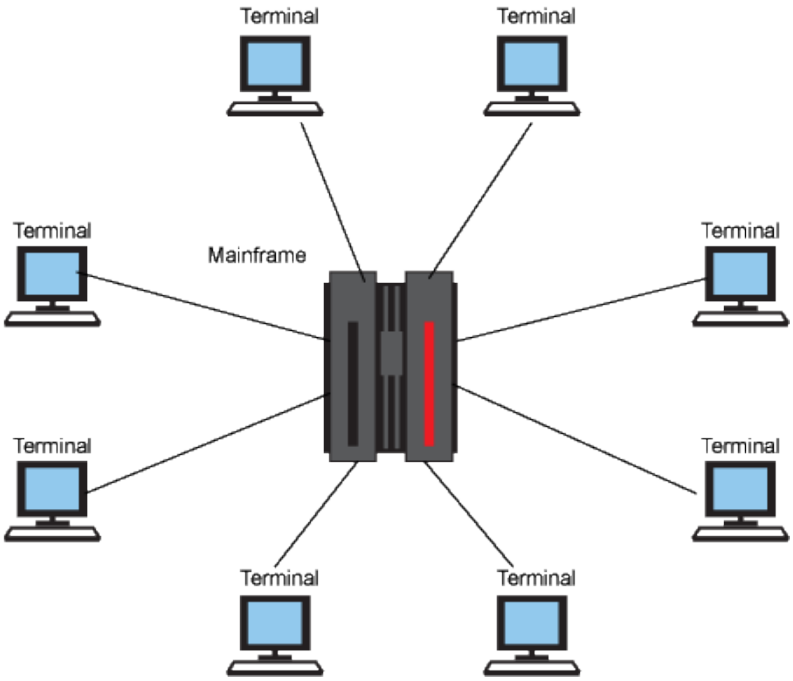
2.2 Les architectures réseaux

L'architecture client-serveur ne repose pas sur un modèle unique comme la figure évoquée précédemment, il en existe plusieurs autres. Cependant celle représentée auparavant est aussi la plus commune et on l'appelle l'architecture à 2 niveaux.

2.2.1 Architecture « mainframe »

Nous avons tout d'abord l'architecture dite « *mainframe* ». Il s'agit de la toute première architecture client-serveur ayant été mise en place au début de l'informatique. Le principe est le suivant : il y a un ordinateur central, le « *mainframe* » auquel sont connectés des ordinateurs passifs (c'est-à-dire des écrans et des claviers, les utilisateurs n'ont pas d'unité centrale). Ainsi, tous les utilisateurs sont connectés au « *mainframe* ». Ce modèle repose entièrement sur les performances et les capacités de l'ordinateur central, ce qui fait de lui un modèle informatique lourd.

Figure 3 : Architecture « *mainframe* »

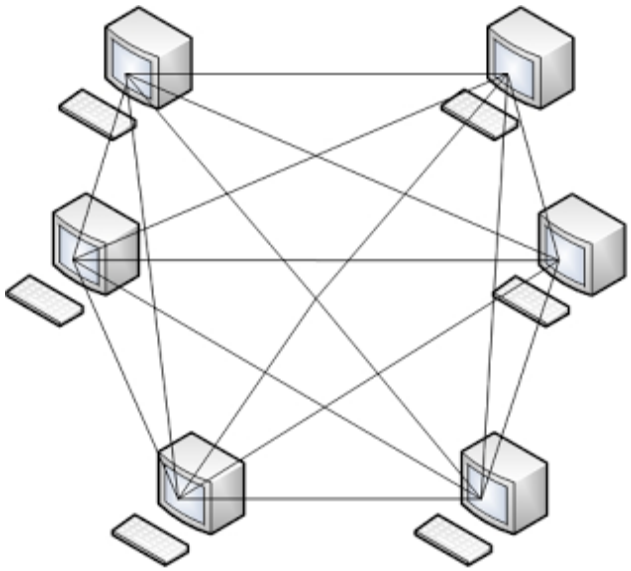


https://www.ibm.com/developerworks/cloud/library/cl-cloudservices3saas/figure1.gif

2.2.2 Architecture pair à pair

Ensuite, nous avons le modèle dit pair à pair (Peer to Peer en anglais ou plus connu sous l'abréviation p2p). Ici, chaque ordinateur connecté au réseau peut être à tour de rôle client et serveur. Cette architecture permet à des données d'être transférées directement d'un ordinateur à un autre sans passer par un serveur central. Ce modèle est surtout connu pour le partage de fichiers sur internet. Cette architecture posant beaucoup de problèmes de sécurité, elle n'est que très peu utilisée par les entreprises et surtout pour les applications de gestion qui contiennent souvent des données sensibles.

Figure 4 : Architecture pair à pair



A peer-to-peer network.

http://3.bp.blogspot.com/_r6XZdgcSL8o/TFX3a0XYoEI/AAAAAAAAANk/6voEUAfXoQE/s320/Peer-to-peer+network.png

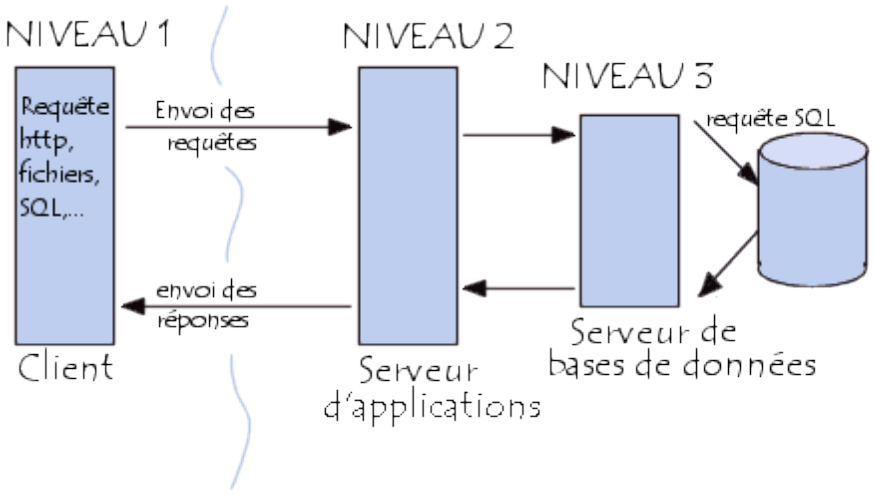
2.2.3 Architecture à 3 niveaux et *n* niveaux

Puis, nous avons l'architecture à 3 niveaux et *n* niveaux. Dans celle-ci nous ajoutons une couche par rapport à la plus commune (celle à 2 niveaux). Elle est composée des éléments suivants :

- Un client équipé d'une interface utilisateur, généralement un navigateur web ;
- Un serveur d'application dédié au traitement métier des données (logique d'application, règle de gestion) ;
- Un serveur de données qui fournit à la couche précédente les données requises ;

Dans cette architecture, chaque couche ne communique qu'avec son voisin direct. De cette structure, découle un modèle à *n* niveaux. On multiplie les intermédiaires pour décharger les serveurs afin qu'ils puissent fournir dans les meilleurs délais les données et le traitement requis par le client. L'inconvénient est qu'une modification sur une entité aura des conséquences sur les voisins immédiats ce qui augmentera le coût de la maintenance. Le nombre de machines rend également cette architecture très coûteuse.

Figure 5 : Architecture à 3 niveaux



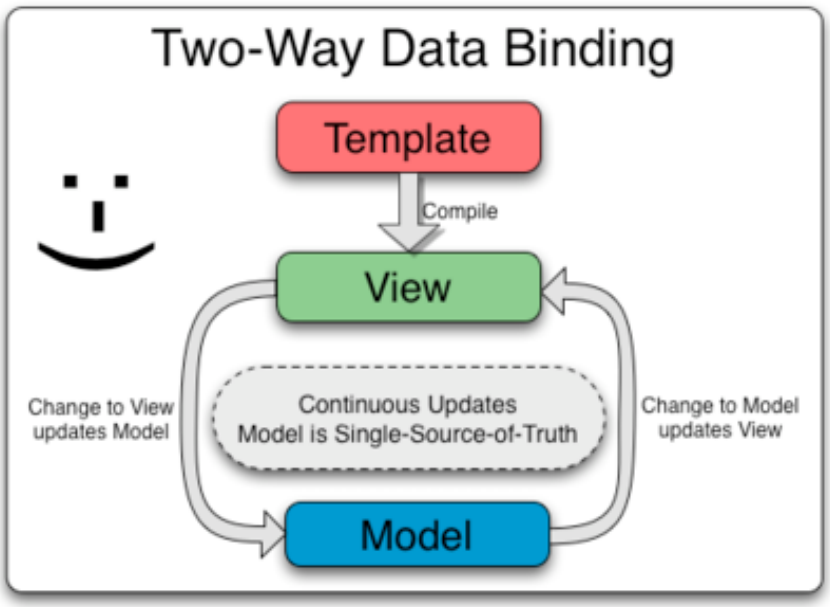
http://static.commentcamarche.net/www.commentcamarche.net/pictures/cs-images-3-tier.gif

2.3 Le Framework AngularJS

AngularJS étend le langage HTML[1] et est une infrastructure que l'on appelle en génie logiciel Modèle-Vue-Contrôleur (abrégé MVC) ou Modèle-Vue-VueModèle (abrégé MVVM) et Google, supporteur du projet, parle même d'infrastructure Model-View-Whatever[2] (abrégé MVW), où le « *Whatever* » signifie en quelque sorte « *ce qui te convient le mieux* ». La librairie laisse donc le choix totalement libre de son type d'infrastructure, le développeur n'est pas cloisonné à un modèle unique.

Le modèle proposé par le Framework permet de séparer les données, la présentation et le traitement. Dans ce contexte, il met à disposition une liaison de données à deux sens « *two-way data binding* », fonctionnant entre la vue et le modèle. Concrètement cela permet de mettre à jour les données affichées côté client dès que le modèle est modifié et vice-versa. Plus besoin de recharger la page, la mise à jour se fait automatiquement et est visible instantanément sur l'écran du client. Tout ceci fait, qu'AngularJS, est beaucoup plus dynamique que les approches traditionnelles des applications web.

Figure 6 : Liaison de données par AngularJS



https://docs.AngularJS.org/img/Two_Way_Data_Binding.png

[1] Hypertext Markup Language
[2] Littéralement Modèle-Vue-Peu importe

AngularJS a changé la façon de développer des applications. En effet, le Framework permet d'effectuer, du côté client, des services qui étaient traditionnellement faits du côté serveur, ce qui réduit la charge et la dépendance à ce dernier.

Le Framework n'empêche cependant en rien la collaboration de l'application avec un serveur, puisqu'il est possible de synchroniser notre modèle avec une base de données sur un serveur et ainsi mettre à jour le modèle chez tous les utilisateurs de l'application. Notre application est autonome, le serveur n'est pas nécessaire au bon fonctionnement de celle-ci.

2.4 Identifier les besoins

Avant de se lancer dans la création d'une application WEB, une entreprise se doit d'identifier ses besoins. Cela lui permettra d'orienter le développement de son application WEB : si elle doit s'orienter vers l'utilisation d'un serveur de données ou si elle peut utiliser une librairie telle qu'AngularJS, voire combiner les deux.

Chaque entreprise est différente ce qui fait que chacune a ses propres besoins. Malgré tout, des questions permettent d'identifier rapidement les nécessités, telles que :

- Combien de collaborateurs sont susceptibles d'utiliser l'application ?
- Les collaborateurs sont-ils susceptibles de consulter l'application en dehors du bureau ?
- Quel sera le volume de données nécessaire à l'application ?

Cela représente bien entendu un échantillon des questions que les entreprises doivent se poser pour identifier leurs besoins. Elles permettent cependant de répondre en partie au côté structurel qu'elles devront mettre en place pour développer leur application.

2.5 L'évolution des applications

Les applications WEB ont évolué et les développeurs ainsi que les infrastructures, s'adaptent au changement de comportement des utilisateurs et de leur besoin. Elles n'ont de cesse d'évoluer.

2.5.1 Client léger

Au début d'internet, donc du web 1.0, les applications web étaient principalement développées en tant que « *client léger* ».

Le web 1.0 se définit par sa passivité, c'est-à-dire que le client se contente juste de consommer de l'information. Ensuite un client léger, ne nécessite qu'un navigateur, présent sur toutes les machines, ce qui permet d'afficher une interface graphique à l'utilisateur, le tout étant fourni par un serveur.

En effet, la page accessible à l'utilisateur n'est que la partie interactive des fonctionnalités, toute la logique et le code métier sont effectués sur le serveur. Le client léger reste donc totalement dépendant du serveur.

2.5.2 Client riche

Le client riche apparaît avec le web 2.0 qui, lui, se définit par son interactivité et son dynamisme. Il s'agit d'un compromis entre un client lourd et un client léger.

Grâce à l'apparition et l'utilisation d'AJAX[3], il devient désormais possible de mettre à jour les données affichées sur la page WEB ou d'accéder à des informations sans avoir à recharger entièrement la page comme cela pouvait être le cas avec le client léger. Malgré tout, le client riche reste dépendant d'un serveur. Même si les pages n'ont pas besoin d'être rechargées, pour accéder aux données les applications ont toujours besoin d'un accès au serveur.

2.5.3 L'arrivée d'AngularJS

L'arrivée du Framework AngularJS vient enrichir les clients riches. En effet, grâce au Framework, il est désormais possible d'intégrer des données directement dans l'application. Celle-ci pourra donc s'affranchir totalement de toute connexion à un serveur distant.

De fait, il devient alors possible de sauvegarder des changements dans les données en local pour que lorsqu'une connexion sera disponible, les données se mettent à jour sur le serveur et que celui-ci répercute cette mise à jour sur les applications de tous les utilisateurs.

2.6 Les nouveaux besoins des applications de gestion

La mobilité croissante des utilisateurs, surpassant même le PC aux USA, on peut aisément imaginer que la tendance arrivera en Europe. Cela entraîne des changements dans les infrastructures nécessaires aux applications WEB et le Framework AngularJS tente d'y répondre.

[3] Asynchronous JavaScript and XML

Renforcé par le soutien de Google, AngularJS devient un poids lourd des Frameworks disponibles pour les développeurs.

Là où l'architecture dite « *classique* » du client-serveur a rendu les applications totalement dépendantes du serveur, que ce soit au niveau de l'interface ou de l'accès aux données, AngularJS renverse les codes établis et donne tous les pouvoirs au client, qui devient maître de l'application.

Les coûts d'exploitation, de maintenance et de développement d'une application WEB développée autour du Framework AngularJS sont bien moindres que ceux engendrés par une infrastructure « *classique* ».

Malgré tout, le choix que l'entreprise effectue doit également répondre à ses besoins et non pas uniquement aux coûts et ce afin d'éviter d'avoir une application qui se révélera alors inutilisable.

3. La librairie AngularJS

3.1 Historique du Framework

AngularJS a été créé par Miško Hevery et Adam Abronsw en 2009, et était à la base, un projet qu'ils développaient indépendamment de leur travail chez Google et qu'ils avaient alors appelé GetAngular. Ils ont dû renommer leur projet en AngularJS, parce que le nom de domaine GetAngular.com était pris. Le nom Angular, fait référence aux balises de déclaration du code HTML « < /> ».

Figure 7 : Logo d'AngularJS

https://upload.wikimedia.org/wikipedia/commons/thumb/c/ca/AngularJS_logo.svg/695px-AngularJS_logo.svg.png

Chez Google, Miško et son équipe de développeurs (ils étaient 3 au total) se sont vus confier un projet nommé Google Feedback. Ils ont écrit en l'espace de six mois de travail environ 17'000 lignes de code. Mais, plus le code grandissait, plus il devenait difficile de tester et de modifier le code, ce qui devenait frustrant pour Miško. Il proposa donc à Google de réécrire entièrement l'application avec le pari qu'il réussirait à le faire en deux semaines. Il échouera son pari, puisque réécrire toute l'application avec son Framework lui prit 3 semaines, mais, l'application est passée des 17'000 lignes de code originelles à seulement 1'500. C'est à partir de ce moment que Google réalise alors que le Framework a un fort potentiel et ce qui était un petit projet indépendant, va alors devenir un projet à part entière soutenu par Google, ce qui va considérablement accélérer son développement.

Le projet a aujourd'hui basculé dans le domaine de l'open-source sous licence MIT et la première version du projet a été mise à disposition sur GitHub le 30 juin 2011 tandis que la version 1.0 d'Angular est, elle, sortie début mai 2012.

3.2 Futur du Framework

Aujourd'hui une version 2.0 du Framework est en préparation et la philosophie de la 2.0 en dit long sur son contenu puisqu'elle sera « Mobile first ». Cependant la refonte complète d'AngularJS risque d'être douloureuse pour les développeurs.

En effet, aucune rétrocompatibilité avec la version une n'est pour l'heure à l'ordre du jour, mais cela reste possible. De plus, il va falloir que les développeurs apprennent les nouveaux concepts et langages introduits dans la version 2.0. Le temps investit dans l'apprentissage du Framework et dans le développement des applications semble avoir un goût amer aujourd'hui.

Suite à ses annonces, la communauté Angular JS, à qui le Framework doit en grande partie son succès, a été déçue. On imagine donc assez mal AngularJS ne pas réagir afin de satisfaire sa communauté.

Cependant, la version finale n'est pas encore sortie mais serait planifiée pour la fin de l'année 2015, ce qui laisse encore la place à des changements.

Il est possible de suivre le développement de la 2.0 sur leur nouveau site : https://angular.io.

3.3 Principes d'AngularJS

Le Framework est basé sur l'extension du code HTML avec de nouvelles balises et est basé, comme on a pu le voir au chapitre précédent, sur le patron Modèle-Vue-Contrôleur. Il encourage le faible couplage entre la vue, la logique métier et les données. La philosophie d'Angular est de rendre la vie plus simple au développeur comme à l'utilisateur.

De plus, la liaison de données bidirectionnelle est un point essentiel du Framework, puisque cela permet de mettre à jour les données du modèle directement depuis la vue, mais également de mettre à jour la vue une fois les modifications dans le modèle effectuées. Les applications s'en trouvent ainsi plus dynamique et interactives.

Les applications AngularJS sont faites pour être du « *single page* », pas besoin de naviguer sur différentes pages, tout est accessible sur une seule et unique page.

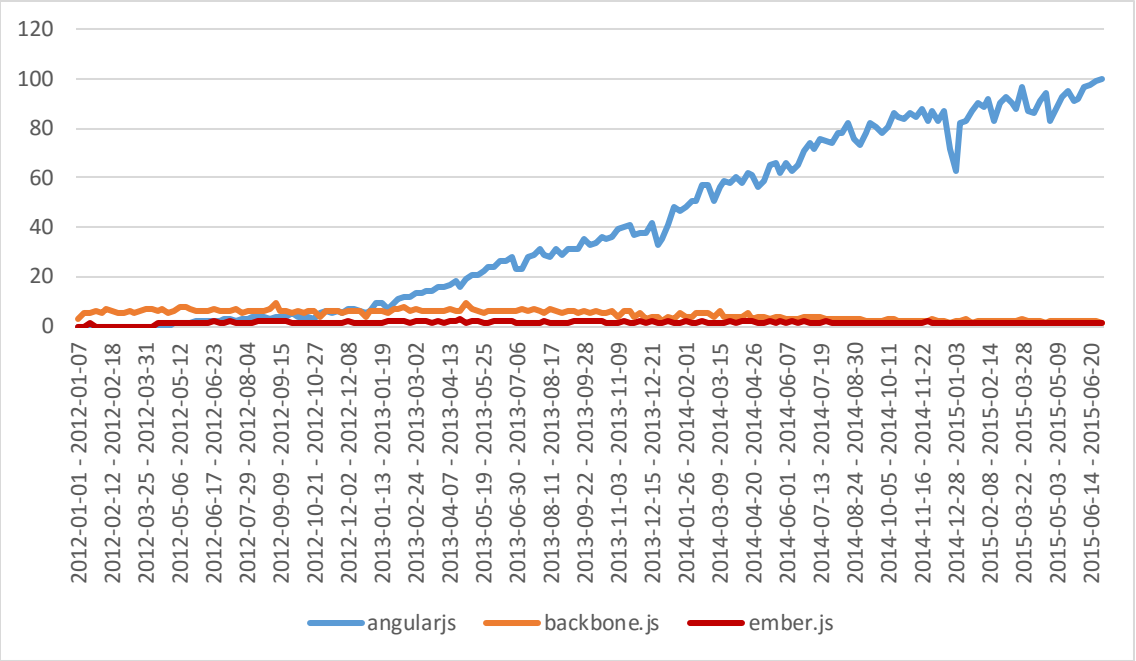
3.4 La concurrence

AngularJS n'est pas le seul Framework disponible, il en existe bien d'autres sur le marché. Ce chapitre va donc distinguer les principaux concurrents d'AngularJS. On trouve deux concurrents directs, il s'agit de Backbone.js et Ember.js

3.4.1 Comparatif de popularité

Une simple recherche Google Trends[4] suffit pour se rendre compte du fossé qui sépare, en termes de popularité au niveau des recherches Google, Angular de ses concurrents.

Tableau 1 : Google Trends de AngularJS, Backbone.js et Ember.js



Google Trends

Le graphique s'étend de janvier 2012 à juillet 2015. A noter que Backbone.js est né en 2010 tandis que les origines d'Ember.js remontent elles à 2007 sous le nom de SproutCore, avant d'être repris en 2011 et renommé Ember.

Chaque Framework peut compter parmi ses utilisateurs, des sites de renommée mondiale. Ci-dessous une liste d'exemple pour chaque Framework :

Backbone.js

- Reddit.com
- Linkedin.com
- Tumblr.com

Ember.js

- Nasa.gov
- Vine.co
- Twitch.tv

[4] outil qui permet de connaître la fréquence d'utilisation d'un terme dans le moteur de recherche Google

Angular.js

- Vevo.com
- Msnc.com
- Youtube.com

3.4.2 L'aspect communautaire

La communauté est une part importante dans le choix d'un Framework. En effet, plus la communauté est grande plus on a de chance, lorsque l'on rencontre un problème, de trouver la réponse via un moteur de recherche ou, via des forums spécialisés comme Stackoverflow où il est possible d'expliquer un problème et de recevoir une réponse d'un utilisateur. La quantité de tutoriels aussi s'agrandit avec la communauté, que ce soient des tutoriels écrits ou bien encore vidéos que l'on peut trouver sur Youtube par exemple.

La communauté participe également aux améliorations du Framework en proposant ses propres contributions ou en proposant des librairies annexes gravitant autour du Framework. Tout ceci est rendu possible grâce à GitHub.

Tableau 2 : Comparatif des communautés

Metric	AngularJS	Backbone.js	Ember.js
Stars on GitHub	40.2k	18.8k	14.1k
Third-Party Modules	1488 ngmodules	256 backplugs	1155 emberaddons
StackOverflow Questions	104k	18.2k	15.7k
YouTube Results	~93k	~10.6k	~9.1k
GitHub Contributors	96	265	501
Chrome Extension Users	275k	15.6k	66k
Open Issues	922	13	413
Closed Issues	5,520	2,062	3,350

https://www.airpair.com/js/javascript-framework-comparison

On peut voir la force communautaire d'AngularJS au travers de ce tableau, notamment par les questions sur StackOverflow qui sont presque dix fois supérieures aux deux autres Frameworks. Malgré tout, étant open-source et avec une telle popularité AngularJS n'a que peu de contributeurs vis-à-vis de ses concurrents.

Angular est un Framework à part entière et n'a aucune dépendance avec d'autres librairies, ce qui n'est pas le cas pour Ember.js et Backbone.js, puisqu'ils ont tous les deux besoins de JQuery et d'une autre librairie pour fonctionner.

Tableau 3 : Comparatif des dépendances

Framework	Net Size	Size with required dependencies
AngularJS 1.2.22	39.5kb	39.5kb
Backbone.js 1.1.2	6.5kb	43.5kb (jQuery + Underscore) 20.6kb (Zepto + Underscore)
Ember.js 1.6.1	90kb	136.2kb (jQuery + Handlebars)

https://www.airpair.com/js/javascript-framework-comparison

3.4.3 Les autres

Backbone.js et Ember.js ne sont pas les seuls concurrents qui existent pour AngularJS, il en existe d'autres qui sont marginaux et n'ont donc que peu d'utilisateurs, mais ils ont malgré tout le mérite d'exister.

On peut citer entre autres Knockout.js, Aurelia mais aussi MooTools ou encore React.js.

La quantité de Frameworks disponible rend le secteur extrêmement concurrentiel et il devient difficile de tirer son épingle du jeu.

4. Apprentissage d'AngularJS

Après avoir expliqué et détaillé le Framework, je vais maintenant apprendre à coder via des tutoriels disponibles sur internet et ce afin de voir les possibilités proposées par le Framework mais aussi pour voir s'il est difficile à manipuler. Je vais donner mon ressenti par rapport aux tutoriels que j'ai suivi pour mon apprentissage du Framework.

4.1 La recherche Google

Le premier réflexe afin de trouver ce que l'on cherche sur internet est, en général, de passer par le moteur de recherche Google.

Le moins que l'on puisse dire dans le cadre de l'apprentissage du Framework, c'est que la quantité est au rendez-vous, puisqu'une simple recherche avec les mots clés « *tutorial angularjs* » affiche 1'380'000 résultats (la recherche a été effectuée dans une fenêtre de navigation privée). La grande majorité des résultats disponibles sont en anglais, ce qui, pour moi, n'est pas un obstacle. Malgré la quantité de résultats pour ce critère de recherche, un utilisateur va rarement plus loin que la première page. En effet, 91% des cliques se concentrent sur les dix premiers résultats de la première page, au-delà de ces dix premiers résultats il est donc difficile pour le site d'être consulté. Si l'utilisateur n'a pas trouvé ce qu'il cherche parmi les dix premiers, il y a de fortes chances qu'il change ses critères de recherche.

Sur la première page en sixième position, avec ce critère de recherche, on ne trouve qu'un seul site en français et il s'agit de Grafikart. Le tutoriel, est uniquement vidéo[5], et il s'agit ici de faire une « *To do list* ». Cependant si l'accès à la vidéo est gratuit, puisqu'elle est disponible sur YouTube, le code source de l'application finie est quant à lui payant. Le tutoriel vidéo dure environ quarante minutes mais les tutoriels vidéo sont difficiles à suivre tout en codant. On se retrouve à devoir mettre en pause très souvent pour écrire le bout de code qui vient d'être fait en vidéo.

On trouve également un autre résultat en français[6]. Ici ce n'est pas un tutoriel mais un article qui prodigue des conseils sur comment bien démarrer avec AngularJS en donnant des conseils et des références à suivre, notamment sur le site officiel d'AngularJS.

[5] http://www.grafikart.fr/tutoriels/angularjs/angularjs-todo-369
[6] http://www.frangular.com/2013/03/comment-aborder-angularjs.html

4.2 Le tutoriel du site officiel

Avant de commencer par faire des recherches de tutoriels sur Google, mon premier réflexe a été d'aller sur le site officiel du Framework afin de voir s'il proposait un tutoriel, ce qui est bel et bien le cas[7].

Le tutoriel propose de construire une application qui liste des appareils fonctionnant sous Android. Avant de commencer l'introduction au code, il faut auparavant installer quelques logiciels. Il est demandé d'installer Git afin de pouvoir récupérer sur GitHub le répertoire de l'application proposé dans le tutoriel, puis d'installer Node.js qui nous permet d'avoir un serveur web préconfiguré et des outils de tests. Une fois les deux logiciels installés, le tutoriel nous demande de tester certaines commandes afin de savoir comment lancer le serveur WEB (tu l'as mis en majuscule dans les paragraphes précédents) et les outils de tests.

L'application complète du tutoriel est disponible en ligne[8].

L'application se construit pas à pas, via une liste de chapitres, en partant au début d'une page HTML statique classique à une page HTML dynamique intégrant AngularJS. Au fur et à mesure on ajoute des fonctionnalités à notre page comme par exemple une barre de recherche, un tri des appareils selon l'ancienneté ou par ordre alphabétique, des images cliquables, des liens vers le détail de chaque appareil, etc.

Le tutoriel ne propose pas de « faire par soi-même » l'application, l'utilisateur est invité à taper des commandes Git afin de récupérer le code complet de chaque chapitre. Il montre cependant les ajouts et les différences par rapport au chapitre précédent. A la fin de certains chapitres, il nous est proposé de faire des ajouts directement dans le code des contrôleurs ou de l'index, ce sont des « *Experiments* [9]» qui rajoutent des fonctionnalités ou qui embellissent certaines déjà existantes.

A noter que le tutoriel n'est disponible qu'en anglais, ce qui peut être un frein pour les anglophobes.

J'ai suivi ce tutoriel et installé tous les éléments nécessaires sur une machine virtuelle tournant sous Ubuntu 14.04 sans rencontrer aucun problème. J'ai refait la même chose sur ma machine, tournant sous Windows 10 Professionnel, pour vérifier si tout fonctionnait également et là encore je n'ai rencontré aucun problème.

[7] https://docs.angularjs.org/tutorial
[8] http://angular.github.io/angular-phonecat/step-12/app/#/phones
[9] Expériences

Le site officiel propose également un « Guide du développeur[10] » qui est une documentation complète du Framework. Chaque chapitre de ce guide contient également des exemples de mise en œuvre (contrôleurs, formulaires, filtres, etc.).

Avec son tutoriel et son guide du développeur, le site officiel du Framework est une très bonne référence pour les développeurs puisqu'il est très complet.

4.3 Le tutoriel de W3Schools

Lorsque l'on effectue une recherche Google, W3Schools est le second site à sortir de la liste juste derrière le site officiel.

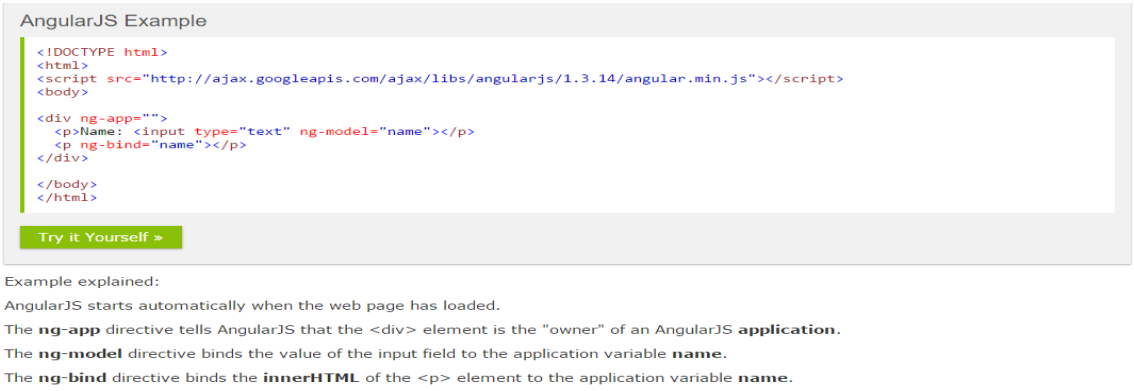
W3Schools est très connu puisqu'il propose des apprentissages pour beaucoup de langage du WEB : c'est un site de référence pour les développeurs. Parmi les apprentissages proposés il y a donc un tutoriel pour AngularJS[11].

Ici, pas d'application concrète à construire, on suit des chapitres qui introduisent de nouvelles notions à chaque fois. Au travers de ces chapitres on voit tout ce qui concerne les bases d'AngularJS (directives, expressions, modules, contrôleurs, filtres) jusqu'à des notions plus poussées qui nous introduisent au DOM, aux événements, aux formulaires, aux inputs et bien d'autres.

Pour chaque nouvelle notion, il y a un exemple de code qui nous montre comment cela est réalisé et un bouton « *Try it yourself* [12]» nous permet de voir ce que cela fait dans le navigateur internet. On peut également modifier le code source pour tester par nous-même et voir le résultat instantanément à la manière de JSFiddle. Chaque exemple est ensuite expliqué afin de savoir à quoi correspondent les éléments dans le code juste au-dessus.

[10] https://docs.angularjs.org/guide
[11] http://www.w3schools.com/angular/default.asp
[12] Essayez par vous-même

Figure 8 : Exemple de W3Schools



http://www.w3schools.com/angular/angular_intro.asp

Contrairement au tutoriel officiel, il n'y a ici pas besoin d'installer de logiciel tiers, un navigateur internet suffit pour l'apprentissage. Les exemples proposés sont simples dans la mise en œuvre ce qui facilite la compréhension.

4.4 Le cours d'OpenClassrooms

Pendant mes recherches pour les chapitres précédents de ce travail j'ai eu l'occasion de tomber sur un cours complet, et en français, concernant AngularJS sur le site OpenClassrooms. C'est un site d'E-éducation, qui propose des cours pour tous, gratuits ou payants avec possibilité d'obtenir des certificats pour certains de ceux-ci. On peut donc y trouver un grand nombre de cours relatifs à la programmation WEB.

Proposé gratuitement, ce cours et ce tutoriel[13] nous introduisent donc à AngularJS. Il s'agit bien d'un cours, puisqu'il y a une partie théorique et une partie pratique avec un tutoriel. A noter, et cela est important, que sur ce site afin de pouvoir profiter entièrement du cours, au bout d'un certain nombre de pages visitées il faut s'inscrire pour voir le reste du contenu. L'inscription est totalement gratuite et ne nécessite rien de particulier, juste une adresse e-mail.

Le tutoriel nous propose de faire une application de cinéma grâce à l'API de themoviedb. Le tutoriel est à réaliser sur une distribution Linux, j'ai donc à nouveau utilisé ma machine virtuelle sous Ubuntu 14.04 pour le suivre.

On retrouve ici le même type de tutoriel que sur le site officiel puisqu'il faut en effet récupérer le projet sur GitHub via une ligne de commande. Cependant, il faut installer énormément de packages annexes et il n'est pas rare de se retrouver avec des erreurs lorsqu'il nous est demandé de taper certaines commandes dans le terminal. Au final,

[13] https://openclassrooms.com/courses/developpez-vos-applications-web-avec-angularjs

on se retrouve très souvent sur Google pour soit trouver les noms des packages manquants et les télécharger, soit pour connaître la cause de notre erreur. Même en suivant le tutoriel à la lettre on se sent un peu seul face au manque d'informations.

En suivant le tutoriel, il est demandé d'ajouter du code dans des fichiers sans spécifier où ils se trouvent et sans non plus que l'on sache vraiment ce que l'on fait. On se contente de rajouter du code sans explication quant à son contenu, ce qui ne facilite pas l'apprentissage.

Au final, j'ai réussi, non sans mal à faire marcher l'application, mais cela a été beaucoup plus fastidieux que sur le site officiel AngularJS.

Même si le tutoriel est en français, je ne le recommande pas. La quantité de temps à passer sur Google pour trouver par soi-même des solutions aux problèmes et le manque d'informations du code que l'on rajoute n'en font pas un bon tutoriel. Cependant la partie théorique s'avère elle intéressante et peut être une bonne base à l'apprentissage avant de coder.

4.5 Les autres tutoriels

Etant donné la quantité de tutoriels disponibles, d'autres ont retenu mon attention.

Je citerai notamment Tutorialspoint[14]. Entièrement gratuit, ce tutoriel est à l'image de celui de W3Schools, pas de création d'application juste la mise en œuvre et l'explication des notions d'AngularJS. Le petit plus est que, le tuto est également disponible en version PDF, ce qui est fort appréciable. On a ici un tutoriel complet qui nous apprend les bases et les concepts d'AngularJS, c'est une bonne alternative à W3Schools.

Pour ceux qui affectionnent les tutoriels vidéo, je recommande Egghead[15]. Ici tout est sous forme vidéo, on nous présente AngularJS et ses possibilités.

Il existe bien d'autres tutoriels disponibles, cela va du gratuit au payant, mais ce qui est certain c'est que tout le monde y trouvera son compte. En revanche, il est conseillé d'avoir de bonnes connaissances en anglais, la plus grande majorité n'étant pas en français.

4.6 Qu'ai-je vraiment appris ?

Après avoir suivi tous ces tutoriels, finalement qu'est-ce que cela m'a permis d'apprendre ?

[14] http://www.tutorialspoint.com/angularjs/index.htm
[15] https://egghead.io/articles/new-to-angularjs-start-learning-here

Grâce à W3Schools, j'ai appris les concepts de base d'AngularJS, même si ceux-ci sont expliqués de manière très simpliste. Les exemples d'applications permettent une mise en pratique rapide du concept avec un exemple à la portée de tous.

Le site officiel, quant à lui, m'a permis de voir tous les concepts dans une application concrète que l'on construit pas à pas. On y trouve également des scripts de tests que l'on peut lancer grâce aux outile de Node.js afin de voir si l'application est fonctionnelle en l'état.

4.6.1 Le concept de route

AngularJS encourage les applications « *single page* », les routes permettent malgré tout de créer des URL qui sont « *user friendly*[16] », l'utilisateur peut ainsi instantanément savoir sur quelle page il se trouve en se référant à l'URL. Plutôt que d'avoir un URL constant du type :

- example.com

Il devient alors possible d'avoir un URL spécifique pour les pages atteintes

- example.com/contact

L'index de l'application est un modèle de présentation dans lequel on affiche nos vues grâce aux routes que l'on aura défini.

Enfin, les routes permettent de définir un contrôleur propre à chaque page, ce qui évite les contrôleurs volumineux en ligne de codes et permet de séparer les tâches en fonction de la route sur laquelle l'utilisateur se trouve.

4.6.2 Le concept de service

Le développeur peut créer ses propres services et les appeler dès que nécessaire, comme il le ferait pour des fonctions JavaScript. Ceci s'avère très utile pour des tâches redondantes à plusieurs contrôleurs. En effet, avec les routes, les contrôleurs sont séparés, un service sera donc plus efficace que de copier/coller la fonction que l'on a créé dans un contrôleur précédent. Un service peut être considéré comme une fonction globale.

4.6.3 Validation de formulaire

Au sein d'une application, il n'est pas rare d'utiliser un formulaire. AngularJS, met à disposition la validation des champs d'un formulaire du côté client. Pas besoin pour le développeur de tester par programmation si les champs sont remplis correctement. Il

[16] Facile d'utilisation

devient donc facile de pourvoir valider notre formulaire pour un ajout ou une édition des données de notre modèle.

Il est bien entendu qu'une validation côté client ne suffit pas à elle seule pour vérifier les données entrées par l'utilisateur, une validation côté serveur est également nécessaire.

4.6.4 Angular et Bootstrap

Bootstrap est certainement la feuille de style la plus populaire grâce à son CSS. Angular permet de travailler avec le style de Bootstrap sans aucun souci. D'ailleurs, le tutoriel du site officiel l'utilise pour le développement de l'application test du Framework.

Les directives Angular s'intègre tout à fait avec le style de Bootstrap.

4.6.5 Le service $http

Le Framework met à disposition le service $http qui permet un accès serveur qui engendre une réponse de type success si la requête a abouti ou error si c'est le contraire, comme en AJAX. Le service $http permet de créer un client REST[17]ful et met donc à disposition les méthodes REST, à savoir :

- $http.get
- $http.post
- $http.put
- $http.delete

Ce sont les méthodes les plus utilisées.

Grâce à ce service, les applications peuvent mettre en place un CRUD[18] complet avec un serveur distant.

4.6.6 Les filtres

Les filtres permettent de transformer les données. Ils peuvent être ajoutés directement aux expressions ou dans les directives AngularJS grâce au caractère « | ». Cela permet notamment de trier et d'effectuer des recherches dans nos tableaux. Des filtres existent déjà dans le Framework comme par exemple « *uppercase* » qui permet d'afficher tout en majuscule ou encore « *orderBy* » qui permet la même chose qu'en SQL, à savoir le tri de nos données selon un critère.

[17] Representaional State Transfer
[18] Create, Read, Update, Delete

Cependant, le développeur peut tout à faire créer son propre filtre adapter à son application si le filtre n'existe pas déjà dans l'API d'AngularJS.

4.6.7 Les directives

Les directives permettent d'attacher des comportements spécifiques à des éléments HTML de notre page. Concrètement cela ressemble un peu à des listeners en Java pour rendre notre page interactive.

Avec tous ces concepts et après avoir suivi les tutoriaux cités précédemment, je peux maintenant développer mon prototype d'application de gestion développé avec AngularJS.

5. Réalisation d'un prototype d'application de gestion avec AngularJS

Ce chapitre a pour but de présenter brièvement mon prototype d'application de gestion.

5.1 Résumé du prototype

J'ai décidé de partir sur une application de gestion des Ressources Humaines d'une entreprise. Pour cela je vais utiliser une base de données MySQL que je vais créer.

La page d'accueil de l'application regroupera la liste des employés que je pourrai trier selon des critères prédéfinis, et dans laquelle je pourrai également faire des recherches syntaxiques.

L'application permet d'afficher toutes les informations concernant un employé, éditer ses informations, gérer ses jours de congés restants ainsi que les jours maladie que l'employé aura cumulé et enfin ajouter un employé à l'entreprise.

5.2 Environnement de développement

Sur ma machine virtuelle tournant sous Ubuntu Desktop 14.04, j'ai créé un serveur de fichiers avec Samba ce qui m'a permis de développer directement sur mon serveur WEB local.

Pour écrire le code j'ai utilisé Notepad++ sous Windows 10 grâce au serveur de fichiers.

L'application WEB a été testée sur les navigateurs Chrome, Firefox et Edge.

5.3 La base de données

La base de données de mon prototype a été faite sous MySQL.

Le nom de la base est « *employe* » et la table qu'elle contient s'appelle « *rh* ». La structure de la table est la suivante :

- rh_id : identifiant de l'employé (s'incrémente automatiquement)
- rh_prenom : prénom de l'employé
- rh_nom : nom de l'employé
- rh_dateNaissance : date de naissance de l'employé (type date)
- rh_adresse : adresse de l'employé
- rh_tel : numéro de téléphone fixe de l'employé
- rh_natel : numéro de natel de l'employé

- rh_mail : adresse e-mail personnel de l'employé
- rh_département : département dans lequel travaille l'employé
- rh_embauche : date d'engagement de l'employé (type date)
- rh_conge : jours de congés restant à l'employé (par défaut 25)
- rh_maladie : jours de maladie cumulés par l'employé (par défaut 0)

Figure 9 : Structure de la table rh

#	Nom	Type	Interclassement
1	rh_id	int(11)	
2	rh_nom	varchar(50)	latin1_swedish_ci
3	rh_prenom	varchar(50)	latin1_swedish_ci
4	rh_dateNaissance	date	
5	rh_adresse	varchar(70)	latin1_swedish_ci
6	rh_tel	varchar(20)	latin1_swedish_ci
7	rh_natel	varchar(20)	latin1_swedish_ci
8	rh_mail	varchar(70)	latin1_swedish_ci
9	rh_departement	varchar(70)	latin1_swedish_ci
10	rh_embauche	date	
11	rh_conge	int(11)	
12	rh_maladie	int(11)	

phpMyAdmin sur mon serveur WEB

5.4 Arborescence de l'application

A la racine de mon application on retrouve l'index qui permet d'arriver sur la page d'accueil du prototype.

5.4.1 bower_components

Ce dossier inclut toutes les librairies que j'ai téléchargées avec Bower et requises pour le bon fonctionnement de l'application.

5.4.2 db

Il contient tous les accès à la base de données. Le fichier *db.php* est quant à lui inclus dans tous les autres fichiers PHP pour la connexion à la base de données. C'est ce fichier qu'il faut modifier avec le nom d'utilisateur et le mot de passe de votre environnement pour que l'application marche. Les fichiers PHP présents dans ce dossier, permettent de sélectionner les employés ou un employé en particulier, d'update et d'effacer les employés.

5.4.3 js

On retrouve dans ce dossier tous les scripts personnalisés et propre aux Frameworks AngularJS.

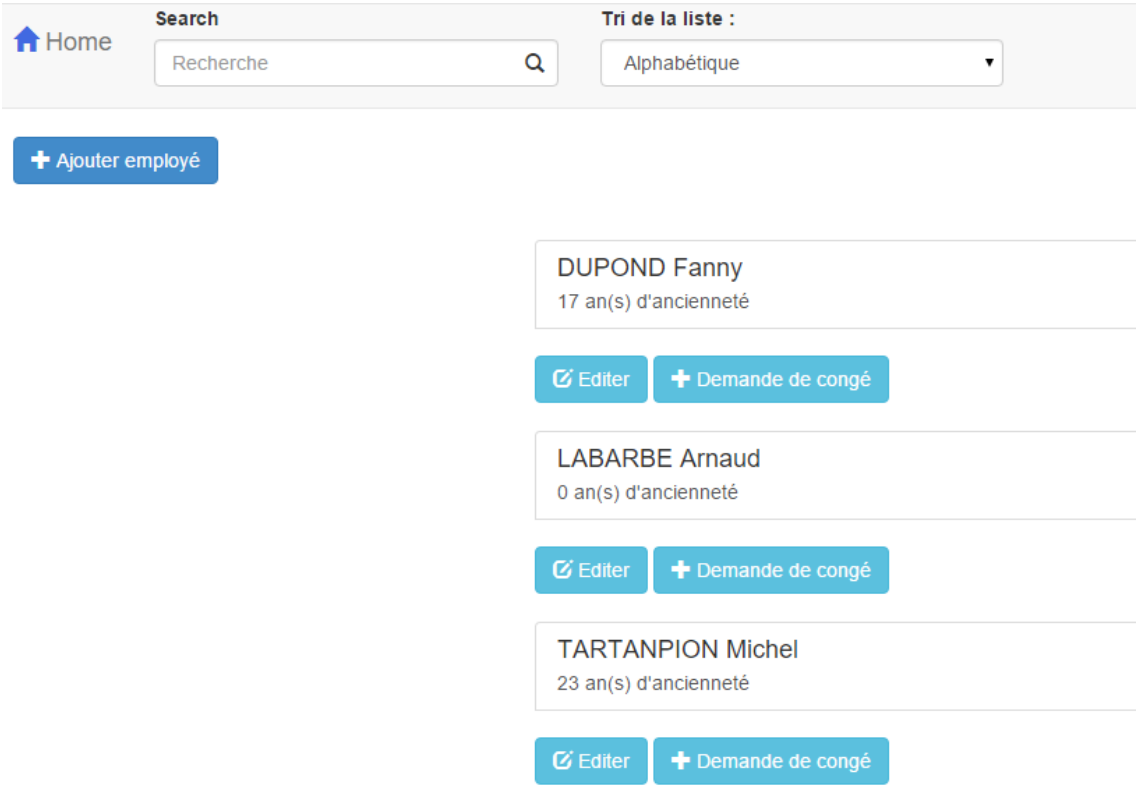
5.4.4 partials

Toutes les vues de mon application se trouvent dans ce dossier sous la forme de fichiers HTML.

5.5 Interface utilisateur

5.5.1 Home page

Figure 10 : index de l'application



Page d'accueil de l'application

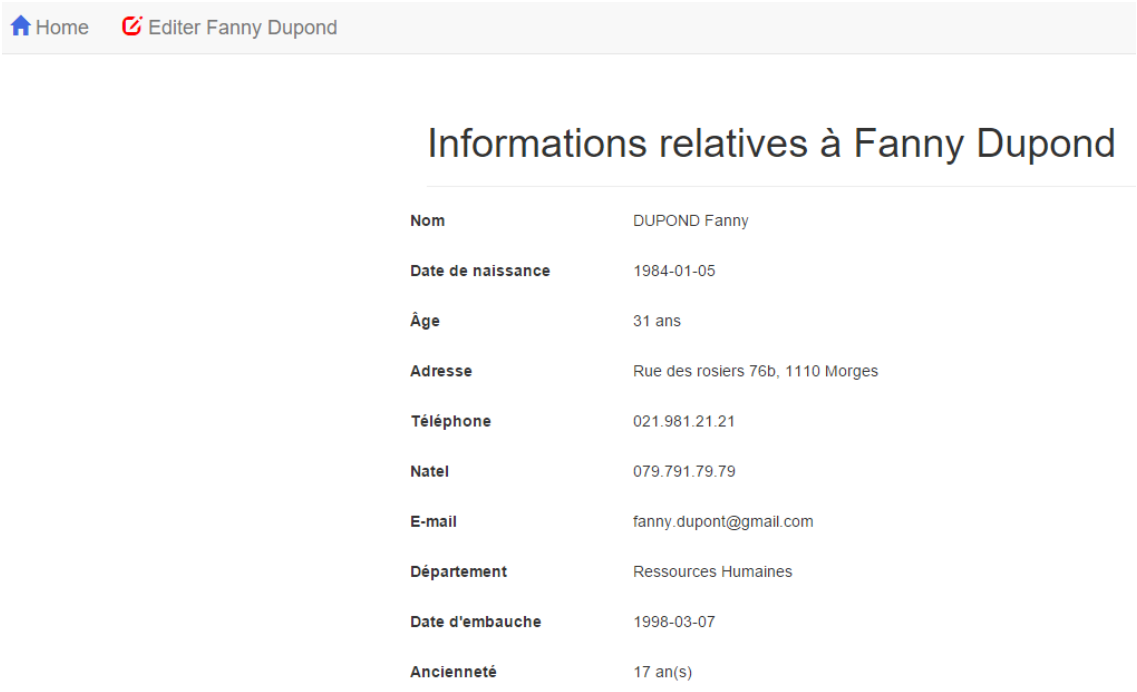
5.5.2 Page d'édition

Figure 11 : Page d'édition d'un employé

Page d'édition de l'application

5.5.3 Page d'informations d'un employé

Figure 12 : Informations d'un employé



Page d'un employé de l'application

5.5.4 Page de demande de congés

Figure 13 : Demande de congés

Page de demande de congés de l'application

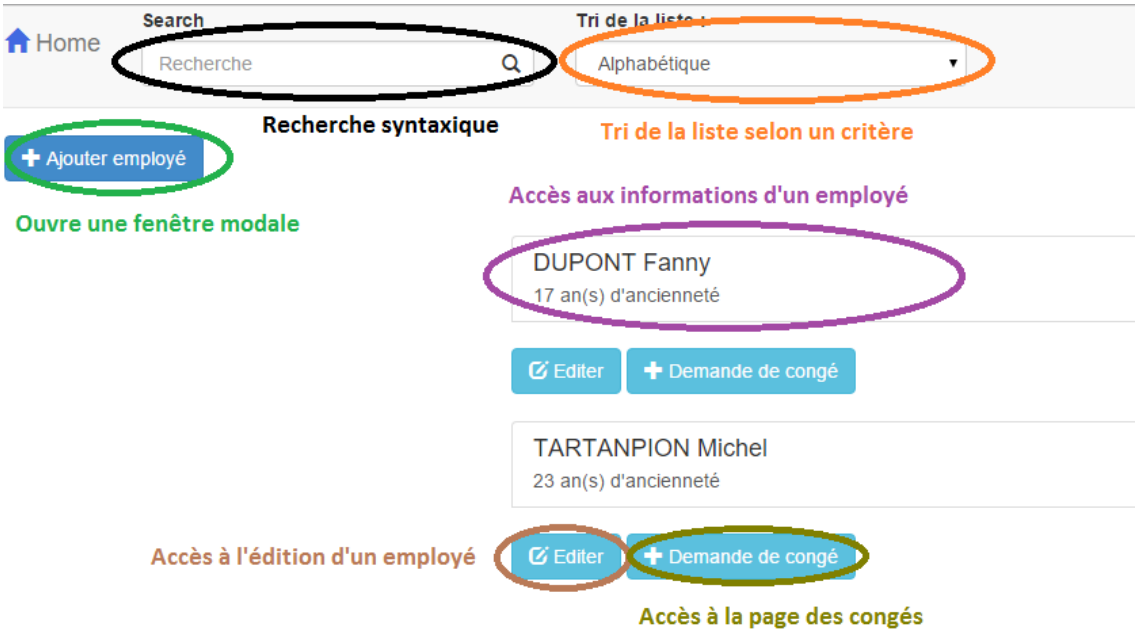
6. Analyse du prototype et expérience de développement

6.1 Les fonctionnalités du prototype

6.1.1 La page d'accueil

La première page à laquelle auront accès les utilisateurs est la page d'accueil. Celles-ci regroupent déjà quelques fonctionnalités.

Figure 14 : Fonctionnalités de la page d'accueil



Page d'accueil de l'application

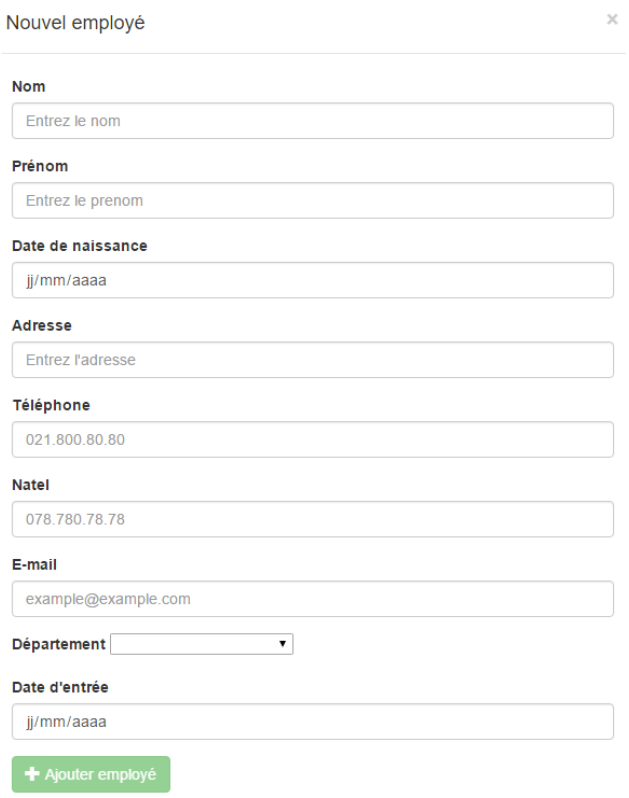
La recherche syntaxique ne s'effectue pas sur un champ prédéterminé tel que le nom ou le prénom, mais elle s'effectue à travers tous les champs du modèle de données. Cela veut dire que dans le champ de recherche, je peux, en tapant 1980 par exemple, trouver toutes les personnes nées en 1980, ou bien encore rechercher toutes les personnes habitant à Lausanne.

Ensuite, le tri de la liste s'effectue ici selon un critère prédéterminé par le développeur, en l'occurrence pour mon application, je propose 3 tris : par ordre alphabétique, par date de naissance et par ancienneté dans l'entreprise. Par défaut, le tri est alphabétique sur le nom.

L'ajout d'un employé se fait par l'intermédiaire d'une fenêtre modal qui s'ouvre suite à l'appui sur le bouton Ajouter un employé. La fenêtre modale comprend un formulaire complet permettant de créer un employé avec toutes les informations nécessaires pour la base de données. Le formulaire doit être valide pour pouvoir enregistrer le nouvel employé dans la base, sinon le bouton d'enregistrement ne sera pas disponible pour l'utilisateur.

Petite particularité pour les numéros de téléphone et de natel, j'ai rajouté deux directives d'input propre à AngularJS, à savoir « *ng-pattern* » qui permet de définir l'input attendu par le champ et « *maxlength* », qui définit la longueur maximale du champ. Dans le cas des champs de numéros de téléphone, pour que le champ soit valide, il est attendu le format : 3 digits, séparateur, 3 digits, séparateur, 2 digits, séparateur, 2 digits.

Figure 15 : Formulaire de création d'employé



Formulaire d'ajout d'un employé

6.1.2 La page d'édition

Lorsque l'utilisateur clique sur le bouton Editer d'un employé de la liste, il se retrouve alors sur une page (voir Figure 11 : Page d'édition d'un employé) contenant un formulaire et trois boutons : Cancel, Mettre à jour et Supprimer.

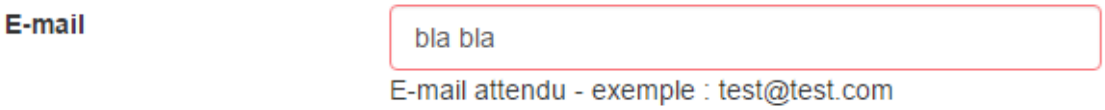
Le formulaire est rempli automatiquement avec toutes les informations de l'employé que l'on peut modifier. Cette page intègre aussi la validation de formulaire par AngularJS. En effet, les champs sont déjà pré remplis, or si un champ se retrouve vide suite à une manipulation de l'utilisateur, le bouton Mettre à jour se retrouve inactif et une alerte sur le champ vide apparaîtra indiquant que celui-ci doit être rempli.

Figure 16 : Validation du formulaire d'édition

Page d'édition de l'application

AngularJS peut également valider les champs de type HTML5 et ainsi détecter si le champ est rempli correctement avec le format attendu, comme c'est le cas avec l'e-mail dans cette page.

Figure 17 : Validation de l'e-mail



Page d'édition de l'application

De plus, comme on peut le remarquer, quand un champ n'est pas correctement rempli les contours de celui-ci deviennent rouges, tandis que lorsqu'il est valide, ils deviennent verts.

Le bouton Mettre à jour, modifie les valeurs de l'employé actuel dans la base de données avec celles présentent dans le formulaire. Une fois les modifications effectuées, l'application retourne à la page d'accueil.

Le bouton Cancel permet un simple retour vers la page d'accueil, tandis que le bouton Supprimer permet d'effacer l'employé actuel de la base de données et effectue un retour à la page d'accueil.

6.1.3 La page de demande de congés

Sur cette page (voir Figure 13 : Demande de congés), l'utilisateur peut ajouter des congés à un employé mais également des jours de maladie. Ici aussi, on a la mise en place d'un formulaire avec validation comme pour la page d'édition d'un employé. Une vérification des données est cependant effectuée, pour que la date de départ ne soit pas inférieure à celle d'arrivée mais aussi, lorsqu'on sélectionne des vacances, que l'employé a assez de jours restants pour prendre ses vacances.

Une fois les jours ajoutés, la mise à jour est appliquée dans la base mais également dans le modèle de données. En effet, les jours sont automatiquement ajoutés ou soustraits le cas échéant, à notre vue, le tout sans rafraîchir la page.

6.2 Import de l'index

L'index de mon application, regroupe toutes les librairies et feuilles de styles que j'ai utilisé pour le développement de mon prototype. Je vais détailler chacun d'elle pour comprendre l'utilité de chacune.

Figure 18 : Import des librairies et feuilles de style dans l'index

```
<!-- Chargement des feuilles de style -->
<link rel="stylesheet" href="bower_components/bootstrap/dist/css/bootstrap.css">
<link rel="stylesheet" type="text/css" href="bower_components/sweetalert/dist/sweetalert.css">
<link rel="stylesheet" type="text/css" href="bower_components/jquery-ui/themes/redmond/jquery-ui.css">
<!-- Chargement des librairies -->
<script src="bower_components/modernizr/modernizr.js"></script>
<script src="bower_components/jquery/dist/jquery.js"></script>
<script src="bower_components/jquery-ui/jquery-ui.js"></script>
<script src="bower_components/angular/angular.js"></script>
<script src="bower_components/angular-route/angular-route.js"></script>
<script src="bower_components/sweetalert/dist/sweetalert.min.js"></script>
<script src="bower_components/angular-sweetalert/SweetAlert.min.js"></script>
<script src="bower_components/bootstrap/dist/js/bootstrap.js"></script>
<!-- Chargement des scripts personnalisés pour AngularJS -->
<script src="js/services.js"></script>
<script src="js/controller.js"></script>
<script src="js/directives.js"></script>
<script src="js/app.js"></script>
```

index.html de l'application

6.2.1 Librairies et feuilles de style

Tout d'abord, les feuilles de style que j'ai utilisé sont celles de Bootstrap, SweetAlert et JQueryUI. Bootstrap a été utilisé pour le rendu visuel des pages HTML à l'utilisateur. La feuille de style SweetAlert est quant à elle utile pour les messages d'alertes générés par la librairie SweetAlert sur laquelle je reviendrai. La feuille de style de JQueryUI sert quant à elle au calendrier pour les navigateurs ne supportant pas le HTML5

Ensuite, viennent les librairies, dont le premier import, le plus évident est le Framework AngularJS. Pour les routes de mon application, j'ai également dû importer un module supplémentaire au Framework, à savoir angular-route.

Pour l'affichage des alertes à l'utilisateur, j'ai utilisé la librairie SweetAlert. Elle permet de styliser les messages d'alerte afin de les rendre plus agréables que ceux intégrés aux navigateurs. Pour que SweetAlert soit compatible avec AngularJS, il faut télécharger la librairie spécialement développé pour le Framework, appelé ngSweetAlert, mais il faut également importer la librairie de base. C'est pour cela que mon index contient deux imports de SweetAlert.

A ma grande surprise, le HTML5 n'est pas encore compatible sur tous les navigateurs. C'est en déployant mon prototype sur mon serveur WEB et en demandant à ma famille de tester l'application que je me suis rendu compte que les inputs de type date, ne fonctionnaient pas sur Firefox. De ce fait, il a fallu trouver une solution, et il s'agit de la librairie Modernizr.

Vient ensuite la librairie Boostrap pour la gestion de ma fenêtre modale qui permet d'ajouter un employé à la base de données.

6.2.2 Les scripts personnalisés

Les imports restants dans l'index, à savoir services, controller, directives et app sont des scripts personnalisés pour le Framework AngularJS. Les noms sont également des standards que l'on peut retrouver dans les applications développées avec le Framework.

Le script app regroupe toutes les routes de mon application mais c'est également ici que l'on retrouve dans la déclaration du module toutes les dépendances de mon application, que ce soit les contrôleurs, les directives, les services, etc. C'est la variable de ce module qui fait office d'application dans l'index afin qu'AngularJS puisse interpréter cette page en tant qu'application. Toutes les routes que j'ai créées sont liés à l'identifiant des employés. En effet, c'est le seul champ qui peut être unique dans la table.

Selon les standards du Framework, toutes les différentes vues de mon application sont regroupées dans un dossier appelé partials.

Le script controller regroupe tous les contrôleurs de mon application ainsi qu'un filtre. Etant donné qu'avec les routes je peux définir un contrôleur pour chaque route, j'ai donc pour mon application quatre contrôleurs distincts (un par vue). Grâce au routing, en fonction de la vue de l'utilisateur, l'accès au bon contrôleur se fait automatiquement.

Les services que je propose dans mon application sont tous regroupés dans le script services. C'est dans celui-ci que j'effectue tous les accès à la base, mais j'ai également

crée deux services : un qui me permet de retourner une date pour l'insertion dans la base et un autre qui compte le nombre de jours ouvrés entre deux dates (samedi et dimanche étant exclus).

Enfin le dernier script, directives, me permet de gérer ma fenêtre modale et de l'afficher avec des éléments supplémentaires et pas uniquement le formulaire.

6.3 Expérience de développement

Au départ, je souhaitais tout avoir sur une seule page, comme le recommande le Framework. C'est-à-dire que je pensais jouer avec le fait de cacher ou montrer des éléments en fonction de l'interaction de l'utilisateur, mais cela se révéla vite compliqué à mettre en place. Le code de la page HTML devenait trop grand et difficilement lisible, même pour moi qui l'avait créée, j'avais du mal à me repérer et à trouver les éléments, le contrôleur était, de fait, dans le même cas.

Je me suis donc rappelé que le tutoriel officiel proposait d'avoir des vues différentes qui s'intègrent parfaitement dans l'index. J'ai donc mis en place le routing et tous les éléments que comptais cacher ou montrer dans des templates HTML. J'ai divisé mon contrôleur en plusieurs et cela a grandement facilité la maintenance et le développement de l'application.

En effet, chaque fois que j'implantais quelque chose de nouveau, il était plus facile de chercher d'où provenait l'erreur. Je savais où chercher précisément, que ce soit dans le code du contrôleur ou celui de l'HTML.

Un des problèmes que j'ai le plus rencontré, c'est les injections, soit parce qu'elles sont mal placés, soit parce qu'elles sont fausses ou soit parce qu'elles ne sont pas correctement écrites.

Typiquement, les contrôleurs en l'état actuel comprennent énormément d'injection car il y a les services, les directives AngularJS et les libraires (SweetAlert notamment) qu'il faut correctement injecter afin de pouvoir les utiliser. Il faut aussi veiller à bien déclarer tous les scripts personnalisés au niveau du script de routing (app.js). Il faut y injecter notre script de contrôleurs, celui des directives et des services.

Le service http proposé par AngularJS a également été difficile à appréhender. En effet, au départ, toutes les requêtes http que j'avais implantées s'effectuaient avec la méthode GET, que ce soit pour des UPDATE, des INSERTS, des DELETE ou des GET. Or, le service http d'AngularJS étant RESTful cela n'était pas du tout dans les standards de REST de tout effectuer avec cette unique méthode.

J'ai donc dû adapter mes requêtes pour être aux normes des standards REST en utilisant les services suivants :

- http.get permet de récupérer des informations depuis la base
- http.post permet d'insérer dans la base
- http.put met à jour la base de données
- http.delete efface une entrée de la base

Une fois ces modifications effectuées, il a donc fallu que j'adapte les fichiers PHP en conséquence puisque pour certaines méthodes, ce sont des données au format JSON qui sont envoyées et qu'il faut pouvoir interpréter en PHP. Le Framework travaillant avec des données au format JSON, lorsqu'on effectue un GET dans la base, il ne faut pas oublier d'encoder celles-ci au format adéquat.

La création de services a également rendu mon code plus lisible et facile pour la maintenance, puisqu'à la base je faisais tout depuis mon contrôleur. Il me permet de regrouper tous mes accès à la base mais contient aussi des services qui me permettent de manipuler des données. Dans le cadre des grosses applications ou d'applications qui s'agrandiront, certains des services seront utilisés plusieurs fois dans les contrôleurs, on évite ainsi la duplication de code. Si un morceau de code doit être répété, il vaut alors mieux en faire un service.

Enfin, je dois dire que malgré les problèmes que j'ai rencontrés ou les difficultés de mise en place de certaines implantations, j'ai toujours réussi à trouver la solution au travers de recherche Google. Le côté communautaire du Framework semble parfaitement fonctionner. Les recherches effectuées m'ont le plus souvent amené vers des sujets sur le forum de Stack Oveflow dont les réponses sont souvent accompagnées d'exemples concrets sur des sites collaboratifs de JavaScript (JS Bin, JSFiddle, Plunker) et très bien expliqués. Les utilisateurs peuvent même valider une réponse en tant que solution pour les futurs développeurs rencontrant le même genre de problème.

7. Conclusion

7.1 Expérience personnelle

Avant ce travail, dans le cadre de mes études, j'ai eu l'occasion de travailler avec un seul Framework JavaScript, à savoir Highcharts pour le compte d'un projet qu'Expedia Genève avait confié à la HEG. C'était également le seul projet orienté WEB auquel je me suis confronté.

De fait, lorsque je me suis vu confier ce travail j'étais très enthousiaste à l'idée de découvrir un Framework qui m'était alors totalement inconnu. C'est donc avec surprise, que j'ai découvert la popularité de celui-ci, par la quantité d'articles et de tutoriels disponibles.

Tout au long de mes recherches et du développement de mon prototype, trouver des informations ou des solutions à un problème de développement n'ont jamais été un problème, même pour une personne totalement novice dans le Framework.

Mais grâce au développement d'une application dans le cadre de ce travail, j'ai pu mettre en œuvre tout ce que j'avais pu apprendre en suivant les tutoriels en ligne ou les cours théoriques, tout en enrichissant mon application de certaines fonctionnalités totalement nouvelles pour moi.

Je peux aujourd'hui affirmer être capable de développer une application avec le Framework AngularJS, mes connaissances pouvant cependant être très certainement approfondies, mais les bases sont d'ores et déjà acquises pour construire une application de bout en bout.

D'un point de vue personnel, ce travail a été très enrichissant puisqu'il m'a non seulement permis d'apprendre à manipuler un Framework que l'on peut qualifier d'avenir pour le développement des applications WEB, mais il m'a également permis de prendre conscience des changements que l'utilisation des smartphones allait engendrer sur le développement des applications.

7.2 Réponse à la problématique

Lorsque l'on a déjà une application existante, n'utilisant pas le Framework AngularJS, que l'on souhaite modifier ou adapter, il est tout à fait concevable de retravailler le code source pour l'adapter à AngularJS. En effet, le Framework étant très structuré, il est facile de s'y retrouver et de mettre les éléments existants au bon endroit dans l'architecture de l'application.

Pour l'utilisation d'AngularJS dans une application, il faut prendre en compte sa compatibilité avec les navigateurs internet mais aussi le fait que le rendu des pages sera plus lent, étant donné que le navigateur fait une grande partie du travail pour le rendu à l'utilisateur. Son utilisation dépend donc de l'environnement dans lequel l'application sera mise en place.

AngularJS peut très bien s'adapter à des projets d'envergure. En effet, du fait du peu d'interaction du Framework avec le serveur, il engendre un trafic très léger sur le réseau, ce qui permet donc à beaucoup d'utilisateurs de se servir de l'application en même temps. La clé pour les projets importants avec AngularJS, c'est la structuration de l'application avec des noms de fichiers et de dossiers explicites afin de retrouver le code facilement.

Pour conclure, je dirai qu'aujourd'hui, il n'y a aucun frein à l'utilisation du Framework AngularJS et ce dans n'importe quel type d'application. L'implication de la communauté, les nouveautés apportées, son évolution (version 2.0) et le support de Google en font un Framework que je recommanderai et que j'utiliserai pour le développement des applications WEB, pour autant que l'environnement me le permette.

Cependant, ce Framework étant récent, tous les développeurs n'ont pas encore les compétences et il n'est peut-être qu'un effet de mode qui ne durera pas aussi longtemps que l'architecture client-serveur, qui est encore loin d'être obsolète.

Bibliographie

FAQ de AngularJS [Consulté le 1er juillet 2015] Disponible à l'adresse : https://docs.AngularJS.org/misc/faq

Guide du développeur AngularJS [Consulté le 3 juillet 2015] Disponible à l'adresse : https://docs.AngularJS.org/guide

Page Wikipedia anglaise de AngularJS [Consulté le 3 juillet 2015] Disponible à l'adresse suivante : https://en.wikipedia.org/wiki/AngularJS

Page Wikipedia anglaise de Client-server [Consulté le 6 juillet 2015] Disponible à l'adresse : https://simple.wikipedia.org/wiki/Client-server

Page Wikipedia française de Client-serveur [Consulté le 6 juillet 2015] Disponible à l'adresse : https://fr.wikipedia.org/wiki/Client-serveur

Page Wikipedia française de l'architecture pair à pair [Consulté le 7 juillet 2015] Disponible à l'adresse : https://fr.wikipedia.org/wiki/Pair_%C3%A0_pair

Page Wikipedia française de l'architecture trois tiers [Consulté le 9 juillet 2015] Disponible à l'adresse suivante : https://fr.wikipedia.org/wiki/Architecture_trois_tiers

La théorie des réseaux locaux et étendus [Consulté le 10 juillet 2015] Disponible à l'adresse : http://hautrive.developpez.com/reseaux/

La notion de data-binding [Consulté le 13 juillet 2015] Disponible à l'adresse suivante : https://openclassrooms.com/courses/developpez-vos-applications-web-avec-AngularJS/la-notion-de-data-binding

MVC in AngularJS [Consulté le 14 juillet 2015] Disponible à l'adresse : http://mrbool.com/mvc-in-AngularJS/28962

Page Wikipedia anglaise de l'injection de dépendance [Consulté le 14 juillet 2015] Disponible à l'adresse : https://en.wikipedia.org/wiki/Dependency_injection

AngularJS : le Framework de Google au crible [Consulté le 14 juillet 2015] Disponible à l'adresse : http://www.journaldunet.com/developpeur/outils/AngularJS-js.shtml

An overview of AngularJS for managers [Consulté le 20 juillet 2015] Disponible à l'adresse suivante : http://andrewaustin.com/an-overview-of-angularjs-for-managers/

Mais qui a créé l'appli Angular ? [Consulté le 21 juillet 2015] Disponible à l'adresse suivante : http://www.cyber-pc.fr/dossier/mais-qui-a-cree-l-appli-angular.html

AngularJS vs Backbone.js vs Ember.js [Consulté le 23 juillet 2015] Disponible à l'adresse suivante : https://www.airpair.com/js/javascript-framework-comparison

Wappalyzer pour AngularJS [Consulté le 24 juillet 2015] Disponible à l'adresse suivante : https://wappalyzer.com/applications/angularjs

Wappalyzer pour Ember.js [Consulté le 24 juillet 2015] Disponible à l'adresse suivante : https://wappalyzer.com/applications/emberjs

Wappalyser pour Backbone.js [Consulté le 24 juillet 2015] Disponible à l'adresse suivante : https://wappalyzer.com/applications/backbonejs

Built with AngularJS [Consulté le 24 juillet 2015] Disponible à l'adresse suivante : https://builtwith.angularjs.org/

Projects and Companies using Backbone [Consulté le 24 juillet 2015] Disponible à l'adresse suivante : https://github.com/jashkenas/backbone/wiki/Projects-and-Companies-using-Backbone

See who's using Ember.js [Consulté le 24 juillet 2015] Disponible à l'adresse suivante : http://emberjs.com/ember-users/

Miško Hevery and Brad Green – Keynote – NG-Cong 2014 [Consulté le 27 juillet 2015] Disponible à l'adresse suivante : https://www.youtube.com/watch?v=r1A1VR0iblQ

Mobile now exceeds PC : The biggest shift since the Internet began [Consulté le 29 juillet 2015] – Disponible à l'adresse suivante : http://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began

Mobile Internet time now exceeds PC – comScore [Consulté le 29 juillet 2015] – Disponible à l'adresse suivante : http://marketingland.com/mobile-internet-time-now-exceeds-pc-time-online-comscore-56492

Le temps passé sur le Web mobile dépasse celui sur PC aux Etats-Unis [Consulté le 29 juillet 2015] – Disponible à l'adresse suivante : http://www.journaldunet.com/ebusiness/internet-mobile/le-temps-passe-sur-mobile-depasse-celui-sur-pc-aux-etats-unis.shtml

Mobile apps overtake PC internet usage in U.S [Consulté le 29 juillet 2015] – Disponible à l'adresse suivante : http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/

Le navigateur comme client riche, l'exemple AngularJS [Consulté le 30 juillet 2015] – Disponible à l'adresse suivante : http://www.improve-technologies.com/2014/10/03/le-navigateur-comme-client-riche-lexemple-dangularjs/

What is AngularJS 2.0 all about ? – [Consulté le 1er août 2015] – Disponible à l'adresse suivante : http://ng-learn.org/2014/03/AngularJS-2-Status-Preview/#what_is_angularjs_20_all_about

Quelques détails sur Angular 2.0 – [Consulté le 1er août 2015] – Disponible à l'adresse suivante : http://www.infoq.com/fr/news/2014/11/angular-2-atscript

AngularJS, les développeurs dans le trouble au sujet de la version 2.0 – [Consulté le 1er août 2015] – Disponible à l'adresse suivante : http://web.developpez.com/actu/80230/AngularJS-les-developpeurs-dans-le-trouble-au-sujet-de-la-version-2-0-quel-va-etre-l-avenir-du-framework-JavaScript-de-Google/

No 1 position in Google gets 33% of search traffic – [Consulté le 10 août 2015] – Disponible à l'adresse suivante : http://searchenginewatch.com/sew/study/2276184/no-1-position-in-google-gets-33-of-search-traffic-study

Representational State Transfer – [Consulté le 27 août 2015] – Disponible à l'adresse suivante : https://en.wikipedia.org/wiki/Representational_state_transfer

CRUD – [Consulté le 27 août 2015] – Disponible à l'adresse suivante : https://fr.wikipedia.org/wiki/CRUD

SweetAlert – [Consulté le 24 août 2015] – Disponible à l'adresse suivante : http://t4t5.github.io/sweetalert/

ngSweetAlert – [Consulté le 24 août 2015] – Disponible à l'adresse suivante : https://github.com/oitozero/ngSweetAlert

Webshim – [Consulté le 26 août 2015] – Disponible à l'adresse suivante : https://github.com/aFarkas/webshim

Modernizr – [Consulté le 3 septembre 2015] – Disponible à l'adresse suivante : http://modernizr.com/

AngularJS est-il fait pour moi ? Les bases – [Consulté le 30 septembre 2015] – Disponible à l'adresse suivante : http://blog.kaliop.com/blog/2014/01/13/angularjs-presentation/

Server vs client side rendering (AngularJS vs server side MVC) – [Consulté le 30 septembre 2015] – Disponible à l'adresse suivante : http://technologyconversations.com/2014/07/10/server-vs-client-side-rendering-angularjs-vs-server-side-mvc/

Angular Structure : Refactoring for growth – [Consulté le 30 septembre 2015] – Disponible à l'adresse suivante : http://www.johnpapa.net/angular-growth-structure/