



MASTER SCIENCE ET TECHNOLOGIE DU LOGICIEL

RAPPORT PROJET DAR
2016

Find Your Flat

CHAWKI Faïssal

MAHAMAT Issa

WAHBI Bilal

Sommaire :

Guide Utilisateur	1
1 Description de l'application	1
2 Manuel d'utilisation	1
3 Liste des fonctionnalités	4
3.1 Profil	4
3.2 Services autour de l'adresse	5
3.3 Social	5
 Architecture	 7
1 Serveur	7
1.1 Hébergement du serveur	7
1.2 Langage Serveur	8
1.3 Persistance des données	8
1.4 API	9
2 Client	11
2.1 Technologies utilisées	11
3 Travail Collaboratif	11
 Remarques générales et limitations	 11
 Perspectives d'évolutions	 12
 Conclusion	 13

Guide Utilisateur

1 Description de l'application

Notre idée consiste en une application dédiée à la recherche de logements en Île-de-France en faisant participer un maximum de protagonistes qui vont être capables de noter ainsi que commenter des lieux (renseignés par des adresses) pour savoir quelle est la tendance moyenne pour une adresse donnée. De plus l'application permet pour une adresse ainsi qu'un rayon donné de voir où sont situés différents services indispensables pour le confort de tous (poste de police, agence postale, hôpital, pharmacie, écoles pré-bac/post-bac, salle de sport, réseau de transport ferré). Cette application web est aussi utile pour une personne résidente qui ne connaîtrait pas très bien son quartier et souhaiterait voir facilement les différents services disponibles près de chez lui.

Cette idée part d'un réel constat puisqu'il n'est pas toujours évident de se renseigner correctement sur un lieu lorsque l'on cherche un appartement que ce soit pour y séjourner brièvement (Airbnb) ou bien de manière permanente. Comme mentionné précédemment, cette application est dédiée à n'importe quelle personne désireuse de se renseigner sur un lieu qui se trouve dans toute l'Île-de-France, la dimension sociale associée à cette application est immédiate. En effet, les différents utilisateurs peuvent échanger en communiquant par le biais de chats, en postant des commentaires ou bien en notant une adresse.

L'utilisateur peut donc voir les différents services disponibles autour d'une adresse et se renseigner au sujet des commentaires/notes postés par d'autres utilisateurs pour l'adresse ainsi que les adresses aux alentours afin d'obtenir un aperçu sur le quartier. Pour plus de précisions l'utilisateur peut entrer en contact avec les autres utilisateurs afin de les questionner, discuter et pourquoi pas créer une ou plusieurs nouvelle(s) amitié(s).

2 Manuel d'utilisation

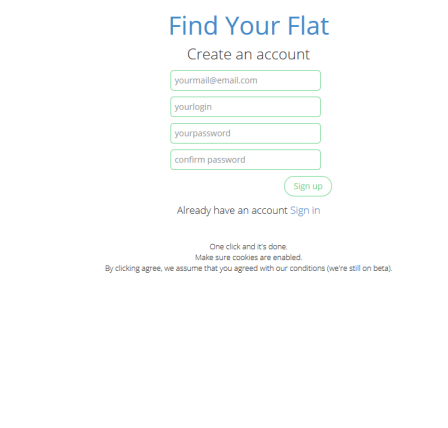
L'interface utilisateur est simple, sobre et intuitive, il est tout à fait possible de l'utiliser sans explication. Mais dans le cadre du rapport de projet DAR, ces explications sont toutefois nécessaires.

L'application peut être utilisée de 2 manières, en étant connecté ou non. En étant connecté l'utilisateur aura en plus la possibilité de noter et commenter une adresse, utiliser la messagerie instantanée, gérer son compte et se déconnecter.

L'application se décompose en 5 pages principales. Il est possible d'accéder

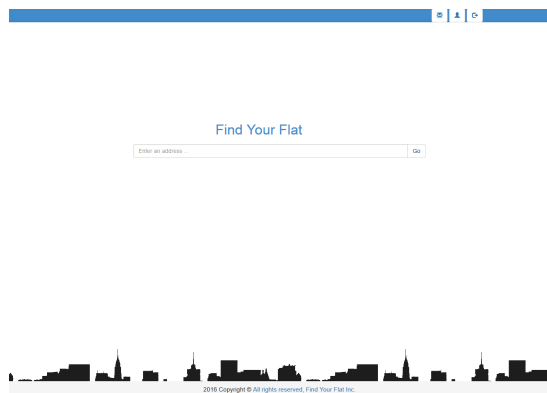
à ces différents services en utilisant la barre de menu situé en haut (à noter qu'elle sera différente si vous êtes connecté ou non), cliquer sur le logo redirige vers la page home.

- Les pages de connexion/inscription, qui vont permettre de se connecter ou s'inscrire.



The screenshot shows the 'Find Your Flat' registration page. At the top, the title 'Find Your Flat' is in blue. Below it, the text 'Create an account' is centered. There are four input fields: 'yourmail@email.com', 'yourlogin', 'yourpassword', and 'confirm password'. A green 'Sign up' button is to the right of the 'confirm password' field. Below the button, the text 'Already have an account Sign in' is centered. At the bottom, there is a small disclaimer: 'One click and it's done. Make sure cookies are enabled. By clicking agree, we assume that you agreed with our conditions (we're still on beta).'

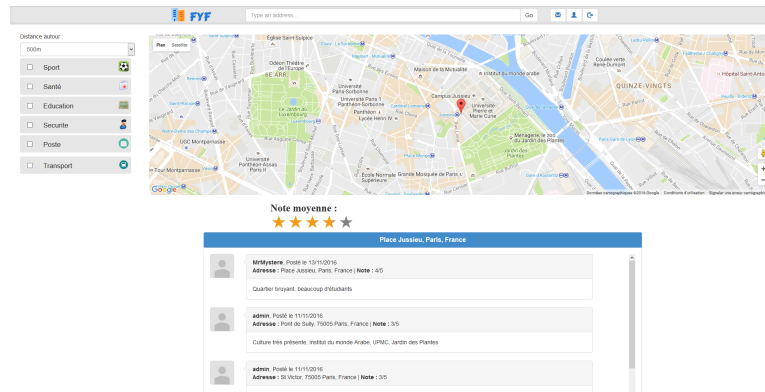
- L'index, qui va simplement permettre de chercher une adresse (similaire à la page d'accueil de Google).



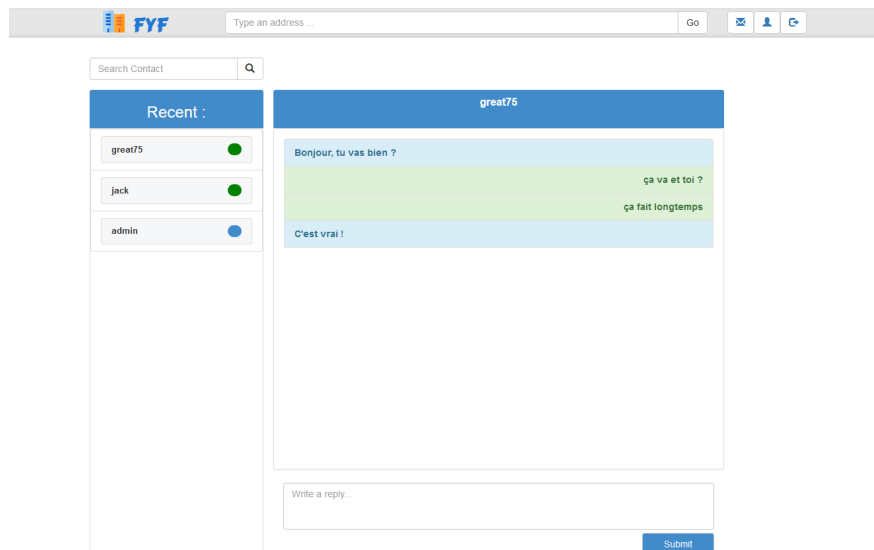
The screenshot shows the 'Find Your Flat' search page. At the top, there is a blue header bar with the text 'Find Your Flat' and a search icon. Below the header, the text 'Find Your Flat' is centered. There is a search input field with the placeholder text 'Enter an address' and a 'Go' button. At the bottom, there is a silhouette of a city skyline and the text '2016 Copyright © All rights reserved, Find Your Flat Inc.'

- Une page principale où il est possible de voir l'adresse indiquée sur une Google Maps, sélectionner la distance aux alentours qui nous intéresse,

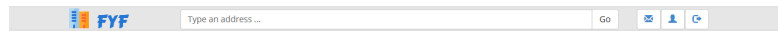
sélectionner des informations à afficher sur la carte (école, hôpital, etc. . .), lire les notes et commentaires, voir la note moyenne, commenter et noter.



- Une messagerie instantanée qui permet de discuter avec les autres utilisateurs de l'application.



- Une page profil qui permet de voir son login ainsi que son adresse mail, changer son mot de passe et supprimer son compte.



Find Your Flat

Your login : paris75

Your mail : paris75@mail.com

Change my password

yourOldPassWord

yourNewPassWord

retypeYourNewPassWord

Modify

Delete Account

password

Delete

3 Liste des fonctionnalités

Diverses fonctionnalités ont été développées à la suite de ce projet, dont voici leurs énumérations ainsi que leurs descriptions :

3.1 Profil

Création d'un compte :

Pour créer un compte il suffit de remplir le formulaire. Les champs à renseigner sont l'adresse mail, l'identifiant (login) et un mot de passe (de six caractères minimum). Il n'est pas possible de créer un compte ayant un login ou une adresse mail déjà existante dans la base de données.

Connexion/Déconnexion :

Le principe est simple, l'utilisateur doit saisir son identifiant ainsi que son mot de passe pour pouvoir s'authentifier. Une fois connecté il peut se déconnecter en cliquant sur le bouton fait pour cet effet dans la barre de menu. De plus un utilisateur connecté peut utiliser plus de fonctionnalité.

Récupération informations utilisateurs :

Un utilisateur connecté peut accéder aux informations relatives à la création de son compte. Ces informations sont son login ainsi que son adresse mail.

Oubli du mot de passe :

Lorsque l'utilisateur est déconnecté, il peut choisir de réinitialiser son mot de passe auquel cas il devra saisir son login et son adresse mail pour recevoir un nouveau mot de passe sur sa boîte mail.

Changement du mot de passe :

Une fois connecté, en saisissant l'ancien mot de passe puis deux fois le nouveau, il est possible de changer son mot de passe.

Suppression d'un compte :

Suppression d'un compte utilisateur en renseignant son mot de passe lorsque celui-ci est connecté. Lorsqu'un compte est supprimé tous les commentaires/messages privés associés à ce compte sont supprimés.

3.2 Services autour de l'adresse

Recherche selon le service souhaité :

Pour un type d'information donné tel que les transports, l'éducation, la santé, la sécurité, les équipements sportifs ou encore les bureaux de postes ainsi qu'une adresse de recherche et une distance aux alentours, s'affichera sur la carte de l'application les informations voulues. Ces informations seront pointées à l'aide de marqueurs sur la carte. La mesure de l'entourage est paramétrable en utilisant le champs en haut des boutons relatifs aux services disponibles (distance autour).

3.3 Social

Commenter une adresse :

Un utilisateur connecté peut laisser un commentaire pour l'adresse précédemment recherchée.

Noter une adresse :

Un utilisateur connecté peut aussi noter une adresse. Cette note de un à cinq représente l'appréciation de l'adresse et de ses alentours.

Note : lorsqu'un utilisateur souhaite noter/commenter une adresse qu'il a déjà noté/commenté alors cela n'ajoute pas un nouveau commentaire/note mais met à jour l'ancien.

Lecture des notes/commentaires :

Quand une adresse est recherchée, ce service collecte les commentaires et les notes des utilisateurs qui sont associées à cette adresse (en fonction du rayon indiqué par l'utilisateur).

Note moyenne :

Cette note est attribuée à une adresse en récupérant les notes des adresses aux alentours (selon la distance indiquée par l'utilisateur) ainsi que les siennes et en faisant la moyenne de celles-ci.

Messagerie instantanée :

Une messagerie instantanée est disponible et permet à un utilisateur de communiquer avec une personne en particulier. Pour communiquer avec un autre utilisateur il suffit de cliquer sur l'image associée à son commentaire dans l'espace commentaire, ce qui redirige l'utilisateur vers la messagerie avec comme contact l'utilisateur voulu. Il est aussi possible de chercher un utilisateur existant afin de communiquer avec lui. Pour ce faire, un champ de recherche est disponible dans la messagerie et propose une auto-complétion selon les premières lettres entrées dans le champ.

Sur la gauche se trouvent les contacts avec qui l'utilisateur a discuté récemment. Ils sont affichés dans l'ordre des discussions les plus récentes (en haut se trouve le contact avec qui l'on a parlé le plus récemment). La couleur du rond à droite du pseudo de chaque contact indique s'il est connecté ou non. De plus si l'utilisateur n'a pas lu les derniers messages d'une conversation, le nombre de messages non lus s'affiche dans le rond.

Cette messagerie se veut d'être un minimum ergonomique, par exemple lorsqu'il existe beaucoup de message une scrollbar permet de défiler. Dans le cas où la scrollbar est en bas, lorsque de nouveaux messages arrivent dans la fenêtre la scrollbar reste en bas de la fenêtre ce qui permet de suivre la conversation sans devoir descendre à chaque fois. Si l'on souhaite lire d'anciens messages, que l'on scrolle vers le haut et que l'on reçoit un nouveau message alors dans ce cas la scrollbar reste à sa position.

De plus le fait de pouvoir communiquer avec soi-même est voulu, cette possibilité est présente dans plusieurs messageries disponibles sur le marché et permet de stocker des informations dont l'on souhaite se rappeler.

Technologies et choix effectués

Architecture

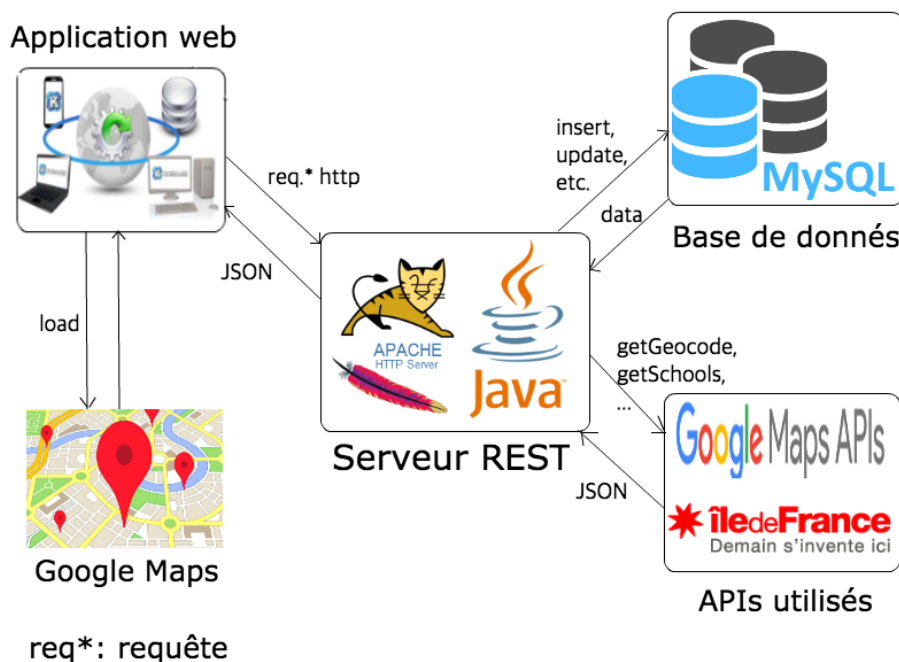


Figure 1: Architecture de l'application

1 Serveur

1.1 Hébergement du serveur

L'hébergement s'effectue sur un serveur privé auprès de l'hébergeur OVH. Ce choix est justifié tout d'abord par le fait que nous possédions déjà un serveur auprès d'OVH donc cela n'incluait pas de frais en plus. Ensuite cela nous a offert une grande liberté puisqu'il était possible de configurer le serveur comme bon nous semblait, sans contrainte et en toute liberté. Nous avons pu utiliser les outils que nous souhaitions sans aucune contrainte. L'hébergement est suffisant

dans le cadre de ce projet qui se limite (pour l'instant ?) à un cadre universitaire, mais dans l'hypothèse d'une mise en production et d'une fréquentation importante du serveur il faudrait bien entendu réfléchir à une infrastructure plus adaptée.

Le serveur HTTP permettant la gestion des servlets est Tomcat. Tomcat est gratuit, facile à installer, et supporte les dernières API Java. C'est un choix idéal dans le cadre de ce projet.

1.2 Langage Serveur

Le langage utilisé pour la partie serveur est le langage Java (imposé de par l'utilisation des servlets). Aucune utilisation de JSP n'a été faite. Un service offert est représenté par une servlet. Les pages web seront générées dynamiquement par le client via des requêtes AJAX.

1.3 Persistance des données

MySQL, une base de données relationnelle, a été utilisée afin de pouvoir stocker. La base de données de l'application est décomposée en 4 tables :

- **users**, [id/mail/login/pw] : Table représentant un utilisateur, les mots de passe sont bien évidemment cryptés avant d'être stockés dans la base de données. Les champs id, mail et login sont uniques.
- **sessions**, [id/session_id/user_id/start] : Table représentant une session autrement dit un utilisateur connecté. user_id est une clé secondaire référençant un user, start la date du début de la session et session_id la valeur du cookie stocké chez le client pour indiquer qu'il est connecté (la valeur est unique pour chaque utilisateur).
- **comments**, [id/user_id/comment/note/lat/lng/adresse/date] : table où sont stockés les commentaires. Le nom de chaque champ est assez explicite. user_id est une clé secondaire référence un user, comment et note peuvent être null dans le cas où l'utilisateur à juste noté ou juste commenté l'adresse. Maintenant pourquoi gardons nous l'adresse si nous avons déjà la latitude et la longitude ? Simplement dans le cas où le commentaire est assez ancien et qu'entre-temps le nom du lieu à changé les utilisateurs seront au courant que le commentaire correspond à l'ancienne adresse (par exemple si une personne commente une adresse où se trouve un jardin et que quelques années plus tard la zone change de nom et devient une zone industrielle cela permettra de savoir que le commentaire est en réalité obsolète).
- **messages**, [id/id_sender/id_receiver/message/date_send/is_read] : Tables où sont stockés les messages (de la messagerie instantanée) là aussi les noms des champs sont explicites. id_sender, id_receiver sont des clés secondaires qui référencent des users.

À noter que lorsqu'un user est supprimé de la table tous les éléments référencés seront supprimés. Donc lorsqu'un utilisateur supprime son compte, ses commentaires et messages privés sont aussi supprimés.

1.4 API

Différentes API ont été utilisés dans l'élaboration de cette application web.

- Google Maps API : Permet l'affichage d'une carte cartographique, ainsi que l'affichage de marqueurs cliquables représentant les différentes informations demandées par l'utilisateur.
- Google Maps Geocoding API : Permet d'obtenir à partir d'une adresse, la latitude et la longitude correspondante. Cette API permet aussi d'effectuer l'opération inverse.
- API issues d'Open data Île-de-France : Différentes APIs issues d'Open data Île-de-France ont été utilisées afin de récupérer des informations sur les différents services disponible autour d'une adresse (école, agence postale, commissariat, pharmacie, etc...).
Dans le code de l'application une classe Java a été créée afin de faciliter l'utilisation des APIs issues d'open data Île-de-France. Il s'agit d'une classe abstraite nommée "RequeteApiIleDeFrancePattern.java" dont voici un exemple de fonctionnement.

```

public class PosteAPI extends RequeteApiIleDeFrancePattern {

    protected PosteAPI(double latitude, double longitude, double
        distance) {
        super(latitude, longitude, distance);
    }

    @Override
    protected String getExclude() {return "";}
    @Override
    protected String getRefine() {return "";}
    @Override
    protected String[] getArrayOfFacettes() {return null;}
    @Override
    protected String getDataSetName() {
        return
            "liste-des-points-de-contact-du-reseau-postal-dile-de-france";
    }

    @Override
    protected JSONObject getMyJsonObjectFromRecord(JSONObject
        record) throws JSONException {
        JSONObject res = new JSONObject();
        JSONObject fields = record.getJSONObject("fields");
        String latitude = fields.getJSONArray("wgs84").getString(0);
        String longitude = fields.getJSONArray("wgs84").getString(1);
        res.put("latitude", latitude);
        res.put("longitude", longitude);
        res.put("nom",
            StringEscapeUtils.escapeHtml4(fields.getString("caracteristique_du_site")));
        res.put("description",
            StringEscapeUtils.escapeHtml4(fields.getString("libelle_du_site")));
        res.put("type", "poste");
        return res;
    }

    public static JSONArray getPosteJSON(double latitude, double
        longitude, double distance) throws JSONException, Exception{
        PosteAPI p = new PosteAPI(latitude, longitude, distance);
        return p.getResJSON(latitude, longitude, distance);
    }
}

```

Pour utiliser une des apis, il suffit d'étendre la classe `RequeteApiIleDeFrancePattern` et de ré-implémenter les méthodes demandées.

Les facettes correspondent à des champs permettant de filtrer et sont définies par l'api. Ensuite il est possible d'indiquer des règles de filtrage via `getExclude()`

qui permet à partir de facettes d'exclure des résultats de la requête. `getRefine()` permet au contraire de s'intéresser uniquement à certains résultats. `getDataSetName()` doit indiquer le nom de l'api à utiliser. `getMyJsonObjectFromRecord()` est la méthode qui doit filtrer les différents champs (JSON) renvoyés par l'api. Le constructeur prend en paramètre une position géographique ainsi qu'une distance pour la recherche. Enfin il suffit de fournir une méthode statique retournant le résultat de l'appel à l'api, cette méthode se contente de créer un objet correspondant à la classe créée et retourne le résultat.

Ainsi toutes les informations qui seront identiques pour toutes les apis sont encapsulées dans la classe abstraite et dès lors que l'on souhaite modifier une information (par exemple le fuseau horaire) il suffit de modifier cette dernière. Nous pensons que la manière dont nous avons procédé afin de factoriser du code relatif à ces apis mérite d'être présenté dans le rapport.

2 Client

2.1 Technologies utilisées

Le client est très classique et a été conçu en HTML/CSS/JavaScript. Le framework Bootstrap a été utilisé afin de pouvoir obtenir un affichage esthétique et responsive. De plus l'utilisation de Bootstrap permet d'avoir un affichage sur mobile qui reste utilisable et agréable. Le client utilise aussi la bibliothèque JavaScript jQuery qui permet de faciliter certaines actions.

Globalement le client est organisé de cette manière : il existe des squelettes de pages HTML qui seront mis à jour dynamiquement en fonction des interactions utilisateurs via des appels AJAX (aucune utilisation de JSP dans le serveur).

3 Travail Collaboratif

L'outil de travail collaboratif choisi est Git. Celui-ci apporte une grande facilité de par le fait qu'il permet de travailler à plusieurs en simultané sans empiéter sur le travail des autres collaborateurs. Git propose aussi une gestion des versions du projet ce qui nous permettait de revenir sur une ancienne version dans le cas où le projet serait devenu corrompu par la cause de mauvais choix d'implémentation.

Remarques générales et limitations

- L'utilisation d'AJAX permet d'avoir une transparence totale d'un point de vue de l'utilisateur, c'est à dire que lorsqu'il lance une requête, la page ne se

rechargera pas pour récupérer les données que celui-ci demande mais effectue un appel asynchrone c'est à dire en arrière plan pour modifier uniquement la partie de la page qui doit l'être pour ne pas recharger toute une page web inutilement.

- Pour se prémunir de l'attaque "SQL injection" toutes les requêtes à la base de données sont faites en utilisant la classe "PreparedStatement" qui pré-compile les requêtes avec des trous (ces trous seront remplis lorsque l'utilisateur soumet une formulation ou a besoin d'un service). Les mots de passe des utilisateurs ne devant pas être stockés en clair dans la base de données sont cryptés (du côté serveur) avant leur stockage. Le choix de BCrypt est dû au fait que c'est une fonction de hachage très difficile à craquer et peut s'adapter à la puissance croissante des ordinateurs.

- Il existe deux méthodes pour rechercher les commentaires et notes associées à une adresse. La première est naïve puisqu'il s'agit de parcourir toute la base de données des commentaires/notes pour afficher ou non les commentaires/notes de chaque adresse.

On va pouvoir en fonction de la définition de la distance d'un lieu retrouver précisément les informations voulues mais plus le nombre de commentaires et de notes augmentent et plus il sera long de parcourir la base.

La seconde méthode consiste à quadriller l'espace pour créer une base de données relativement petite (en fonction du quadrillage effectué) et pouvoir retrouver directement dans la base dans laquelle se trouve l'adresse tous les commentaires/notes des adresses se trouvant également dans cette base.

Cette méthode est plus rapide que la précédente pour récupérer les informations dont on a besoin mais moins précises quant à la distance des adresses environnantes une adresse donnée.

- Il se peut qu'un grand nombre de commentaires soient disponibles pour une adresse recherchée, le cas échéant, il serait plus judicieux de n'en récupérer qu'une partie définie selon un seuil (un nombre de commentaire à afficher par défaut) et pouvoir charger les suivants par tranche de ce seuil à chaque fois que l'utilisateur scrolle pour en faire défiler plus.

Perspectives d'évolutions

Plusieurs axes d'amélioration sont disponibles afin d'obtenir une application web plus évoluée :

- il serait bon d'utiliser le protocole HTTPS afin d'obtenir une connexion sécurisée.
- ajouter encore plus de services à afficher sur la carte en utilisant de nouvelles API.

- ajouter la possibilité d'afficher sur la carte des services de manière plus précise. Typiquement lorsque l'utilisateur affiche les services liés à la santé, toutes les structures liées à la santé sont affichées (hôpitaux, pharmacie, centre dentaire...), il serait agréable de pouvoir affiner encore plus notre recherche en affichant uniquement les pharmacies par exemple.
- avoir une gestion des profils plus poussée, avec le fait de pouvoir ajouter une image de profil, garder des adresses en favoris, avoir l'historique de ses recherches, bloquer des utilisateurs, etc...
- étendre l'application à d'autres régions (et même à l'international)
- le développement d'une application mobile. Cela serait assez simple à développer du fait que l'on dispose d'une api REST.
- avoir un système de financement (publicité, service payant ?) afin de pouvoir nous rémunérer et améliorer la capacité d'accueil du serveur, en effet il est bien connu que tout travail mérite salaire (même si la satisfaction du travail bien accompli est en soi un salaire).

Conclusion

La partie la plus difficile de ce projet a été de trouver une idée d'application pertinente, utile et qui s'arme de plusieurs API originales pour sortir du contexte type des applications telles que 'Renseignements sur les lignes de métro' ou encore 'Monuments à visiter dans Paris'.

Une fois cet axe directeur trouvé, la seconde difficulté était de répartir le développement des tâches en terme de fonctionnalités. L'écriture des Servlet étant assez 'rébarbatif' la réalisation de celles-ci a été plutôt simple, une Servlet correspondant généralement à un service, nous nous sommes réparti la charge de travail au niveau des Servlet à effectuer suivi de leur liaison à la partie frontend de l'application.

Un point sur lequel il serait intéressant de se pencher pour approfondir la qualité de notre travail consisterait en l'étude et l'intégration d'un algorithme qui propose un trade-off parfait entre précision de la répartition des zones et dont la complexité en temps de recherche des commentaires soit faible.

Finalement ce projet a été pour nous une source d'ouverture à l'ensemble des différentes technologies mises en oeuvre pour produire une application riche et utile.