POLYTECH
ORLÉANS
École d'Ingénieurs de l'Université d'Orléans

ANNEE 2016 -2017

Projet Informatique

PARKING POLYTECH

Etudiants :

Xi CHEN

Gladys MONTALBAN

Ymelda PIANKALI

Tuteur du projet :

Arnaud STOLZ

Date du projet :

Client : Polytech Orléans

06/09/2016 au 10/11/2016

Table des matières

INTRODUCTION

L'UE informatique, nous permet de réaliser un projet pour nous familiariser avec différents types de langage informatique. Notre trinôme a fait le choix de réaliser le projet Parking Polytech proposé par Arnaud Stolz. Il s'agit de concevoir une application pour smartphone connectée à un espace web permettant de connaître en temps réel l'occupation des parkings du campus. Chaque fois qu'une personne se gare, elle le signale sur l'application ; de même lorsqu'elle repart. Enfin, un espace web permettra de tenir à jour la base de données des informations d'entrée/sortie sur chaque parking.

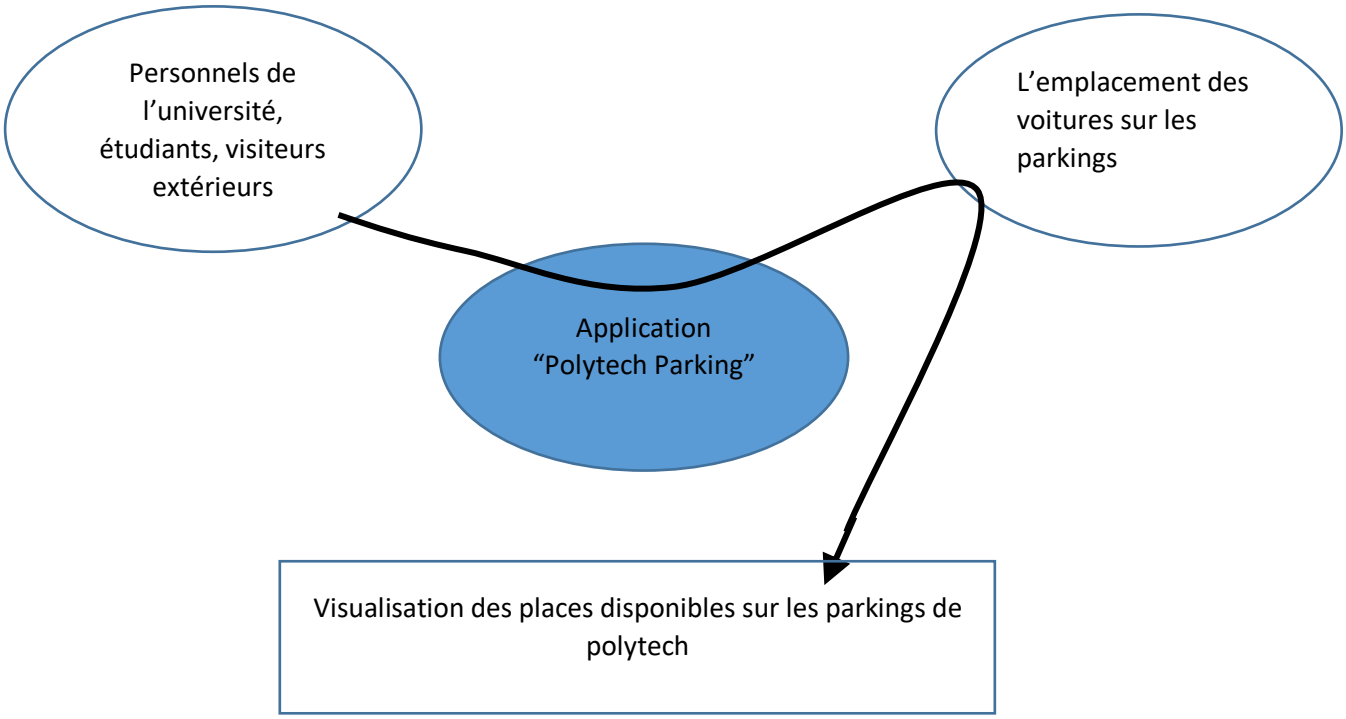
1. Cahiers des charges

1.1. Objectif principal

L'objectif du projet consiste en la réalisation d'une application Android connectée à un espace web. Cette application permettra le comptage des places disponibles sur les parkings de l'université d'Orléans via une base de données.

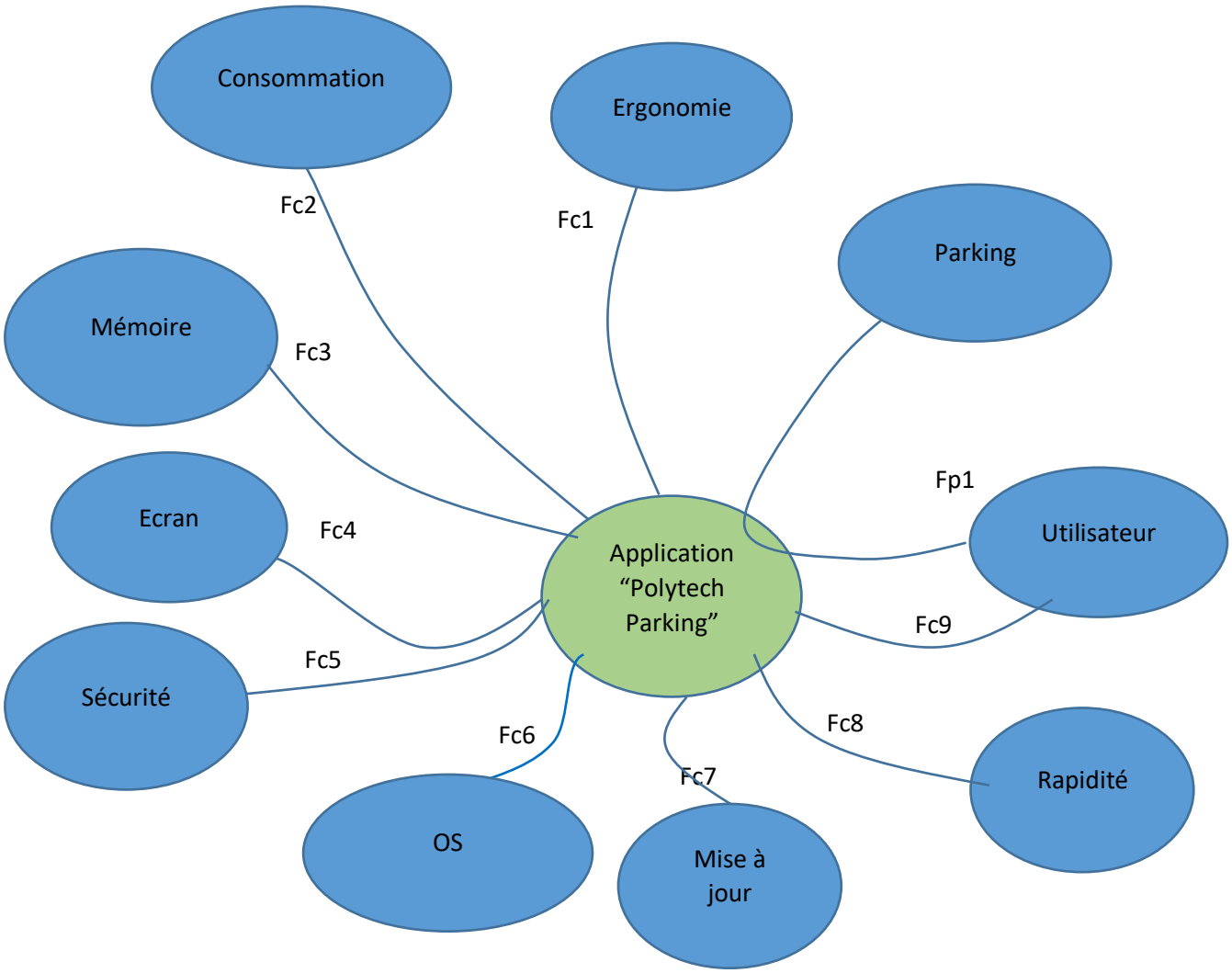
1.2. Contexte et définition du projet

Diagramme bête à cornes



1.3. Contraintes et limites

Diagramme pieuvre



Fonction principale :	
Fp1	Permettre à l'utilisateur de visualiser les places disponibles sur les parkings de l'université.

Fonctions contraintes	
Fc1	Être ergonome. Avoir une interface simple, intuitive et efficace : buttons et interactions visibles.
Fc2	Consommer peu de batterie. Avoir ainsi une consommation du CPU <20%
Fc3	Prendre peu d'espace sur l'espace mémoire du téléphone. <30 Mo
Fc4	S'adapter à toutes les tailles d'écran. Avoir une bonne qualité d'affichage : les images doivent avoir la bonne résolution adaptée à la taille d'écran du mobile utilisée sans déformation.
Fc5	Empêcher la diffusion des données des utilisateurs.
Fc6	Pouvoir être téléchargeable sur une large gamme de version android ou iOs
Fc7	Se mettre à jour automatiquement afin de permettre à l'utilisateur de connaître en temps réel l'évolution du nombre de place disponible.
Fc8	Avoir un temps d'affichage rapide.
Fc9	Authentifier l'utilisateur et le référencer dans la base de données

2. La répartition des tâches

- Tâche 1 : Analyse du sujet
  - ❖ Acteur : Tout le monde
  - ❖ Description : Définition des objectifs du projet et des potentiels outils/logiciels qui entreront en jeu. Recherche de projet existant et de vocabulaire.

- ❖ Jalons : 08/09/2016

- Tâche 2 : Cahier des charges et dossier technique
  - ❖ Acteur : Tout le monde
  - ❖ Description : Prise de rendez-vous avec le tuteur pour être en accord sur l'analyse du sujet. Comparaison de sa vision du projet avec la nôtre et lui faire des propositions, si possible/besoin. Rédaction du cahier de charges (tâches prévisionnelles). Rédaction du dossier technique (solutions envisagées, liste du matériel).
  - ❖ Jalons : 09/09/2016

- Tâche 3 : Renseignement base de données et PHP
  - ❖ Acteur : Ymelda
  - ❖ Description : Savoir créer une base de données reliée à un espace web. S'entraîner à dialoguer sur WAMPServer. Avoir des notions et/ou maîtriser le langage SQL et PHP.
  - ❖ Jalons : 15/09/2016-19/09/2016

- Tâche 4 : Android Studio
  - ❖ Acteur : Gladys
  - ❖ Description : Savoir rendre l'application esthétique et pouvoir l'utiliser sur le web. Avoir des notions/Maîtriser langage PHP et Android studio.
  - ❖ Jalons : 15/09/2016-19/09/2016

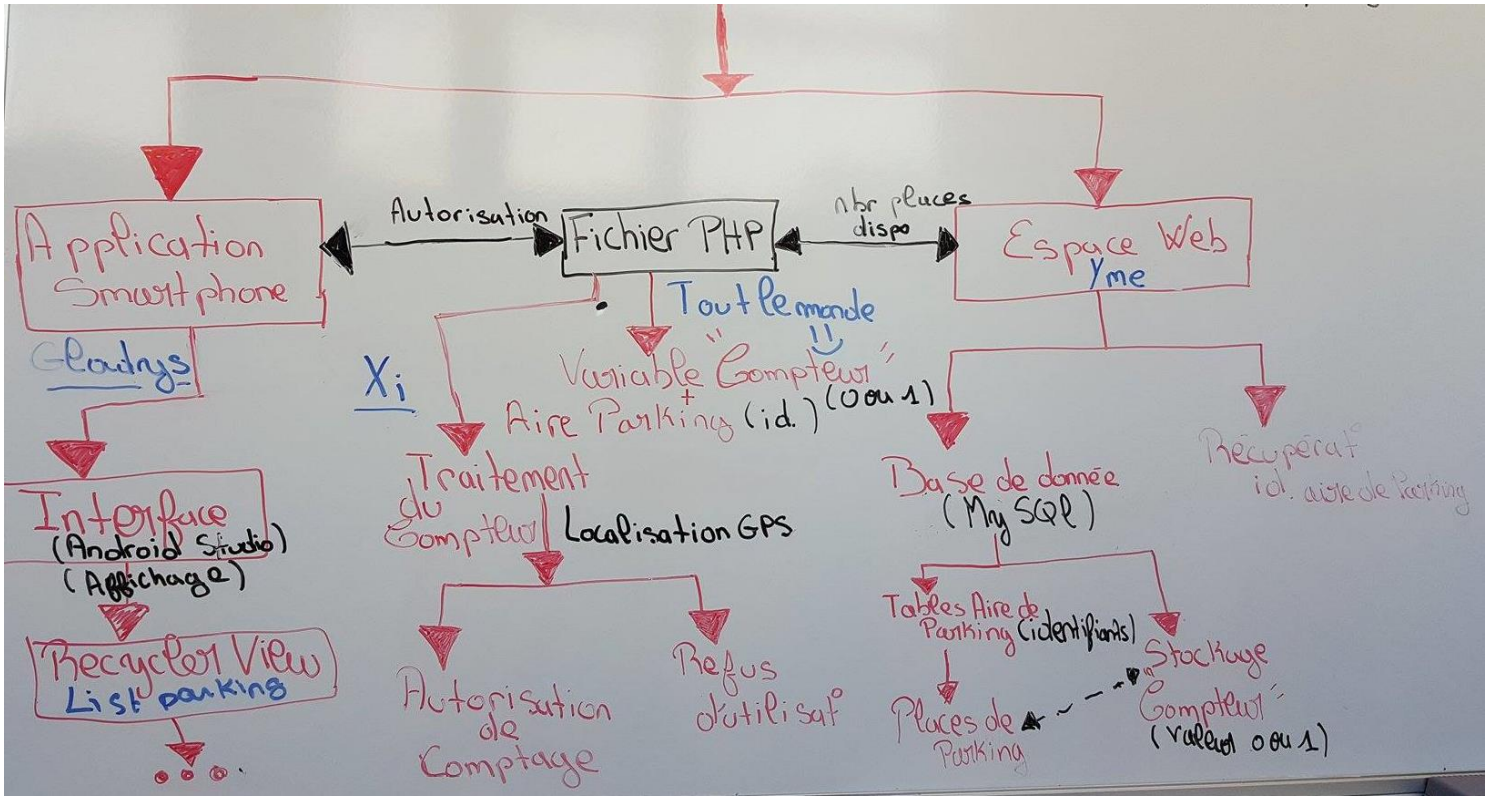
- Tâche 5 : Calibrage du GPS
  - ❖ Acteur : Xi
  - ❖ Description : Utiliser le GPS pour restreindre le champ d'utilisation de l'utilisateur
  - ❖ Jalons : 15/09/2016-19/09/2016

- Tâche 6 : Mise en place du schémas fonctionnel *(cf. photos 2.1)*
  - ❖ Acteur : Tout le monde
  - ❖ Description : Définir les acteurs du programme et les liens.
  - ❖ Jalons : 15/09/2016-24/09/2016

- Tâche 7 : Rédaction du code et test
  - ❖ Acteur : Tout le monde
  - ❖ Description : Android, PHP et SQL
  - ❖ Jalons : 25/09/2016-09/11/2016

Photos 2.1: Schémas fonctionnel

Objectif : Créer une application connectée à un espace web permettant de consulter en temps réel le nombre de place de parking.



3. Les solutions techniques

3.1. Critères de choix

Dans la partie de la localisation de notre projet, nous avons choisi d'utiliser le GPS. Il peut localiser des objets en mouvement avec une bonne performance et une haute précision. Aujourd'hui, le GPS est le meilleur système de navigation et de positionnement, donc nous avons choisi cette méthode pour réaliser la localisation.

3.2. Matériel et logiciels

Nous choisissons le logiciel Android Studio pour écrire les codes Java utilisés dans l'application Android, WAMP pour la mise en place d'une base de données et *Sublime Text* pour écrire le fichier PHP et JSON.

3.3. Etudes comparatives des solutions existantes

Nous trouvons actuellement sur le marché très peu d'applications smartphones pour réserver une place de parking. Certaines utilisent les statiques comme *Path to Park* qui calcule les chances de trouver une place en fonction de l'heure et de la fréquentation. D'autres, comme OnePark , renseignent l'utilisateur sur le nombre de places disponibles sur les parkings des entreprises et hôtels

partenaires. Cependant, il n'existe aucune application de stationnement sur les universités françaises.

4. Structure de l'application

4.1. Partie Android

4.1.1. Tests effectués

Test sur l'interface

Pour accéder à la liste des parkings, nous avons voulu utiliser les boutons images. Nous pensions ainsi, rendre l'application plus simple et efficace. Nous avons ainsi réalisé un premier test (voir image ci-dessous). Chaque image d'un parking amène à une activité suivante. Cette dernière possède les informations du parking sélectionné.

Nous avons abandonné cette idée car l'utilisation des images buttons images ne permettait pas d'avoir une consultation rapide de toutes les informations dont a besoin l'utilisateur. De plus, cela alourdissait le programme car il fallait donc gérer entre 9-10 boutons.

Image : Sélection des données d'un parking en utilisant un bouton image

4.1.2.Interface

Notre application Android comprend 7 classes (voir schéma).

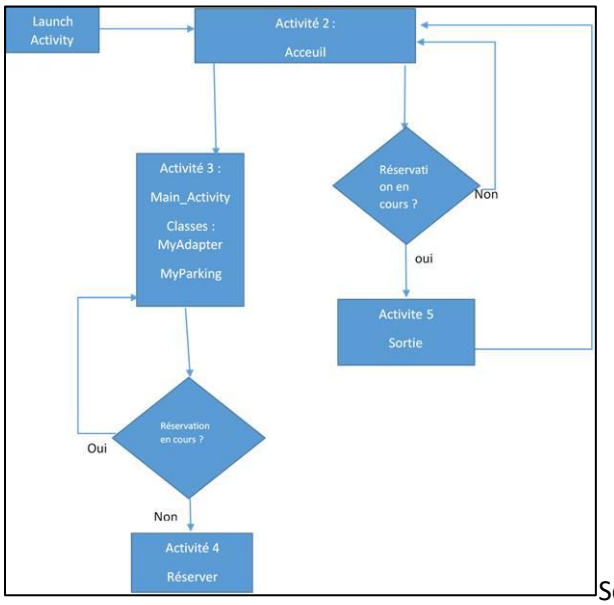


Schéma : Organisation entre les activités

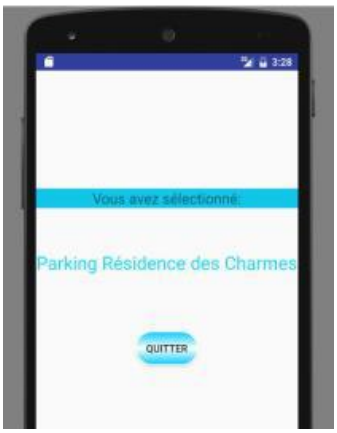
Activité 1: Launch Activité

Classe utilisée : Launch_activity

Cette activité ne dure que 5 secondes. Elle présente le logo polytech et informe sur la nécessité d'activer le GPS et l'internet

Activité 3 : Liste parking

Activité 4 : Réserver



Activité 5 : Sortir

Activité 2 : Accueil

Activité 2 : Accueil

Classe utilisée : Accueil

Cette activité propose deux choix à l'utilisateur. Soit il peut voir la réservation en cours- si celui-ci a précédemment effectué une réservation- soit l'utilisateur peut voir la liste des parkings disponibles. Dans cette activité l'utilisateur est localisé (voir 4.1.3)

Activité 3 : L'activité principale

Classe : Main_Activity

Cette activité présente la liste des parkings disponibles. Elle n'est appelée que si l'utilisation choisi de visualiser la liste

Pour gérer la liste des parkings, nous avons utilisé les RecyclerView. Comme une listView , un recyclerView est une liste d'un même layout (voir image ci-contre) possédant les mêmes caractéristiques(4 textView et une image)mais des donnés différentes.

Pour utiliser une recyclerview, nous avons créé la classe MyParking qui contient les variables et les méthodes nécessaire à l'instanciation d'une liste de parking et donc de la recyclerview.

```
public class MyParking {

    private String m_nomParking; // Nom du Parking
    private String m_DescriptionParking;// description du Parkin
    private int m_Nb_placedisponible; // nombre de place disponible sur le park

    private int m_imageid; // identifiant de l'image correspondante au parking
```

```
public MyParking(String nomParking, String Description, int imageurl, int place_disponible)
```

Nous utilisons classe MyAdapter pour gérer l'appellation des données, l'instanciation des cellules du recyclerView et son affichage. Cette classe en contient une autre ViewHolder. Un objet de type ViewHolder va garder les références vers les vues de chaque cellule du Recycler.

La classe Main_Activity gère l'appellation du fichier JSon pour récupérer le nombre de place disponible de chaque parking (voir 4.2)

Si l'utilisateur clique sur une cellule parking, le programme récupère l'identifiant de la cellule cliquée grâce à la fonction `getAdapterPosition()`. Celle-ci est appelé par la fonction OnClick du ViewHolder qui permet de gérer la clique sur les cellules du recyclerView.

Activité 4 : Réserver

On utilise la classe Reserver utilisant le layout reservation.xml. Grâce au fonction *PutExtra* et *GetIntExtra*, nous avons récupérer l'identifiant du parking (ou de la cellule du recyclerview) sélectionné par l'utilisateur.

L'application stockage le numéro du parking sélectionné, dans la mémoire interne du téléphone. Nous utilisons la fonction Enregistrement () pour créer un fichier "parking.txt" qui contiendra cette donnée.

Activité 5 : Sortir

Pour cette activité nous utilisons la classe sortie et le layout sortie.xml.

L'utilisateur ne peut accéder à cette activité que s'il y a une réservation en cours. Ainsi une boîte de dialogue apparaîtra, si le fichier parking.txt n'est pas trouvé dans la mémoire interne du téléphone ou si ce fichier ne contient pas de nombre mais des espaces.

Quand l'utilisateur clique sur le bouton "signaler ma sortie", la fonction *DeleteStockage()* est appelé pour supprimer la donnée du parking. Nous remplaçons ce nombre par le nombre 5555. Ce nombre permettra à l'application de savoir s'il y a une réservation en cours ou non.

4.1.3 Localisation

Dans notre projet, l'utilisateur peut utiliser la fonction de réservation d'une place de l'application, uniquement lorsque celui-ci est proche de l'Université. Pour ce fait, nous utilisons le GPS pour réaliser la localisation du portable d'utilisateur.

a) Logiciel

C'est une partie qui peut être gérée par Android donc nous utilisons Android Studio pour écrire les codes du GPS.

b) Étapes de réalisation :

- Installer l'autorisation dans le fichier manifest.xml pour utiliser la classe Location :
  android:name="android.permission.ACCESS_FINE_LOCATION"/>

- Connaître la classe Location
  La classe Location contient la longitude,la latitude et quelques informations facultatives sur l'altitude, la vitesse et la direction.

- Obtenir Location Manager
  Location Manager permet d'accéder aux services de localisation du système. Ces services permettent à une application d'obtenir l'emplacement de l'équipement et faire des mises à jour régulières. De plus, vous pouvez lancer une activité spécifique dans l'application, si l'utilisateur est à proximité de l'endroit ciblé.

  locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

- Installer le suivi continu
  Avec les changements de position, pour réaliser la localisation, nous faisons la plage de variation de la position qui contient l'intervalle de temps minimum et le déplacement minimal. Si les deux conditions sont remplies, la position de l'auditeur sera déclenchée.

  ```
  locationManager.requestLocationUpdates("gps", 5000, 0, locationlistener);
  ```

- Installer LocationListener
  Lorsque la position est modifiée, OnLocationChanged () va être exécuté :
  ```
  public void onLocationChanged(Location location) {
  ```

4.2. Espace web

La partie du serveur web concerne la gestion des données. Le fichier PHP permet de lier notre base de données, qui contient le nombre de places disponibles sur les différents parking, avec notre application mobile, qui permettra principalement aux utilisateurs de consulter le nombre de place disponible.Pour ne pas modifier la base de données des parkings nous avons effectués tous nos tests pour l'envoie et la réception des données en JSON sur la base de donnée "*idlville*" car nous ne voulions pas faire de modification dans la base de donnée principal du programme à cause des identifiants que nous avions donnée à chaque parking.

4.2.1. Base de données

Une base de données au nom de "*polytech_parking"* a été créée avec le logiciel MySQL. Ce logiciel est dans WAMP Server, téléchargeable sur google. WAMPServer est un pack pour Windows contenant Apache, Mysql et Phpmyadmin. Il est plus pratique d'installer WAMP ou XAMPP (pour les macs). La base de donnée que l'on a créée est donc virtuelle,non utilisable sur une large quantité d'appareil car elle se trouve uniquement sur le PC. Cependant, la base de données ne fait que traiter les données, suivant les requêtes, décrit dans le fichier PHP.

La base de données se trouve sur un serveur local quel qu'en soit l'IP, pour être utilisable de manière publique, il faudrait la mettre sur un hébergeur.

4.2.2. PHP et JSON

Les fichiers PHP gèrent la base de données puis ils envoient/reçoivent les données sur Android Studio. Toutes les transmissions de donnée sont faites sous format JSON car c'est un langage simple, dont les données sont faciles à transporter. La notation "JSON encode" se trouvant dans les fichiers PHP, permet de transformer les données SQL en JSON. Cependant nous avons choisis d'utiliser la bibliothèque "Volley" qui nous permet d'envoyer et recevoir des données JSON vers PHP de manière. Dans le fichier Android studio, la fonction qui gère la réception des données

Il est très important de modifier les autorisations des fichiers "httpd.conf" et "Virtual host" qui sont dans WAMP Server.

"C://wamp64/bin/apache/conf/httpd.conf"

```
#

#Listen 12.34.56.78:80

Listen 192.168.*.**:80
```

Il faut remplacer "*" par les chiffres correspondants à l'adresse IP du PC qui contiendra la base de données. Si les appareils (smartphone et PC) sont connectés au même wifi, il est donc possible de transmettre les données JSON.

"C://wamp64/bin/apache/conf/extra/httpd-vhosts.conf"

```
# Virtual Hosts

<VirtualHost *:80>

        ServerName localhost

        ServerName 192.168.*.**

        DocumentRoot c:/wamp64/www

        <Directory  "c:/wamp64/www/">

                Options +Indexes +Includes +FollowSymLinks +MultiViews

                AllowOverride All

                Require all granted

        </Directory>

</VirtualHost>
```

Pour connaître l'adresse IP de votre PC, il faut taper la requête "ipconfig" dans le système de commande de Windows. (Voir guide d'installation)

5. Bilan du produit

5.1. Avancement du projet

A l'heure actuelle, l'application est opérationnelle à 80%. La connexion Android à la base de données et de la base de données à Android via le fichier PHP s'effectue avec rapidité. De plus l'actualisation du nombre de place disponible est effectuée sur tous les téléphones utilisés. Ainsi, l'utilisateur peut visualiser en temps réel l'évolution du nombre de place et choisir le parking voulu.
Les hébergeurs de base de données n'étant pas gratuit nous utilisons un serveur local. En effet, pour utiliser l'application il faut que le mobile soit connecté avec la même adresse IP.

<u>Caractéristique de l'application</u>

Espace mémoire : 4.30 Mo

Niveau de consommation d'énergie :14,23 Mah pour 1 minute d'utilisations

5.2. Amélioration du projet

Nous n'obtenons pas le nombre réel de place disponible car nous n'avons pas de dispositif pour prendre en compte les usagers qui n'ont pas l'application. C'est ainsi que nous avons pensé à un dispositif de capteur tel que celui développé par BOSH.

Capteur BOSH sans-fil pour signaler les places de parking libres (www.aruco.com)

Nous avons imaginé un capteur qui équipé d'un dispositif à LED.Si notre dispositif est rouge, l'utilisateur qui n'a pas l'application saura que la place est réservée.

L'utilisateur de l'application, ayant réservé sa place, pourra, quant à lui, se garer à cet emplacement. A l'inverse, si le dispositif est bleu ou vert, l'utilisateur, n'ayant pas fait de réservation, saura qu'il peut se garer. Par conséquent, cette place sera décomptée dans la base de données.

Ainsi, cela permettrait de collecter le nombre de place réelle de chaque parking. De plus, pour rendre l'application utilisable par tous, il faudrait mettre en ligne la base de données et donc avoir un hébergeur comme le site Polytech 'Orléans ou créer un site avec "Webhost", ce dernier n'étant pas gratuit.

Conclusion

Actuellement, l'application est fonctionnelle uniquement sur le serveur local mais les interactions se font bien en temps réel. Nous avons réalisé les objectifs de départ, demandée par le client. Après, différentes recherches, nous nous sommes rendu compte que le marché sur ce type d'application est toujours en développement. Il est donc naturel de notre part de vouloir améliorer la partie informatique en la combinant avec une partie électronique, essentiel pour l'exactitude de nos résultats et la prise en compte de tout utilisateur. Au cours de ce projet, nous avons pu acquérir des connaissances en langage SQL, JAVA, HTML, PHP et l'importance de l'adresse IP.

En dépit du fait que la gestion de projet était minime et par manque de temps pour effectuer nos idées d'améliorations, nous avons tout de même terminer le code informatique de l'espace web et de l'application. Enfin, nous aimerions que par la suite le projet soit proposé dans les UE domotique ou multimédia pour l'utilisation des capteurs.