



Projet proposé par



Advanced Sport Instrument Sàrl

# FieldWiz Monitoring

## Rapport

Projet de semestre 6 - Informatique



Auteur

Daniel Garcia

Superviseurs

Joël Dumoulin

Omar Abou Khaled

Elena Mugellini

Fribourg, le 12 mai 2017

HumanTech

Technology for  
Human Wellbeing Institute



# Table des matières

<b>1. Biographie.....</b>	<b>5</b>
<b>2. Introduction .....</b>	<b>6</b>
2.1. Contexte .....	6
2.2. Objectifs.....	6
2.2.1. <i>Récupération des données</i> .....	6
2.2.2. <i>Affichage des données</i> .....	6
2.2.3. <i>Gestion des données</i> .....	6
2.2.4. <i>Objectifs secondaires</i> .....	7
<b>3. Analyse .....</b>	<b>8</b>
3.1. Architecture générale .....	8
3.1.1. <i>FieldWiz</i> .....	8
3.1.2. <i>Smartphone Android</i> .....	9
3.1.3. <i>USB</i> .....	10
3.1.4. <i>Node.js</i> .....	11
3.1.5. <i>Scrpits MATLAB</i> .....	12
3.1.6. <i>Neo4j</i> .....	12
3.2. Librairies Android.....	15
3.2.1. <i>USB Host</i> .....	16
3.2.2. <i>Geocoder</i> .....	16
3.2.3. <i>GSON</i> .....	16
3.2.4. <i>okhttp3</i> .....	16
3.2.5. <i>CircleImageView</i> .....	16
<b>4. Conception .....</b>	<b>17</b>
4.1. Diagramme de cas d'utilisation .....	17
4.2. Diagramme de séquences.....	18
4.2.1. <i>Se connecter</i> .....	18
4.2.2. <i>Créer un compte</i> .....	19
4.2.3. <i>Uploader une session</i> .....	20
4.2.4. <i>Visualiser les statistiques</i> .....	22
4.2.5. <i>Se déconnecter</i> .....	23
4.3. Diagramme d'état-transition .....	24
4.4. Maquettes .....	25
4.4.1. <i>Login</i> .....	25
4.4.2. <i>Création d'un compte</i> .....	25
4.4.3. <i>Page principale</i> .....	26
4.4.4. <i>Upload</i> .....	27
4.4.5. <i>Statistique</i> .....	27
4.5. Requêtes .....	28
4.5.1. <i>Contrôle l'authentification d'un utilisateur</i> .....	28
4.5.2. <i>Création d'un compte (1ère partie: données utilisateur)</i> .....	28
4.5.3. <i>Création d'un compte (2ème partie: envoie de l'avatar)</i> .....	28
4.5.4. <i>Récupération des infos de l'utilisateur</i> .....	28
4.5.5. <i>Récupération de l'avatar de l'utilisateur :</i> .....	29
4.5.6. <i>Récupération des sessions d'un fichier .FWZ</i> .....	29



4.5.7. <i>Récupération des dates de début et de fin de session</i> .....	29
4.5.8. <i>Création d'une session</i> .....	29
4.5.9. <i>Récupération des statistiques d'un utilisateur</i> .....	29
<b>5. Implémentation.....</b>	<b>30</b>
5.1. Application Android .....	30
5.1.1. <i>Interface de l'application</i> .....	30
5.1.2. <i>Activités</i> .....	31
5.1.3. <i>Client HTTP</i> .....	32
5.1.4. <i>Formatage en JSON</i> .....	34
5.1.5. <i>Capture de photos</i> .....	34
5.1.6. <i>Affichage d'informations</i> .....	35
5.1.7. <i>Broadcast receiver</i> .....	36
5.1.8. <i>Recherche de fichier « .FWZ »</i> .....	36
5.1.9. <i>Gestion des ListView</i> .....	37
5.1.10. <i>Geocoder</i> .....	37
5.2. Node.js.....	38
5.2.1. <i>URL avec le nom d'utilisateur</i> .....	38
5.2.2. <i>Retour d'un JSON</i> .....	38
5.2.3. <i>Réception de fichiers</i> .....	39
5.2.4. <i>Exécution des commandes</i> .....	39
5.2.5. <i>Envoie de l'avatar</i> .....	40
5.3. Neo4J .....	40
5.3.1. <i>MATCH</i> .....	40
5.3.2. <i>CREATE</i> .....	40
5.4. Script MATLAB .....	40
5.4.1. <i>Liste des sessions</i> .....	40
5.4.2. <i>Statistiques</i> .....	41
<b>6. Tests .....</b>	<b>44</b>
6.1. Tests fonctionnels.....	44
6.1.1. <i>Création de comptes</i> .....	44
6.1.2. <i>Connexion</i> .....	48
6.1.3. <i>Upload des sessions</i> .....	50
6.1.4. <i>Visualisation des données</i> .....	52
6.1.5. <i>Déconnexion</i> .....	53
6.1.6. <i>Synthèse des tests</i> .....	54
6.2. Tests utilisateur .....	54
6.2.1. <i>Feedback</i> .....	54
6.2.2. <i>Synthèse</i> .....	55
6.3. Améliorations .....	55
6.3.1. <i>Saisie du clavier</i> .....	55
6.3.2. <i>Contrôle des saisie</i> .....	55
6.3.3. <i>Unicité</i> .....	55
6.3.4. <i>Blogeage</i> des boutons .....	56
6.3.5. <i>Rendre cliquable ce qui est cliquable</i> .....	56
6.3.6. <i>Affichage de messages</i> .....	56
<b>7. Problèmes rencontrés.....</b>	<b>57</b>



7.1.	Application Android .....	57
7.1.1.	<i>Impossibilité de lire le fichier .FWZ</i> .....	57
7.1.2.	<i>Format JSON</i> .....	57
7.1.3.	<i>Gestion des thread</i> .....	57
7.2.	Node.js.....	57
7.2.1.	<i>Exécution des commandes</i> .....	57
7.2.2.	<i>Retour des résultats Neo4j</i> .....	58
7.3.	Script MATLAB .....	58
7.3.1.	<i>Retour des session</i> .....	58
7.3.2.	<i>Format des dates</i> .....	59
<b>8.</b>	<b>Conclusion.....</b>	<b>60</b>
<b>9.</b>	<b>Références.....</b>	<b>61</b>
1.	Documentation .....	61
2.	Figures .....	62



## 1. Biographie

En phase de terminer le Bachelor en informatique, Daniel Garcia Barros étudie dans la Haute école d'ingénierie et d'architecture de Fribourg. Passionné de sport, il pratique du football depuis plus de 20 ans et fait parti du FC Corminboeuf, en 4ème ligue fribourgeoise. Malheureusement, depuis peu, il a été victime d'un accident de football, au niveau du genou droit (rupture du ligament croisé antérieur).

Daniel Garcia Barros éprouve un grand intérêt dans la programmation, plus particulièrement dans le domaine des applications mobiles. En effet, son objectif à la fin de ses études est de trouver un emploi qui lui donnera l'opportunité de continuer dans ce domaine.

Il a toujours rêvé de combiner sa passion avec son métier, chose que le projet *FieldWiz Monitoring* permet de le faire.





## 2. Introduction

### 2.1. Contexte

De nos jours, il est possible de recueillir toutes sortes d'informations sur le suivi d'un objet grâce à la technologie GPS. Il existe un instrument de mesure permettant de récupérer les données : FieldWiz, développée et commercialisée par la société Advanced Sport Instrument. L'entreprise est basée à Paudex, dans le canton de Vaud. Cet appareil est spécialement conçu dans le domaine du sport et plus particulièrement les sports d'équipe en extérieur. Il est utilisé dans de nombreuses équipes professionnelles et semi-professionnelles.

Cependant, il n'est pas possible de visualiser ses statistiques sur un smartphone, à moins de passer par un ordinateur hôte qui prélève les informations du FieldWiz et les envoie en format PDF à un smartphone. Cette manière de faire prend beaucoup de temps et nécessite un ordinateur. La solution proposée dans ce projet est de connecter directement le FieldWiz à un smartphone grâce à une connexion micro USB.

Ce projet n'est pas la suite d'un autre projet et sera repris dans un projet Bachelor intitulé *FieldWiz Monitoring II*. Une partie des objectifs secondaires sera entre autres implémenté dans l'avenir.

### 2.2. Objectifs

L'objectif de ce projet de semestre est de réaliser une application mobile Android servant aux sportifs afin qu'ils puissent visualiser les statistiques d'un entraînement ou d'un match en récupérant au préalable les données du FieldWiz grâce à une connexion câblée de type micro USB vers micro USB avec le mode OTG. Le sportif aura un compte utilisateur afin d'enregistrer ses performances.

#### 2.2.1. Récupération des données

Le premier objectif de ce projet est de comprendre le principe de la connectivité des ports micro USB avec le mode OTG. Ensuite, il faut traiter le fichier généré par le FieldWiz contenant les données d'une session de sport pour que l'application puisse traiter les informations.

#### 2.2.2. Affichage des données

Le deuxième objectif de ce projet est d'afficher les données reçues du Pod FieldWiz de sorte que l'utilisateur puisse visualiser ses informations personnelles et les statistiques d'une session.

#### 2.2.3. Gestion des données

Le troisième objectif est de sauvegarder les sessions d'un utilisateur dans une base de données externe. L'application fournit un moyen de créer un login et de se connecter au système.



## 2.2.4. Objectifs secondaires

Plusieurs objectifs secondaires ont été pensés surtout pour la suite de ce projet qui sera un projet de Bachelor :

- Comparaison des statistiques d'une session avec plusieurs sessions, d'afficher la distance moyenne parcourue durant des matchs ou des entraînements.
- Protocole de tests utilisateur afin de garantir l'utilisabilité de l'application
- Deux types de compte : compte entraîneur et compte joueur. L'entraîneur aura le droit de consulter les données d'un joueur, de comparer avec d'autres joueurs.
- Implémentation d'un réseau social de sportifs. Le sportif aura la possibilité d'ajouter des amis et d'avoir la permission de consulter les statistiques des amis. L'utilisateur pourra lancer un challenge à un de ses amis afin de voir qui aura parcouru la plus grande distance ou bien celui qui a couvert le plus les zones en défense.

## 3. Analyse

Dans cette partie de document, l'analyse globale du projet sera développée en introduisant chaque technologie utilisée, les alternatives et les raisons des choix.

### 3.1. Architecture générale

L'image ci-dessous représente l'architecture générale du projet de semestre.

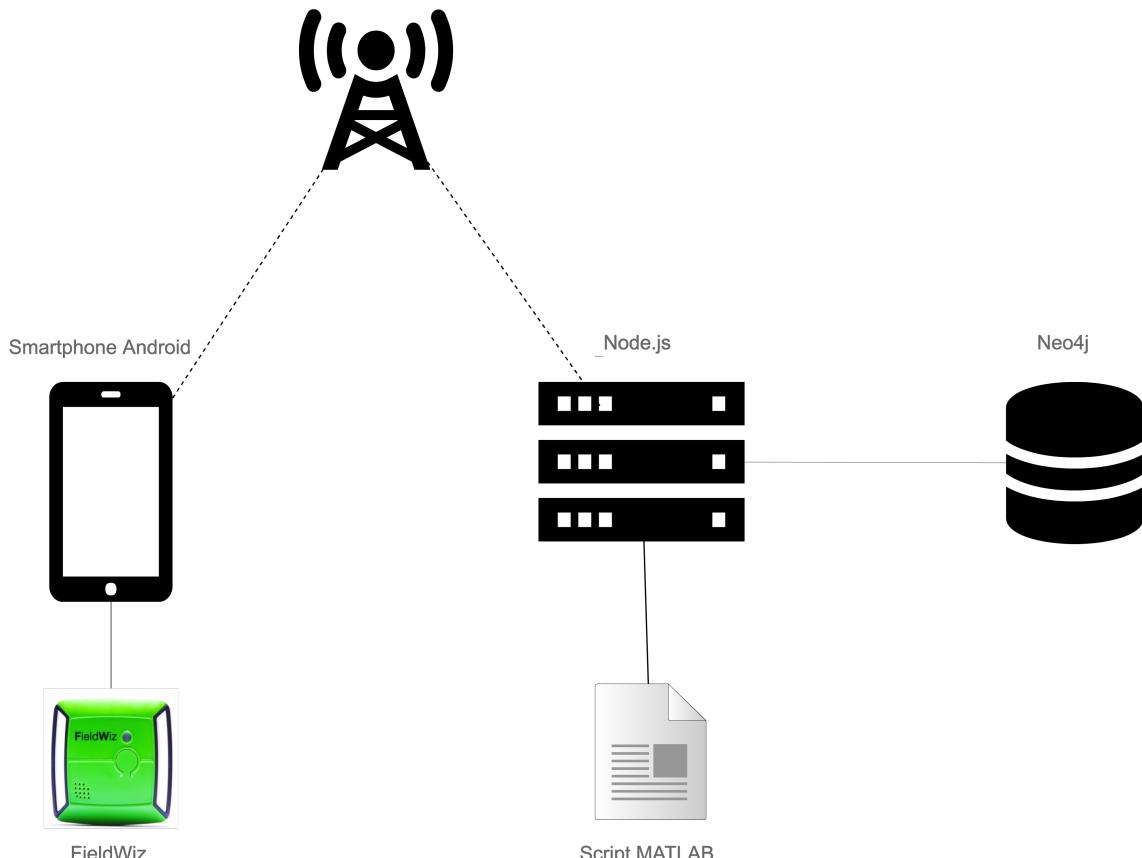


Figure 1 : Architecture générale de l'application

#### 3.1.1. FieldWiz

Le FieldWiz est l'appareil de mesure qui contient toutes les données de toutes les sessions d'un utilisateur. En se basant sur les signaux GPS, l'appareil capture 10 mesures par seconde.

Le système actuel permet de ressortir les statistiques telles que la carte de chaleur, la distance parcourue, la vitesse ou encore l'accélération. Pour ce faire, il suffit d'avoir en possession un FieldWiz, d'avoir fait une session de sport et d'un ordinateur connecté à internet, et de se rendre à la page Web suivante :

<http://upload.fieldwiz.com/>

L'utilisation de cet appareil est simple : il suffit de se trouver dans un environnement non fermé, et d'appuyer sur le bouton. La LED du FieldWiz clignote de façon lente : il est en train de détecter la position GPS. Cela dure environ une minute. La LED du FieldWiz clignote de façon rapide : cela veut dire que la session a démarré. Une fois la session de sport terminée, il faut appuyer à trois reprises sur le bouton. Le FieldWiz s'éteint et la session est terminée.

Pour pouvoir porter l'appareil, il faut se munir d'un gilet, et de l'enclencher à l'intérieur de celui-ci



Figure 2: un FieldWiz et un gilet

Dans le cadre de ce projet, le FieldWiz est directement branché au smartphone en utilisant la connectivité USB. La technologie USB offre la possibilité de récupérer les données grâce au mode OTG (On-The-Go).

### 3.1.2. Smartphone Android

L'utilisateur se sert du smartphone Android pour exécuter l'application FieldWiz Monitoring pour exécuter les tâches telles que la récupération des données, l'affichage des données ou encore la sauvegarde des données. Il joue le rôle de client HTTP. Il devra avoir accès au serveur Web en utilisant une connexion intranet.

Afin de pouvoir visionner les statistiques sur l'application mobile, il est nécessaire d'avoir un moyen permettant de transférer les données du FieldWiz vers un smartphone. La technologie USB offre la possibilité de récupérer les données grâce au mode OTG (On-The-Go).

Le montage du FieldWiz dans le smartphone est fait automatiquement. Les données du périphérique se trouvent dans **/storage/F0E3-25FC**. Lors de la connexion des deux composants, il est possible de parcourir l'arborescence et de transférer des fichiers. L'image ci-dessous montre l'arborescence du FieldWiz via un smartphone Android :

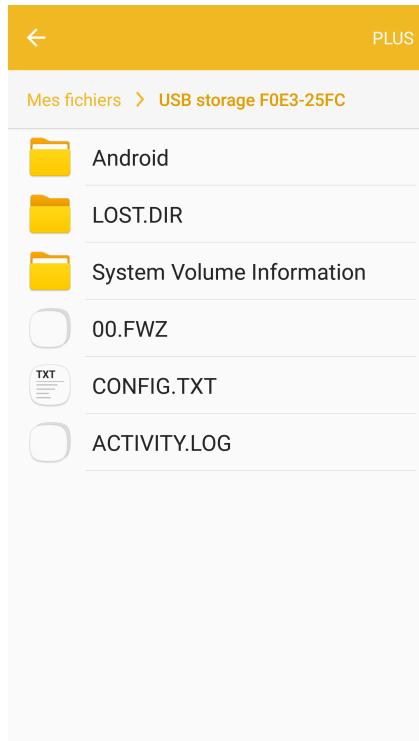


Figure 3: Exploration du Pod FieldWiz sur un smartphone (Samsung S7)

Dans le cadre de ce projet, le smartphone utilisé est un Samsung Galaxy S7 (version Android : 7.0). La technologie Android a été imposée par le descriptif du projet de semestre.

### 3.1.3.USB

USB (Universal Serial Bus) est un standard utilisé sur la plupart des appareils électroniques qui permet d'interconnecter un périphérique et un ordinateur dans le but de transférer des données.



Figure 4: Logo USB

Il existe plusieurs connecteurs USB classés par type (A, B ou C) et par taille (normal, mini, micro) et sont soit mâle, soit femelle. Dans le contexte du projet, la liaison entre le smartphone et le Pod FieldWiz se fera avec un câble USB OTG **micro-B mâle**.

Généralement, l'utilisation de l'USB se fait entre un ordinateur et un périphérique qui jouent le rôle de maître et d'esclave respectivement. Le contrôle du périphérique se fait depuis un ordinateur.



La norme USB OTG (On-The-Go) permet de transférer des données d'un appareil vers un autre appareil sans devoir passer sur une machine hôte en choisissant celui qui va jouer le rôle du maître et de l'esclave. Cependant, le maître doit être compatible OTG. Le périphérique hôte doit avoir l'embout du câble où est inscrit « OTG ».

Dans le contexte du projet, le maître sera le smartphone Android et l'esclave sera le Pod FieldWiz. La technologie USB en mode OTG a été imposée par le descriptif du projet de semestre.

### 3.1.4. Node.js

Node.js est un serveur Web utilisant une architecture RESTfull (paradigme client-serveur, sans état, mise en cache). Il exécutera les requêtes HTTP demandé par les clients (à savoir les utilisateurs de l'application *FieldWiz Monitoring*). Les principales tâches du serveur seront d'exécuter des scripts MATLAB, requêter la base de données et répondre aux clients.

Node.js fourni une panoplie de modules. Dans le cadre de ce projet de semestre, les modules utilisés sont :

- Express : permet de faciliter l'utilisation du Node.js
- Multer : permet de récupérer des fichiers des méthodes POST
- Exec : permet de d'exécuter des commandes
- neo4j-driver : permet de requêter la base de données Neo4j
- fs : permet de gérer la lecture et l'écriture des fichiers

#### A. Installation

La phase d'installation a été effectuée avec un MAC OS 10.12. En premier lieu, il faut télécharger l'exécutable à l'adresse suivante :

<https://nodejs.org/en/>

Ensuite il faut exécuter le fichier téléchargé. Il n'y a rien de spécial concernant l'installation. Il suffit de suivre les instructions.

Une fois installée, la commande « node » peut être invoqué depuis un terminal. Pour lancer un script il suffit de taper la commande :

```
node « fichier.js »
```

« fichier.js » est le fichier contenant les configurations et les méthodes GET, POST du serveur Web.

Par défaut, Node.js utilise le port 3000.



## B. Avantages

Node.js offre de nombreux avantages en comparant à d'autres types de serveurs Web :

- Il utilise le JavaScript, un langage basé sur les événements : utilisation des méthodes non bloquantes
- Il utilise le moteur V8 de Google Chrome : l'exécution du code JavaScript rapide

### 3.1.5. Scripts MATLAB

Les scripts MATLAB permettent de traiter les fichiers « .FWZ » (fichiers propriétaires), de retourner les sessions disponibles et de retourner les valeurs concernant la session d'un utilisateur (vitesse max, vitesse moyenne et distance parcourue). Malheureusement, la plateforme Android ne permet pas de lire et exécuter les scripts MATLAB. C'est pour cela que le script sera exécuté côté serveur et non côté client.

En ce qui concerne le traitement du fichier, ASI fournit le script pour résoudre ce problème, avec l'extension « .m », ce qui veut dire que seul deux logiciels sont capables de traiter ce type de fichier: Octave et MATLAB.

## A. Octave vs MATLAB

Il n'y a pas vraiment de réelles différences entre ces deux logiciels. MATLAB offre une multitude de boîtes à outils spécialisés pour plusieurs domaines. Par conséquent, le logiciel est lourd et les scripts exécutés depuis la ligne de commande ne sont pas autant performants qu'avec Octave. De plus, MATLAB nécessite une licence, alors que Octave est un logiciel libre.

## B. Installation

Il faut se procurer le logiciel Octave sur le lien suivant :

<https://www.gnu.org/software/octave/download.html>

Ensuite, il suffit de suivre les instructions de l'installation. Pour exécuter les commandes Octave depuis la ligne de commande, il suffit de taper la commande « octave ».

### 3.1.6. Neo4j

Neo4j est un système de gestion de base de données qui a la particularité d'être basée sur les graphes orientés et utilisent la théorie des graphes pour la résolution des algorithmes. Il fait partie de la famille des SGBD NoSQL. Il est composé de trois éléments essentiels :

- Nœuds
- Relations
- Propriétés

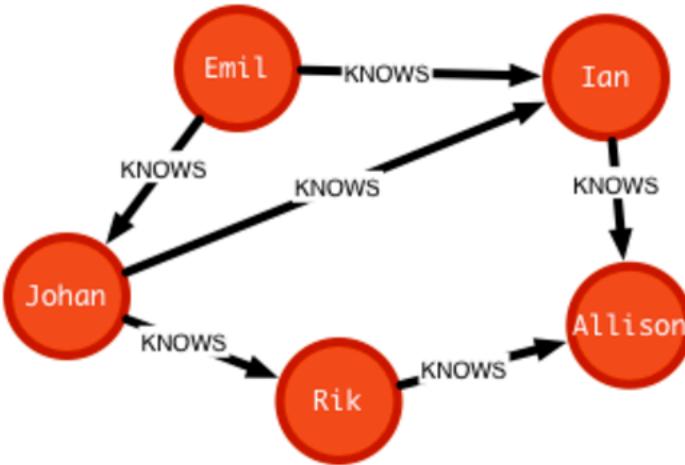


Figure 5: exemple d'un graphe Neo4j

On peut s'apercevoir sur l'exemple ci-dessus qu'il y a des nœuds « Person », possédant une propriété « name » et qui ont des relations « KNOWS ». Les nœuds et les relations peuvent contenir zéro ou plusieurs propriétés.

Neo4j utilise un langage bien spécifique pour les requêtes: Cypher. Voici un exemple de création d'un nœud :

```
CREATE (p:Person { name: "Daniel", locality: "Fribourg", zip: 1700 })
```

La requête crée un nœud « Person » avec les propriété « name », « locality » et « zip ».

Neo4j devra sauvegarder les informations personnelles d'un utilisateur du système (login, mot de passe, données personnelles) et les sessions d'un utilisateur. La base de données du projet de semestre se trouvera sur un MAC OS X 10.12.

### A. Avantages

Neo4j possèdent plusieurs avantages par rapport aux bases de données relationnelles ou NoSQL.

Les SGBD de type NoSQL ont généralement été conçus pour répondre à des problèmes d'accroissement de données rapide. Ils sont en quelque sorte un complément pour les bases de données relationnelles. En partant du principe que le nombre d'utilisateur de *FieldWiz Monitoring* peut être conséquent, il est préférable d'opter pour un SGBD de type NoSQL.

Il existe quatre types de base de données NoSQL : Associatif, orienté colonne, orienté document et orienté graphe. En vue d'une future implémentation d'un réseau social à la suite de ce projet, il faut bien s'imaginer qu'il y aura une quantité énorme de relations entre les données. La future base de données devrait être capable de répondre à des requêtes de type « quels sont les amis de mes amis ? » afin de ressortir une suggestion d'ami, ou bien encore « quels sont mes amis qui ont joué

plus de match que moi ? ». Neo4J est la solution idéale pour répondre aux besoins des futures requêtes de l'application. En effet, contrairement aux autres bases de données NoSQL, Neo4J est performant dans la résolution des requêtes en profondeur.

### B. Installation

En premier lieu, il faut télécharger Neo4J sur le lien suivant :

<https://neo4j.com/download/?ref=home>

Lors de l'installation, il suffit de suivre les instructions. Il n'y a pas de manipulation nécessaire à faire.

### C. Configuration

Lors du démarrage de Neo4j, il est possible de configurer le serveur, ou bien de le lancer directement.

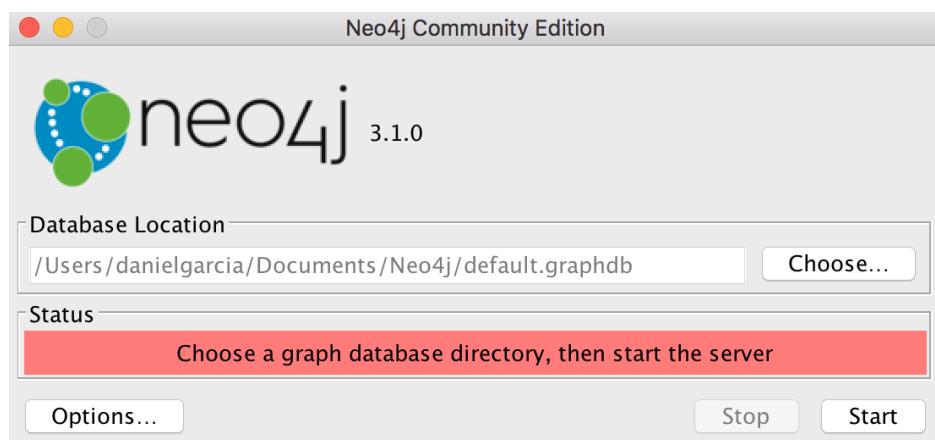


Figure 6: Ouverture de Neo4j (Mac)

Il est nécessaire de configurer le serveur afin qu'il soit disponible pour requêter depuis l'extérieur. Dans le cadre de ce projet, Node.js doit requêter la base de données.

Il y a la possibilité de modifier le fichier de configuration à sa guise. Le fichier (dans les systèmes OS X 10.12) se trouve dans :

/Users/ « username »/Documents/Neo4j/.neo4j.conf

### D. Interface

Pour visualiser les données de la base de données, il suffit d'ouvrir un navigateur web, et de se rendre à l'adresse suivante :

<http://localhost:7474/browser/>

Neo4j utilise le port 7474 par défaut.

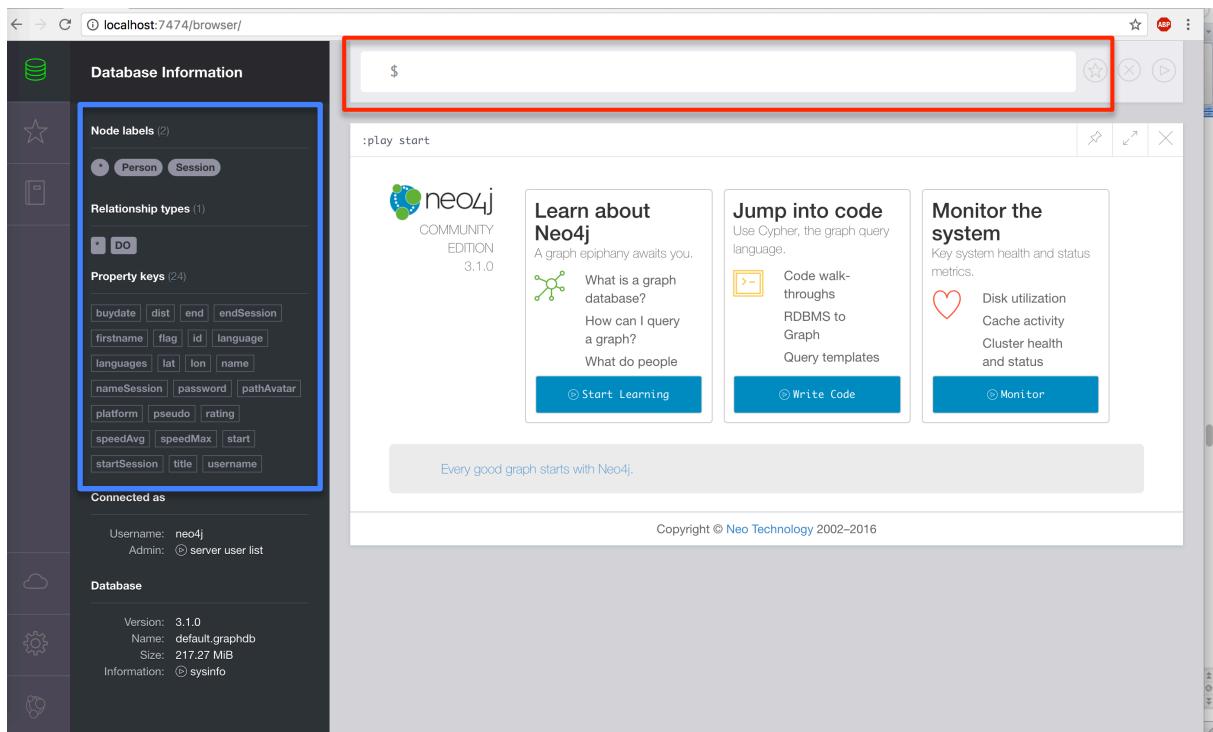


Figure 7: interface de Neo4j

Dans la partie encadrée en rouge de la figure 6, c'est dans cette zone que les utilisateurs de Neo4j peuvent faire leurs requêtes. Dans la partie encadrée en bleu se trouvent les informations du contenu de la base de données (nœuds, relations et propriétés).

## 3.2. Librairies Android

L'utilisation des librairies externes est essentielle pour le développement de l'application. Pour la gestion des librairies, Android utilise Gradle pour la gestion des modules et dépendances de librairies Maven. Pour ajouter des librairies, il faut ouvrir le fichier « build.gradle » qui se trouve dans le répertoire « Gradle Script » d'un projet Android. Pour finir, il suffit d'ajouter le lien de la librairie dans « dependencies ».

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.google.code.gson:gson:2.2.4'
    compile 'com.squareup.okhttp3:okhttp:3.6.0'
    compile 'de.hdodenhof:circleimageview:2.1.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
}
```

Figure 8: Ajout de librairies



Trois librairies externes seront utilisées dans le cadre de ce projet de semestre :

- GSON
- okhttp3
- CircleImageView

D'autres librairies internes seront utilisées :

- USB Host
- Geocoder

### 3.2.1.USB Host

L'API USB Host<sup>1</sup> offre la possibilité de gérer la connexion entre deux périphériques en USB en fournissant plusieurs classes, dont **UsbManager** qui sera utilisé dans le cadre de ce projet. Cette classe permet d'énumérer les périphériques connectés et de communiquer avec des périphériques USB connectés.

### 3.2.2. Geocoder

Geocoder permet de manipuler le géocodage, c'est-à-dire de retourner des informations d'un lieu en indiquant la latitude et la longitude et inversement.

La librairie est utilisée pour la géolocalisation d'une session.

### 3.2.3. GSON

GSON est une librairie Java permettant de gérer et manipuler des données en JSON en les convertissant en des objets Java, et vice-versa.

L'API sera utilisé essentiellement lors des réponses des requêtes HTTP. En effet, les données transférées entre le serveur Web et l'application Android seront en format JSON. GSON permet de faciliter grandement le traitement de données.

### 3.2.4. okhttp3

Cette librairie sera utilisée pour envoyer et recevoir les données demandées par l'utilisateur au serveur Web de ce projet de semestre. La librairie permet de créer le corps de la requête, choisir la méthode (GET, POST). Deux méthodes de callback sont disponibles lors de l'envoi de la requête HTTP (onResponse et onFailure).

### 3.2.5. CircleImageView

CircleImageView est une librairie permettant d'afficher et de gérer les images en les affichant en forme de rond. Cette API est utilisée pour l'affichage de l'avatar d'un utilisateur de l'application et pour la capture d'une photo lors de la création d'un compte.

---

<sup>1</sup> <https://developer.android.com/guide/topics/connectivity/usb/host.html>

## 4. Conception

Dans cette partie de document, la conception de l'application *FieldWiz Monitoring* sera développée. Il y aura le diagramme de cas d'utilisation, les diagrammes de séquence et les maquettes de l'application.

### 4.1. Diagramme de cas d'utilisation

Dans le système, il y a deux types d'acteurs :

Utilisateur non logué : lors du démarrage de l'application, l'utilisateur doit se loguer ou bien créer un compte s'il n'a pas encore de compte. Lors de la création de compte, il doit insérer ses informations personnelles.

Utilisateur logué : Il a la possibilité d'utiliser la totalité des fonctionnalités de l'application, c'est-à-dire l'upload des sessions et la visualisation des sessions. Il peut également se déconnecter.

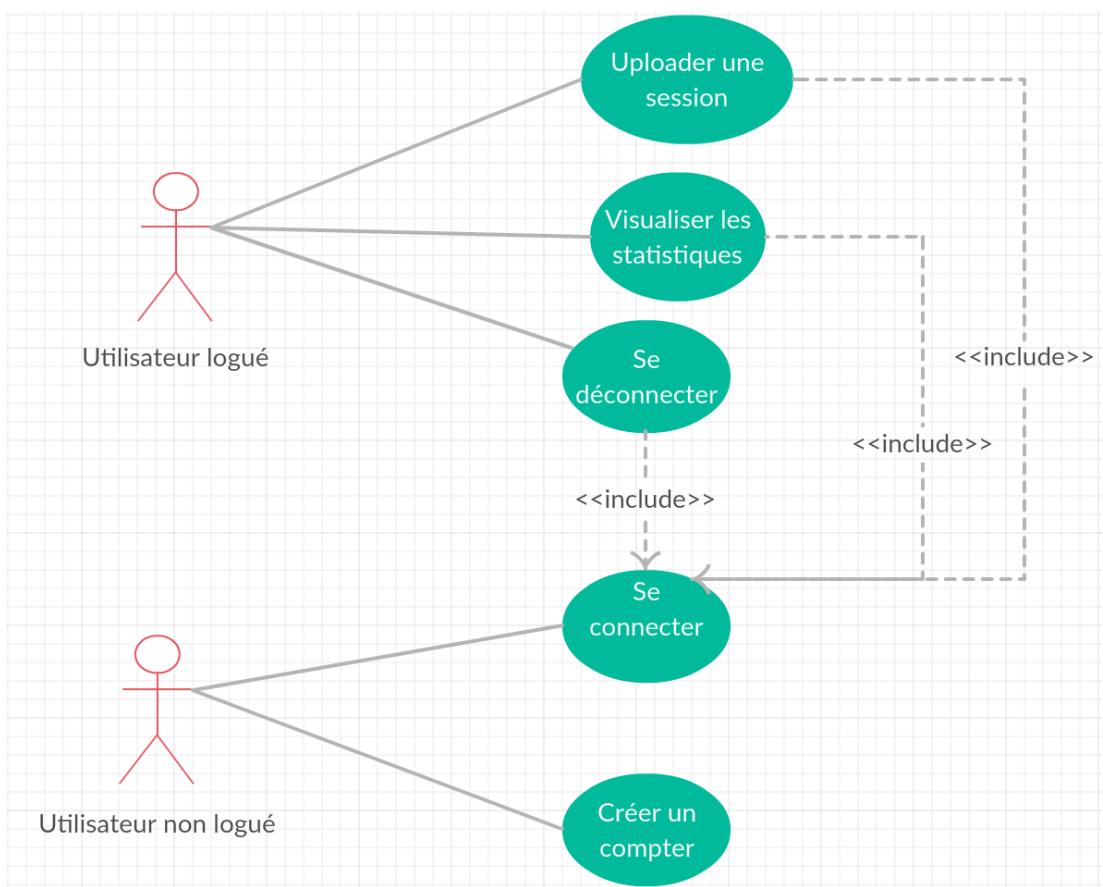


Figure 9: Cas d'utilisation

## 4.2. Diagramme de séquences

Les diagrammes de séquences ont été élaborés suite au diagramme de cas d'utilisation.

### 4.2.1. Se connecter

Au démarrage de l'application, le système demande à l'utilisateur non logué de se connecter au système.

Le système vérifie les informations de login en faisant une requête GET au serveur Web, qui lui va faire un MATCH sur la base de données (MATCH est l'équivalent de SELECT dans le langage SQL).

La base de données répond au serveur Web s'il y a bien un login avec les informations envoyées par l'utilisateur non logué. Sinon un message d'erreur apparaît. Les informations tels que le « username » et le « password » sont conservées en dur dans la base de données.

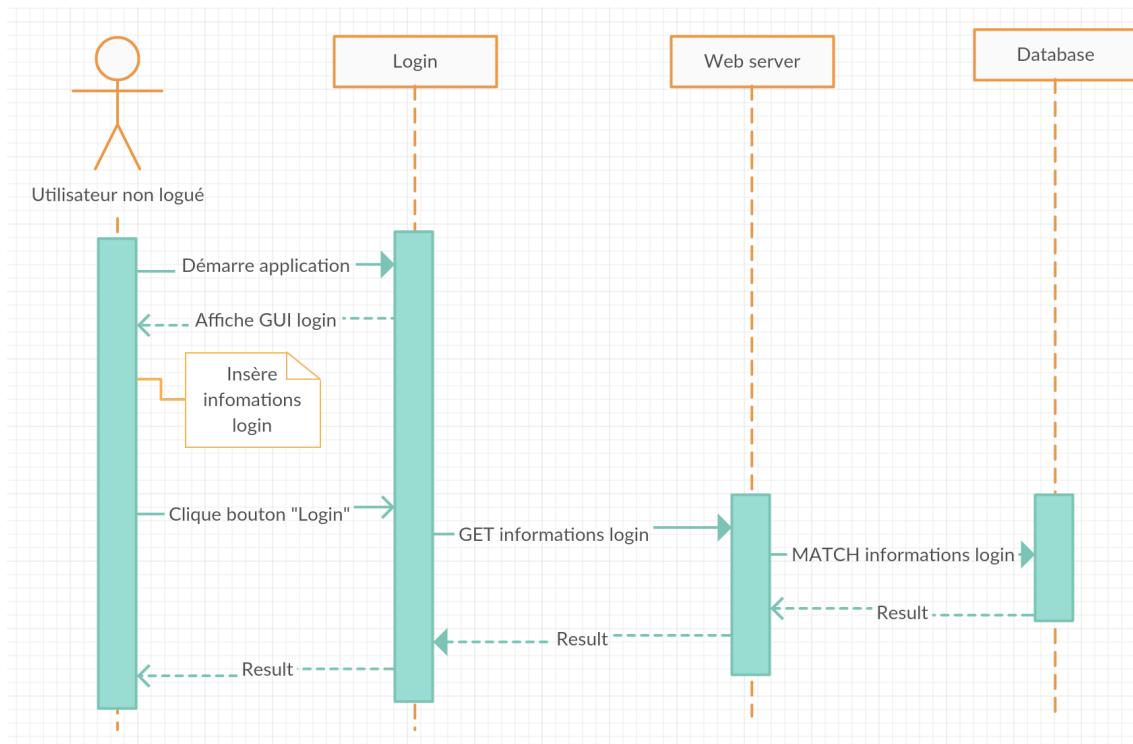


Figure 10: diagramme de séquence "se loguer"

#### 4.2.2. Créer un compte

L'utilisateur, s'il ne possède pas encore de login, peut créer un compte utilisateur. Pour se faire, il suffit d'appuyer sur le bouton « Create account », ce qui a pour effet d'ouvrir une nouvelle page pour la création de login. L'utilisateur insère ses données (possibilité de se prendre une photo, nom d'utilisateur, mot de passe, prénom et nom).

Une fois les informations insérées, l'utilisateur peut appuyer sur le bouton « Validate » ce qui crée le compte. Une méthode POST contenant les informations du nouvel utilisateur est faite sur le serveur Web, qui va lancer la commande CREATE sur la base de données.

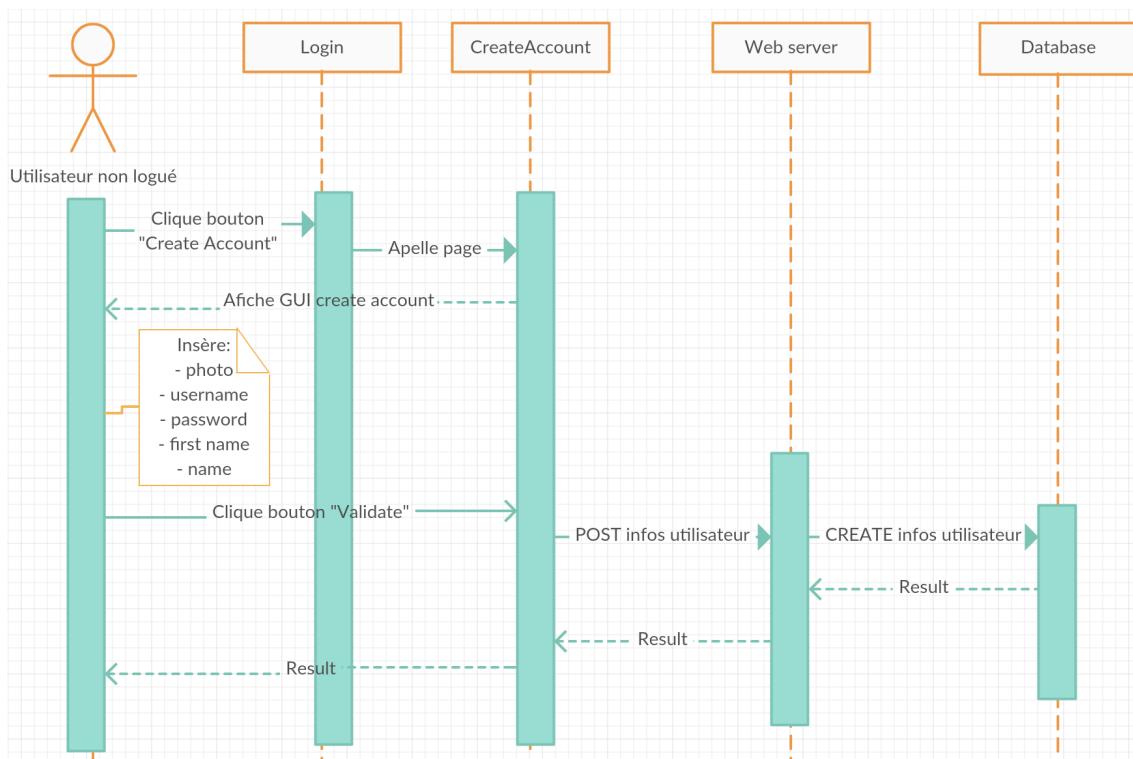


Figure 11: diagramme de séquence "créer compte"



### 4.2.3. Uploader une session

Une fois connecté, l'utilisateur peut uploader une session depuis l'application en appuyant sur le bouton « Upload session » sur la page principale de l'application.

Dans un premier temps, l'utilisateur insère le nom de la session. Puis, si le FieldWiz est branché au smartphone, l'utilisateur choisit un des fichiers avec le format .FWZ parmi la liste. En sélectionnant un fichier, le système va solliciter le serveur Web afin de retrouver toutes les sessions d'un fichier. Le serveur Web va faire appel à un script MATLAB qui va nous retourner la liste de toutes les sessions. Une session a une date de début de session et une date de fin de session. Une fois la liste des sessions en sa possession, le serveur Web envoie au client cette dernière. La liste retournée contient uniquement les sessions qui n'ont pas encore été sauvegardées dans la base de données.

L'utilisateur aura le choix de choisir une session uniquement en sélectionnant un élément dans la liste.

Une fois après avoir sélectionné un élément de la liste, l'utilisateur pourra sauvegarder la session souhaitée en appuyant sur le bouton « Upload ». Cette action aura pour effet de solliciter encore une fois le serveur Web avec les informations de la session. Le serveur Web va pouvoir lancer une requête à la base de données. La requête sera du type « CREATE ». Une session est créée en relation avec le nom d'utilisateur connecté actuellement à l'application.

Un message à l'utilisateur sera retourné pour informer que la session a bien été insérée dans la base de données. La liste des sessions est mise à jour en supprimant la session qui vient d'être ajoutée.

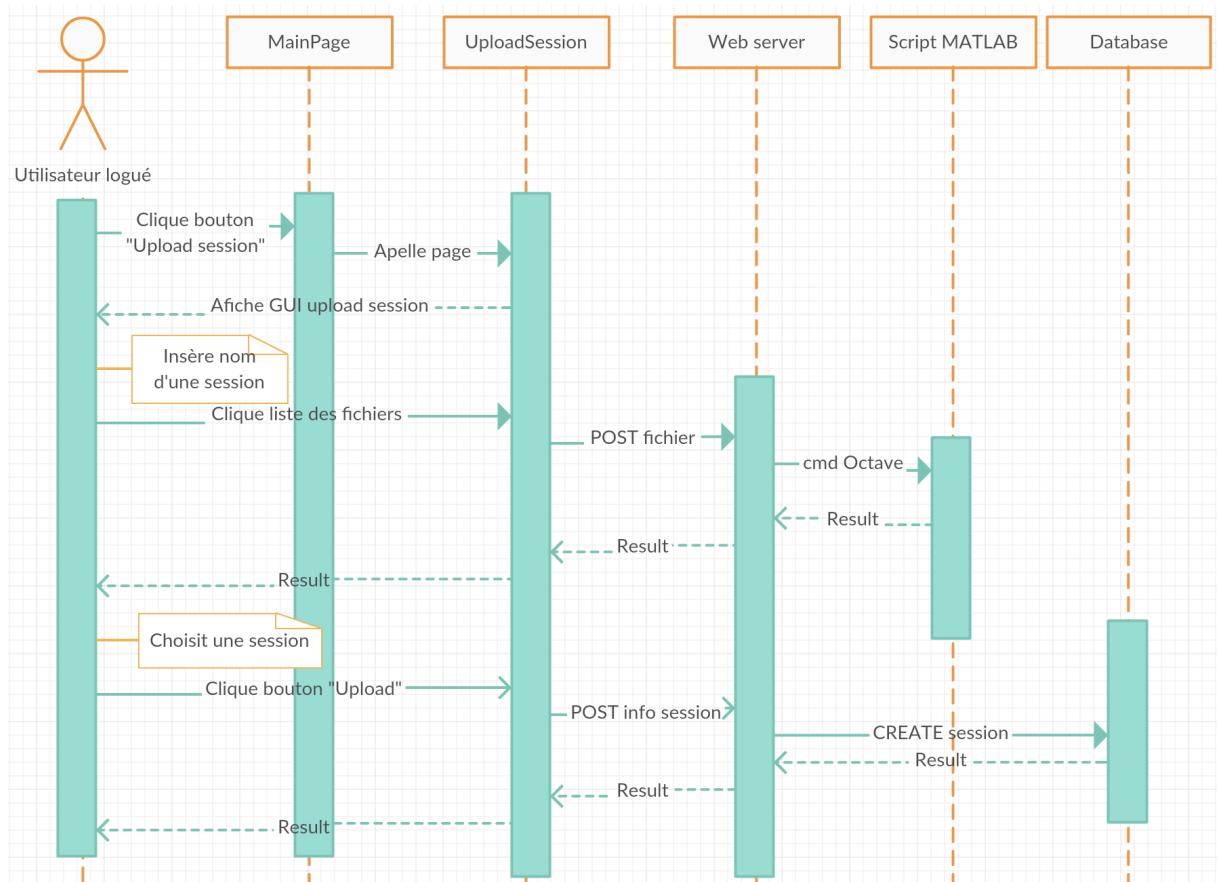


Figure 12: Diagramme de séquence "uploader une session"

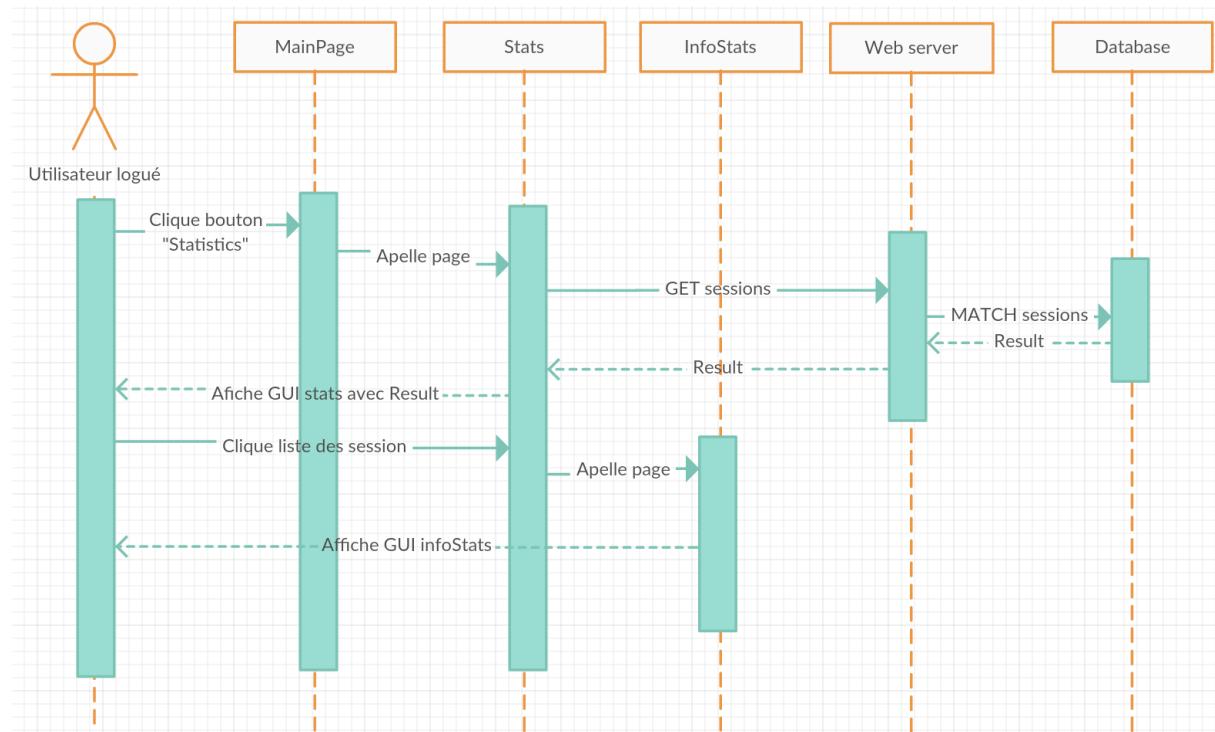
#### 4.2.4. Visualiser les statistiques

Depuis la page principale, l'utilisateur peut également visualiser les statistiques des sessions sauvegardées dans la base de données. Pour ce faire, l'utilisateur doit cliquer sur le bouton « Statistics ».

Une fois sur le bouton appuyé, l'application va faire un appel au serveur Web afin de récupérer toutes les informations des sessions. Le serveur Web va requérir la base de données pour ressortir toutes les sessions de l'utilisateur courant.

Ensuite, l'application affiche la liste des sessions avec le nom de la session, sa date de début et sa date de fin. L'utilisateur aura le choix de visualiser une des statistiques de la liste en cliquant sur une session.

Le système affiche les informations de la statistique sélectionnée, en affichant la localité de la session, la vitesse maximum, la vitesse moyenne et la distance parcourue.



#### 4.2.5. Se déconnecter

Dans la page principale, il est possible de se déconnecter du système. Pour se faire, il faut simplement aller dans le menu d'options, puis cliquer sur « Disconnect ». L'application va retourner un message si l'utilisateur souhaite vraiment se déconnecter ou pas. En cas de déconnection, le système affiche la page de login. En cas de refus, rien ne se passe et l'application continue de tourner normalement.

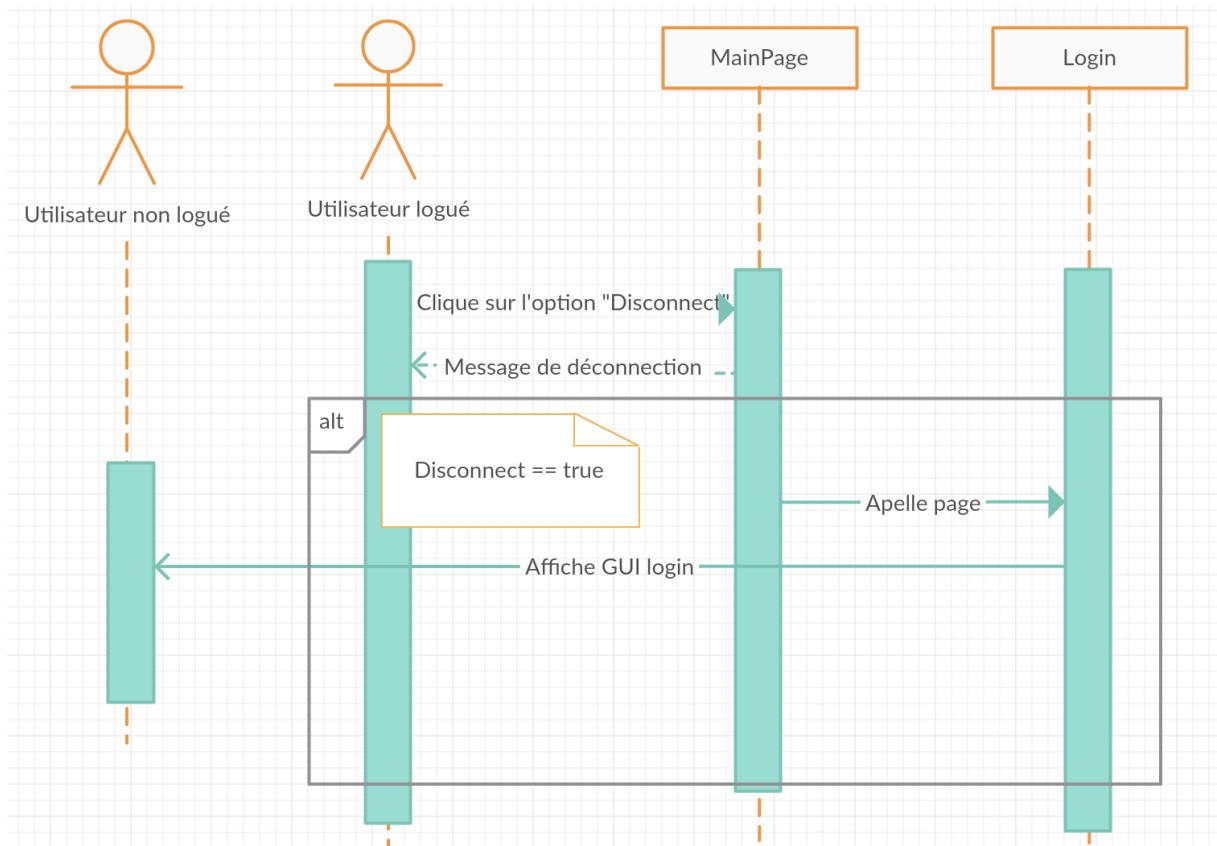


Figure 13: Diagramme de séquence "se déconnecter"

## 4.3. Diagramme d'état-transition

Lors du démarrage de l'application, le système affiche la page de login. L'utilisateur a le choix de créer un nouveau compte ou bien de se loguer. En cas de création de compte, le système affiche la page de la création de compte. Une fois la création de compte terminée, le système affiche la page de login.

Une fois loguée, l'application affiche la page principale. L'utilisateur a le choix de se déconnecter, de voir ses statistiques ou d'uploader des sessions.

Lorsque l'utilisateur se dirige vers l'upload de sessions ou les statistiques, le système affiche la page de l'upload respectivement la page des statistiques. À tout moment, l'utilisateur peut revenir sur la page principale.

La déconnection de compte se fait uniquement sur la page principale. S'il se déconnecte, le système affiche la page de login.

À tout moment, le système peut se fermer, ce qui met fin au cycle de vie de l'application.

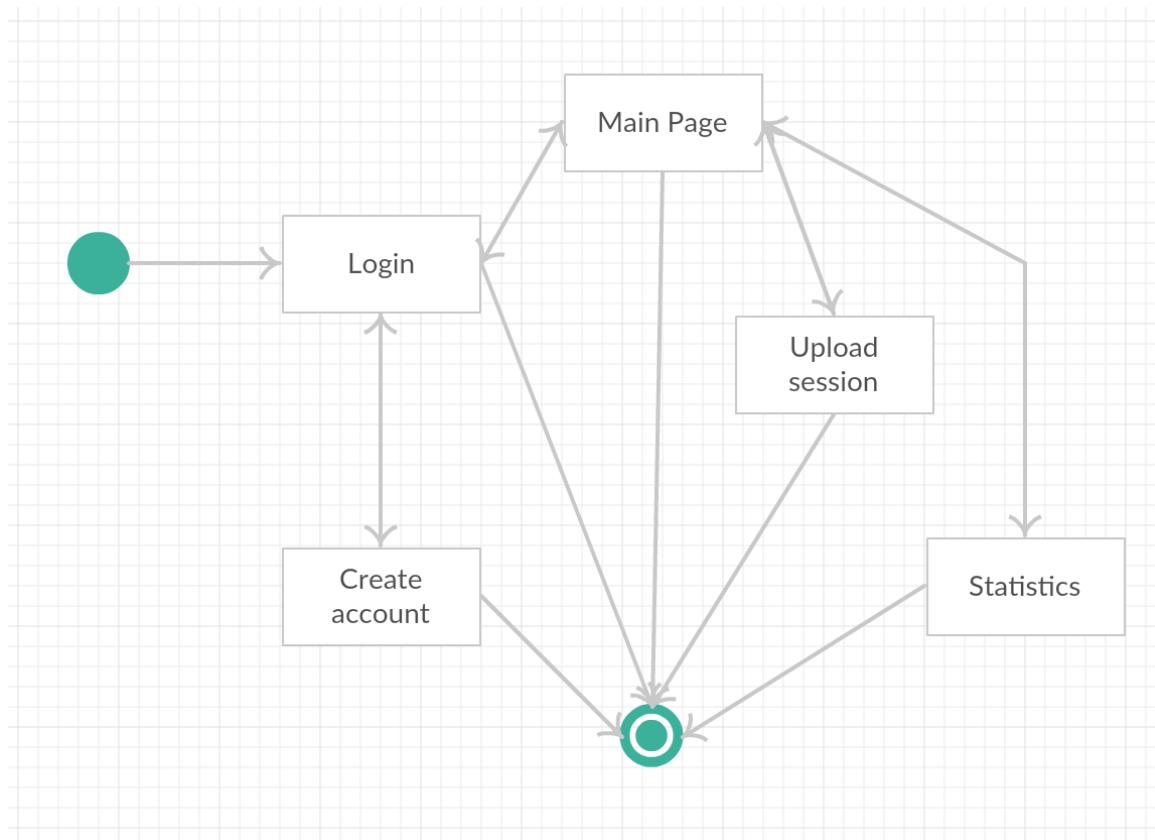


Figure 14: Diagramme d'état-transition

## 4.4. Maquettes

Toutes les interactions de l'application seront décrites dans ce chapitre.

### 4.4.1. Login

Lors du démarrage de l'application, l'utilisateur devra introduire ses informations de login, ou se créer un nouveau compte s'il ne l'a pas encore fait.



Figure 15: Maquette "login"

### 4.4.2. Crédation d'un compte

L'utilisateur a la possibilité de créer un compte afin qu'il puisse utiliser l'application. L'utilisateur doit insérer un nom d'utilisateur, un mot de passe, son nom et son prénom. Il peut également se prendre en photo en cliquant sur l'image de profil.

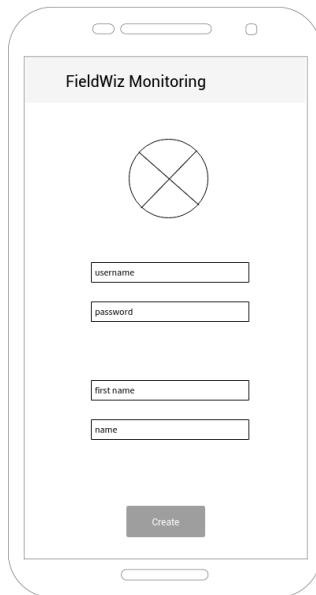


Figure 16: Maquette "création d'un compte"

#### 4.4.3. Page principale

Une fois l'utilisateur logué, il se trouve sur la page principale de l'application. Depuis cette page, il a accès à l'intégralité des fonctionnalités. Pour ce projet de semestre, les fonctionnalités sont :

- Voir les statistiques
- Upload une session

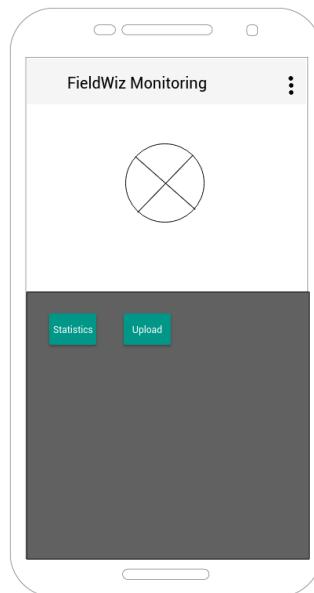


Figure 17: Maquette "page principale"

L'utilisateur a également le choix de se déconnecter via le menu en haut à droite.



#### 4.4.4. Upload

Depuis la page principale, l'utilisateur peut se rendre sur la page de l'upload. Il doit sélectionner un des fichiers .FWZ se trouvant dans le FieldWiz. Ensuite il insère l'heure du commencement de la session et l'heure de la fin de la session.

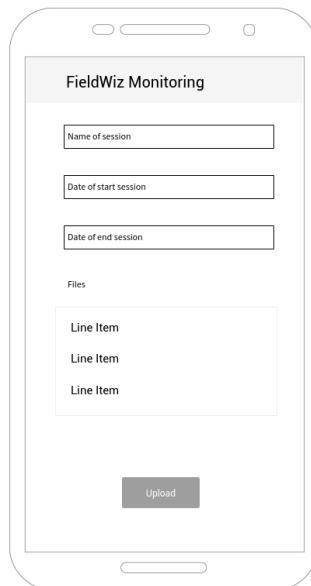


Figure 18: Maquette "upload"

#### 4.4.5. Statistique

Depuis la page principale, l'utilisateur peut se rendre sur la page des statistiques. Il peut visualiser l'historique de l'ensemble de ses statistiques.

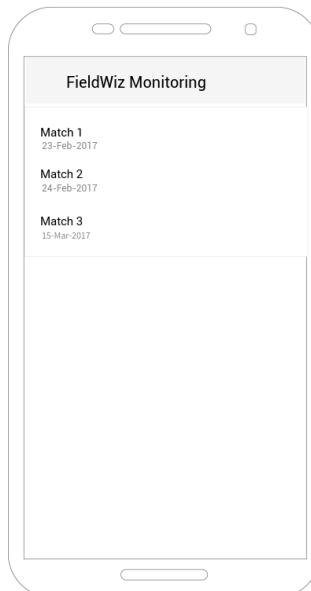


Figure 19: Maquette "Statistiques"

Lorsque l'utilisateur sélectionnera une session, l'application affichera les informations supplémentaires de la session.



Figure 20: Maquette "info statistique"

## 4.5. Requêtes

Différentes requêtes seront implémentées dans ce projet de semestre. Les requêtes seront faites au niveau du client HTTP (application Android) - serveur HTTP (Node.js) et au niveau du serveur HTTP (Node.js) – Base de données (Neo4j).

Les méthodes utilisées pour interagir avec le serveur Node.js sont uniquement les méthodes GET et POST.

### 4.5.1. Contrôle l'authentification d'un utilisateur

Méthode POST qui requête la base de données, retourne un « boolean » si les informations de login sont correctes.

### 4.5.2. Crédation d'un compte (1ère partie: données utilisateur)

Méthode POST qui enregistre dans les variables globales du serveur les informations d'un utilisateur (nom, prénom, nom d'utilisateur et mot de passe).

### 4.5.3. Crédation d'un compte (2ème partie: envoie de l'avatar)

Méthode POST qui possède l'avatar de l'utilisateur et lance la requête de création d'utilisateur dans la base de données.

### 4.5.4. Récupération des infos de l'utilisateur

Méthode GET permettant de récupérer les informations d'un utilisateur (nom et prénom) depuis la base de données.



#### **4.5.5. Récupération de l'avatar de l'utilisateur :**

Méthode GET permettant de récupérer l'avatar d'un utilisateur depuis la base de données.

#### **4.5.6. Récupération des sessions d'un fichier .FWZ**

Méthode POST permettant de rechercher les sessions d'un fichier .FWZ en sollicitant un des scripts MATLAB pour ressortir la liste des sessions.

#### **4.5.7. Récupération des dates de début et de fin de session**

Méthode POST qui enregistre dans les variables globales du serveur les informations d'une session (nom de la session, date de début et date de fin).

#### **4.5.8. Crédit de session**

Méthode POST permettant de créer une session. Au préalable, le script MATLAB calculant les informations d'une session (vitesse max, vitesse moyenne et distance parcourue) est lancée.

#### **4.5.9. Récupération des statistiques d'un utilisateur**

Méthode GET permettant de récupérer les statistiques des sessions d'un utilisateur en sollicitant la base de données.



## 5. Implémentation

Dans cette partie de document, l'implémentation de l'application *FieldWiz Monitoring* sera développée.

### 5.1. Application Android

Dans cette section, les extraits de codes issus de l'application Android est décrit.

#### 5.1.1. Interface de l'application

Toutes les interfaces de l'application ont été conçue de la même manière : utilisation de « `LinearLayout` » comme conteneur en premier niveau. Son utilisation est simple. Voici quelques paramètres principaux utilisés :

- **`android :orientation`** : permet de placer les éléments du conteneur verticalement ou horizontalement
- **`android :gravity`** : permet de modifier la gravité de la façon dont les éléments sont placés
- **`android :layout :width`** : définit la largeur du conteneur (généralement l'utilisation totale de la largeur est faite : **`match_parent`**)
- **`android :layout_height`** : définit la hauteur du conteneur

En ce qui concerne le jeu de couleur utilisé, le noir, le blanc et le vert ont été utilisés. Ce sont les couleurs utilisées par le site Web officiel de FieldWiz<sup>2</sup> le noir (couleur hexadécimal : #302F2E) est utilisé pour le fond des conteneurs et des boutons. Le vert (couleur hexadécimal : #00CC00) est utilisé pour les images et le surlignement des champs texte. Le blanc est utilisé pour les écritures sur fond noir et sur les fonds des conteneurs.

La langue de l'application est uniquement en anglais. Il n'y a aucune option concernant le changement de langue.

Les images utilisées dans l'application proviennent du site Web officiel de FieldWiz. L'utilisation de l'image d'un avatar sans photo de Google a été utilisée lors de la création de compte.

Le menu des options n'est visible que sur la page principale, et se trouve en haut à droite de la page. Seule la déconnection n'est pas possible.

<sup>2</sup> <http://www.fieldwiz.com/>



## 5.1.2. Activités

Les activités représentent chaque page de l'application *FieldWiz Monitoring*. Il y en a en tout six :

- Login
- Création de compte
- Page principale
- Upload
- Liste des statistiques
- Informations sur une statistique

Lors de la transition entre les pages « login-page principale », « page principale-upload » et « page principale-liste des statistiques », on passe le nom de l'utilisateur :

```
intent = new Intent(this, StatsActivity.class);
intent.putExtra("username", username);
startActivity(intent);
```

L'exemple ci-dessus illustre la façon de passer un « String » à d'autres activités. Dans ce cas, on passe de la page principale à la page de la liste des statistiques. La méthode « putExtra » permet d'insérer tout type d'objet qui hérite de « Serializable ». Dans ce cas, on passe un « String » avec une clé. Cette clé est utilisée dans l'activité appelée.

Pour récupérer la valeur passée dans la méthode « putExtra », il faut utiliser la méthode « getStringExtra » dans l'activité appelée :

```
username = getIntent().getStringExtra("username");
```

L'exemple ci-dessus illustre l'affectation du nom d'utilisateur (avec la clé « username » utilisée précédemment) à la variable de type « String » « username ».

### 5.1.3.Client HTTP

A plusieurs reprises, l'application doit communiquer avec le serveur Web. La librairie « okhttp3 » permet d'implémenter des fonctionnalités afin de requêter avec Node.js. Ci-dessous, un exemple sur l'utilisation de cette librairie :

```
OkHttpClient client = new OkHttpClient();
HashMap<String, String> listInfo = new HashMap<String, String>();
listInfo.put("username", editTextUsername.getText().toString());
listInfo.put("password", editTextPassword.getText().toString());
String json = new Gson().toJson(listInfo);
MediaType JSON= MediaType.parse("application/json; charset=utf-8");
RequestBody body = RequestBody.create(JSON, json);
Request request = new Request.Builder()
    .url(getString(R.string.POST_USER))
    .post(body)
    .build();
client.newCall(request).enqueue(new okhttp3.Callback() {
    @Override
    public void onFailure(okhttp3.Call call, IOException e) {
        Log.i("Fail", e.getMessage());
    }

    @Override
    public void onResponse(okhttp3.Call call, okhttp3.Response response) throws IOException {
        Log.i("Response", "Response");
        boolean isConnected =
Boolean.parseBoolean(response.body().string());
        if(isConnected)
            startMainActivity();
        else{
            showErrorMessage();
        }
    }
});
```

En premier lieu, on crée les objets suivants :

- Client : le client http
- listInfo : HashMap contenant les information de login
- json : contient la HashMap formaté en JSON
- body : le corps de la requête http, il faut passer le JSON avec les informations de login
- request : la requête contenant toutes les informations nécessaires, y compris la méthode utilisé (GET, POST)



Pour finir, on appelle la méthode « newCall » qui va lancer la requête au serveur Web. La méthode « enqueue » permet de se mettre dans une file d'attente. En paramètre, les méthodes de callback sont créées :

- onFailure
- onResponse

En cas de réussite (méthode « onResponse » lancée), on reçoit depuis le serveur l'information de la réussite de la connexion en « String » (côté serveur, le texte envoyé doit impérativement être « true » ou « false »). Il faut le convertir en boolean, et teste celui-ci si la connexion a réussi ou non.

Pour l'envoi d'un multimédia, le « body » de la méthode doit être différents. En effet, on n'envoie pas un JSON.

JSON :

```
RequestBody body = RequestBody.create(JSON, json);
```

Multimédia :

```
RequestBody requestBody = new MultipartBody.Builder()  
    .setType(MultipartBody.FORM)  
    .addFormDataPart("file", "file",  
        RequestBody.create(MultipartBody.FORM, new  
File(selectedFile)))  
    .build();
```

Les méthodes « setType » et « addFormDataPart » permettent de modifier le type respectivement d'ajouter le fichier.

Important : lors du lancement des méthodes de callback, ces derniers ne sont pas exécutés par le thread principal, ce qui a pour effet de ne pas avoir la possibilité de modifier le GUI. Il y a un moyen pour palier à ce problème. Il s'agit de forcer le thread principal à se lancer. Ci-dessous un exemple :

```
this.runOnUiThread(new Runnable() {  
    @Override  
    public void run() {  
        // exécution du code dans le thread principal  
    }  
});
```

Cette méthode d'utilisation est également utilisée lorsqu'il faut appeler une méthode de callback provenant d'un bouton ou autres éléments de l'application.



## 5.1.4. Formatage en JSON

Lorsque le serveur nous répond, il est possible de récupérer des données en format JSON. L'extrait ci-dessous illustre la façon de récupérer les données depuis le serveur Web :

```
Gson gson = new Gson();
try(Reader reader = response.body().charStream()){
    tabStats = gson.fromJson(reader, Stats[].class);
}
reloadData();
```

L'objet « gson » permet de formater et de transformer les données JSON. Dans ce cas, on a un objet « tabStats » qui est un tableau de « Stats ».

L'objet « Stats » est composé de :

- nameSession : nom de la session
- startSession : date de début de la session
- endSession : date de fin de la session
- speedMax : vitesse max
- speedAvg : vitesse moyenne
- dist : distance parcourue
- lat : latitude d'un point
- lon : longitude d'un point

Important : Le JSON envoyé par le serveur Web doit impérativement contenir le même nombre de champs, avec les mêmes noms de champs et la même hiérarchie. Dans ce cas, le JSON doit être de ce type :

```
[  
    "nameSession" : ...  
    "startSession" : ...  
    "endSession" : ...  
    "speedMax" : ...  
    "speedAvg" : ...  
    "dist" : ...  
    "lat" : ...  
    "lon" : ...  
]
```

## 5.1.5. Capture de photos

Lors de la création de compte, il est possible de se prendre en photo pour compléter son profil. L'extrait ci-dessous illustre la façon de faire :



```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
}
break;
```

L'exemple montre qu'il faut créer une nouvelle activité de capture de photo. Lorsque l'application lance l'activité, il faut obligatoirement lancer avec la méthode « `startActivityForResult` », afin de lancer la méthode « `onActivityResult` » ci-dessous pour récupérer le résultat de l'activité de capture de photo.

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode ==
RESULT_OK) {
        Bundle extras = data.getExtras();
        imageBitmap = (Bitmap) extras.get("data");
        imageViewAvatar.setImageBitmap(imageBitmap);
    }
}
```

L'exemple illustre la façon d'afficher une photo prise à l'utilisateur.

### 5.1.6. Affichage d'informations

Dans l'application, l'utilisateur est à quelques reprises informé sur l'état d'une action. Voici dans les cas où l'utilisateur est informé :

- mauvais login / déconnection
- ajout d'une session
- Indisponibilité de sessions

#### Mauvais login / déconnection :

Pour ces deux cas, l'utilisateur est informé par un message de type « `AlertDialog` ». Ces alertes sont affichées en premier plan à l'utilisateur et demande une interaction avec celui-ci. Ci-dessous un extrait de code pour le mauvais login :

```
AlertDialog.Builder alertDialogError = new
AlertDialog.Builder(LoginActivity.this);
alertDialogError.setTitle("Incorrect login");
alertDialogError.setMessage("The username or password is
incorrect.");
alertDialogError.setPositiveButton("Ok",
    new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {}
    });
alertDialogError.show();
```



### Ajout d'une session :

Pour ce cas, un message plus discret est utilisé. Le message est du type « Toast ». le message apparaît durant un certain temps, et disparaît ensuite. Ci-dessous un exemple d'utilisation :

```
Toast toast = Toast.makeText(getApplicationContext(), "Upload successful",  
Toast.LENGTH_SHORT);  
toast.show();
```

### Indisponibilité de sessions :

Un simple « TextView » doit être implémenté pour ce cas. Il apparaît lorsque le système ne trouve plus de sessions dans le fichier .FWZ.

### 5.1.7. Broadcast receiver

Les « broadcast receiver » sont des diffusions d'événements, ce qui permet d'envoyer ou de capturer des événements système. Dans le cadre de ce projet de semestre, les « broadcast receiver » seront utilisés pour savoir si un appareil est connecté au smartphone via le port USB.

Lors de la création du « broadcast receiver », une méthode de callback est créé : « onReceive(Context context, Intent intent) ». Dans l'objet « intent », il est possible de connaître le genre d'action que subit le smartphone. Il est donc possible de récupérer l'action lorsque l'utilisateur branche le FieldWiz ou le débranche.

UsbManager est une librairie fournit par le SDK Android permettant de gérer les périphériques USB. Pour connaître l'action d'un « intent », il faut appeler la méthode « getAction() ». L'action qui génère le branchement et le débranchement sont :

```
UsbManager.ACTION_USB_DEVICE_ATTACHED // branchement  
UsbManager.ACTION_USB_DEVICE_DETACHED // débranchement
```

### 5.1.8. Recherche de fichier « .FWZ »

Lors de l'upload d'une session, il faut rechercher les fichiers .FWZ se trouvant dans l'arborescence du smartphone et du FieldWiz connecté. Voici un algorithme de recherche de fichiers avec l'extension « .FWZ » :

```
public void walkdir(File dir) {  
    String pdfPattern = ".FWZ";  
    File listFile[] = dir.listFiles();  
    if (listFile != null) {  
        for (int i = 0; i < listFile.length; i++) {  
            if (listFile[i].isDirectory()) {  
                walkdir(listFile[i]);  
            } else {  
                if (listFile[i].getName().endsWith(pdfPattern)){  
                    arrayList.add(listFile[i].getPath());  
                }  
            }  
        }  
    }  
}
```



Cet algorithme récursif permet de récupérer l'ensemble des fichiers d'un type quelconque d'un dossier donné en paramètre. Dans le cas du projet, il faudra appeler deux fois cette méthode : pour le parcours du Pod FieldWiz et du « sdcard » (l'environnement de l'utilisateur d'un smartphone Android)

### 5.1.9. Gestion des ListView

Sous les systèmes Android, la « **ListView** » est la liste « physique » qui se crée à partir du fichier XML. C'est ce que les utilisateurs vont voir depuis l'écran de leur smartphone. Cependant, pour afficher quelque chose dans la « **ListView** », il faut lui fournir des données. Ces données sont contenues dans des **tableaux** (HashMap, ArrayList, tableaux statiques). Pour insérer les données du tableau dans la « **ListView** » il faut adapter les données en utilisant un **adapter**. Voici un extrait de code :

```
adapter = new ArrayAdapter<Stats>(getBaseContext(),
    android.R.layout.simple_list_item_2, android.R.id.text1,
tabStats) {
    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {
        View listView = super.getView(position, convertView, parent);
        TextView text1 = (TextView)
listView.findViewById(android.R.id.text1);
        TextView text2 = (TextView)
listView.findViewById(android.R.id.text2);
        text1.setText(tabStats[position].getNameSession());
        text2.setText(tabStats[position].getStartSession() + " | "
+ tabStats[position].getEndSession());
        return listView;
    }
};
listViewStats.setAdapter(adapter);
```

adapter : objet qui permet d' « adapter » les données dans la ListView

tabStats : tableau contenant les données

listView : la ListView dans lequel les données vont apparaître

### 5.1.10. Geocoder

« Geocoder » permet de ressortir la localité selon une position dans la carte du monde, en lui fournissant la latitude et la longitude.

L'extrait de code ci-dessous permet de géolocaliser une position :



```
Geocoder geocoder = new Geocoder(this, Locale.getDefault());
ArrayList<Address> localization = new ArrayList<Address>();
try {
    localization = (ArrayList<Address>)
geocoder.getFromLocation(stats.getLatitude(),stats.getLongitude(),1);
} catch (IOException e) {
    e.printStackTrace();
}

String locality = localization.get(0).getLocality()
```

Le contenu de « locality » contient la localité suivant la latitude et la longitude données.

## 5.2. Node.js

Dans cette section, les subtilités utilisées dans le serveur Web Node.js sont décrites.

### 5.2.1. URL avec le nom d'utilisateur

Dans la plupart des méthodes utilisées dans le script, il est nécessaire d'envoyer le nom d'utilisateur dans l'URL de la requête. En effet, cette information va permettre de gérer les comptes.

L'extrait ci-dessous est la méthode GET pour la récupération des statistiques d'un utilisateur :

```
app.get('/GetStats/:username/', function(req, res){
...
})
```

Lorsque le client va faire appel à cette méthode, il doit passer un URL du type « /GetStats/daniel ». Dans ce cas, pour récupérer la valeur « daniel », il faudra utiliser l'objet req, avec les paramètres suivants :

```
req.params.username
```

A partir de ce point, on peut requérir la base de données en dépendant d'un nom d'utilisateur.

### 5.2.2. Retour d'un JSON

Il est conseillé de retourner un format JSON au client (et recevoir aussi). JSON présente de nombreux avantages :

- facile à lire les informations à l'œil
- léger
- utiliser dans de nombreux langages (dont Javascript et Android)



Dans ce cas, étant donné que le retour d'un résultat d'une requête Neo4j fournit un format JSON qui ne convient pas, il faut créer son propre format.

L'extrait suivant permet de retourner du JSON au client :

```
res.setHeader('Content-Type', 'application/json');
res.send(JSON.stringify(resultStats, null, 3));
```

On indique le « header » que c'est du JSON qui est envoyé.

### 5.2.3. Réception de fichiers

Il est nécessaire, dans certains cas, de récupérer un fichier. Dans ce projet, il faut récupérer l'avatar ou le fichier .FWZ :

```
app.post('/', multer({ dest: 'files'}).single('file'),
function(req,res){
...
})
```

Le module « multer » de Node.js permet de gérer les fichiers qui sont envoyés depuis le client HTTP. Node.js transforme le nom de fichier en une suite de lettres et de chiffre à 32 caractères.

Le nom du fichier se trouve en utilisant l'objet « req » :

```
req.file.filename
```

### 5.2.4. Exécution des commandes

Il est possible d'exécuter des commandes du terminal. Le module « exec » permet de le faire. Dans le cadre de ce projet, il faut lancer les commandes :

- octave : pour exécuter les scripts
- mkdir : pour créer les répertoire des utilisateur
- mv : pour déplacer l'avatar dans le répertoire de l'utilisateur

Ci-dessous un exemple avec la commande octave :

```
cmd = 'octave --silent --eval \"readFwz(\'' + req.file.filename
+ '\',\' + startSession+'\',\'' + endSession+'\',\' + username +
'\')\"';
exec(cmd, function(error, stdout, stderr) {
...
})
```



Important: Il faut faire très attention concernant l'utilisation des caractères tels que les guillemets simples et les double guillemets.

### 5.2.5. Envoie de l'avatar

Il est possible d'envoyer les images depuis le serveur Node.js vers le client HTTP, en connaissant l'URL de l'image. Le module « fs » de Node.js permet de gérer les images. L'extrait ci-dessous montre comment gérer cela :

```
var img = fs.readFileSync(result.records[0].get("pathAvatar"));

res.writeHead(200, {'Content-Type': 'image(bitmap' });

res.end(img, 'binary');
```

La variable image contient une image. Dans ce cas, il contient l'avatar d'un utilisateur. Il faut également indiquer à l'en-tête qu'il faut envoyer un bitmap.

## 5.3. Neo4J

Les requête Neo4j sont lancées depuis le serveur Web, les deux types de requêtes utilisés sont uniquement **MATCH** (équivalent au SELECT dans SQL) et **CREATE**.

### 5.3.1. MATCH

Voici un exemple d'une requête pour la récupération de session :

```
MATCH (a:Session)-[r:DO]-(p:Person{username: 'daniel'})
RETURN a.startSession AS startSession
```

La requête retourne toutes les sessions dont le nom d'utilisateur est « daniel » possédant une relation « DO », uniquement les paramètres « startSession » et « endSession ».

### 5.3.2. CREATE

Voici un exemple de création d'une personne dans la base de données :

```
CREATE (a:Person {username: 'dada', password: '1234', firstname: 'daniel', name: 'garcia', pathAvatar: '/Users/dada/img'})
```

La requête crée un nouveau nœud « Person » avec les paramètres se trouvant entre accolades.

## 5.4. Script MATLAB

Deux scripts MATLAB sont utilisés dans ce projet de semestre.

### 5.4.1. Liste des sessions

Le premier script permet de retourner toutes les sessions d'un fichier .FWZ. La société ASI fournit le script permettant de non seulement décoder le fichier, mais calcule et trouve également les sessions. Les informations étant confidentielles, seules les parties produits par le fruit du travail seront illustrées.



Le code ci-dessous permet de sortir le début et la fin de chaque session :

```
for...
    disp(datestr(time0));
    ...
    disp(datestr(time(numel(time))));
end
```

La fonction « disp » permet de sortir un résultat dans la console. « time » est un vecteur contenant tous les temps d'une session. La fonction « numel » permet de ressortir le nombre d'élément d'un tableau. La fonction « datestr » permet de convertir une date en un format de type **DD-MMM-YYYY HH :MM :SS**. Le format des dates dans le vecteur time est un float à 5 décimales. Pour représenter une seconde dans ce format, il s'agit du nombre 0.00001. Ce format de date est basé sur le 1<sup>er</sup> janvier, à l'an 0 (et non le 1<sup>er</sup> septembre 1970).

### 5.4.2.Statistiques

Le deuxième script permet de calculer les valeurs d'une session. Les valeurs calculées sont la vitesse max, la vitesse moyenne et la distance parcourue. Il faut passer en paramètre le chemin du fichier, la date de début de la session, la date de fin de la session et le nom d'utilisateur.

Dans un premier temps, il est nécessaire de récupérer la tranche de la session grâce au paramètre du début et de la fin de session.

```
startRow=[];
endRow=[];
startSessionNum = datenum(startSession);
endSessionNum = datenum(endSession);
while isempty(startRow)
    startRow = find(abs(time-startSessionNum) < 0.0001);
    startSessionNum = addtodate(startSessionNum, 1, 'second');
end
while isempty(endRow)
    endRow = find(abs(time-endSessionNum) < 0.0001);
    endSessionNum = addtodate(endSessionNum, 1, 'second');
end
```



Le but est de retrouver l'index du vecteur quand commence la session et l'index quand se termine la session. Pour retrouver le temps, il faut parcourir le vecteur « time ». Les valeurs fournies par le FieldWiz peuvent présenter des défaillances, il faut se permettre une marge de 10 secondes (en datenum : 0.001) lorsqu'on utilise la fonction « find ».

Une fois l'index de début et l'index de fin trouvés, il est possible de calculer les informations voulues :

```
speed = speed(startRow:endRow,:);  
  
speedMax = max(speed(:));  
  
speedAvg = sum(speed) / (endRow - startRow);  
  
dist = 0;  
  
for i = 1:endRow-startRow;  
    dist = dist + speed(i) * 0.1 / 3600;  
end
```

Dans un premier temps, on réduit la taille du vecteur « speed » qui représente toutes les vitesses (en km/h) de la session courante. Pour trouver la vitesse max, il faut simplement ressortir le maximum d'un vecteur, avec la fonction « max ». Pour la vitesse moyenne, il faut faire la somme de toutes les vitesses, et diviser cela par le nombre d'éléments du vecteur. Pour la distance, il faut additionner les distances d'une entrée. Sachant que le FieldWiz possède une fréquence de 10Hz, la distance (en km) est égale à la multiplication de la vitesse (en km/h) et du nombre d'entrée que fait le FieldWiz en une heure, soit la fréquence (10 Hz) / 3600 secondes.

Une fois les calculs réalisés, il faut ressortir les valeurs en utilisant la fonction « disp » :

```
disp(speedMax);  
  
disp(speedAvg);  
  
disp(dist);  
  
disp(allValues(1,1));  
  
disp(allValues(1,2));
```

Le script ressort également la latitude et la longitude d'un point (pour calculer la géolocalisation côté client).



Pour finir, le script enregistre toutes les valeurs de la session dans un fichier CSV. Ces données seront exploitées dans d'autres projets futurs :

```
startSession = strrep(startSession,'-','_');
startSession = strrep(startSession,' ','_');
startSession = strrep(startSession,:,'_');
pathDir = strcat("Users/", username, "/", startSession, ".csv");
delete(filename);
csvwrite(pathDir, allValues);
```

Le nom du fichier CSV a un nom du type « username\_date\_debut\_session ». Le paramètre « username » est utilisé pour indiquer dans quel répertoire enregistrer le fichier. Il est également nécessaire de supprimer le fichier .FWZ, au risque d'avoir le serveur rempli de fichier .FWZ.

## 6. Tests

Dans cette partie de document, les tests de l'application *FieldWiz Monitoring* seront développés. Deux types de tests ont été réalisés:

- Tests fonctionnels
- Test utilisateur

### 6.1. Tests fonctionnels

Les tests fonctionnels permettent de détecter d'éventuels problèmes dans l'application. Toutes les fonctionnalités sont testées.

#### 6.1.1. Crédation de comptes

Test n°1 : création d'un compte normal, tout est à zéro :

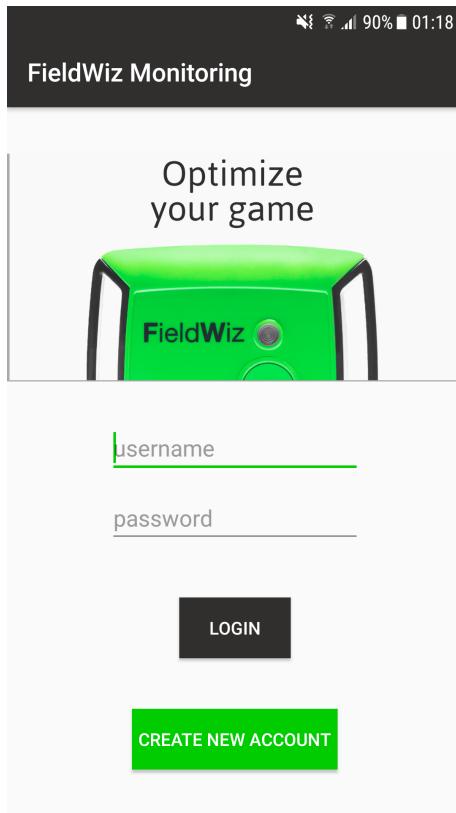


Figure 21: création de comptes

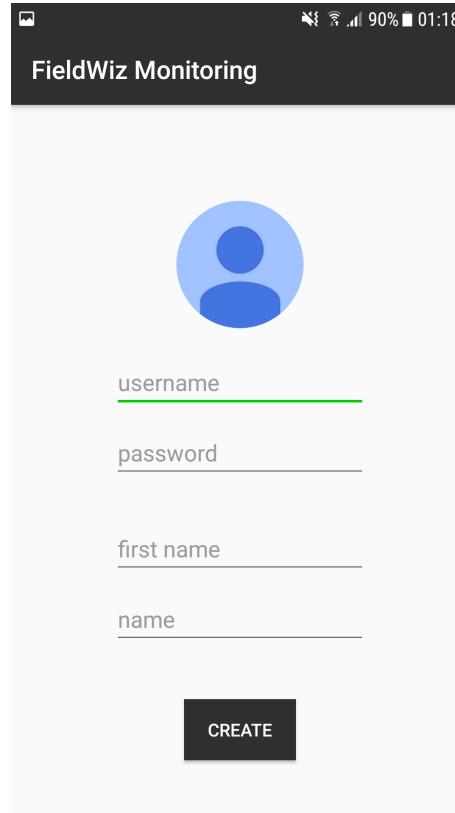


Figure 22: ouverture de l'application

L'utilisateur appuie sur le bouton « Create new Account » et remplit les informations d'un utilisateur (photo, nom d'utilisateur, mot de passe, prénom et nom).



Figure 23: champs complétés

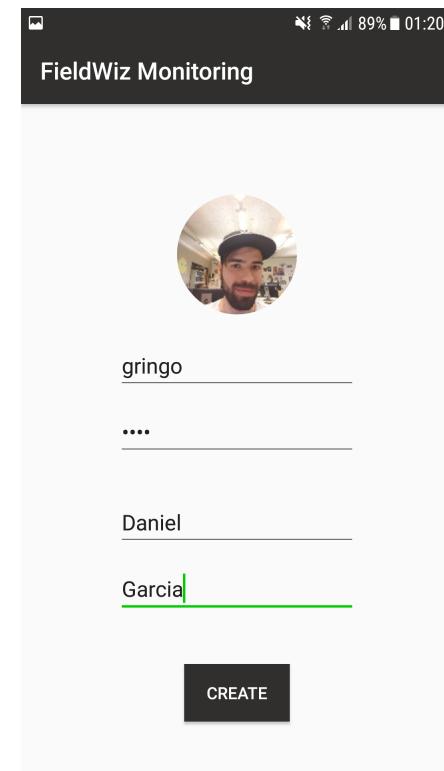


Figure 24: Prise de photo

Une fois les champs remplis, l'utilisateur appuie sur le bouton « Create ».

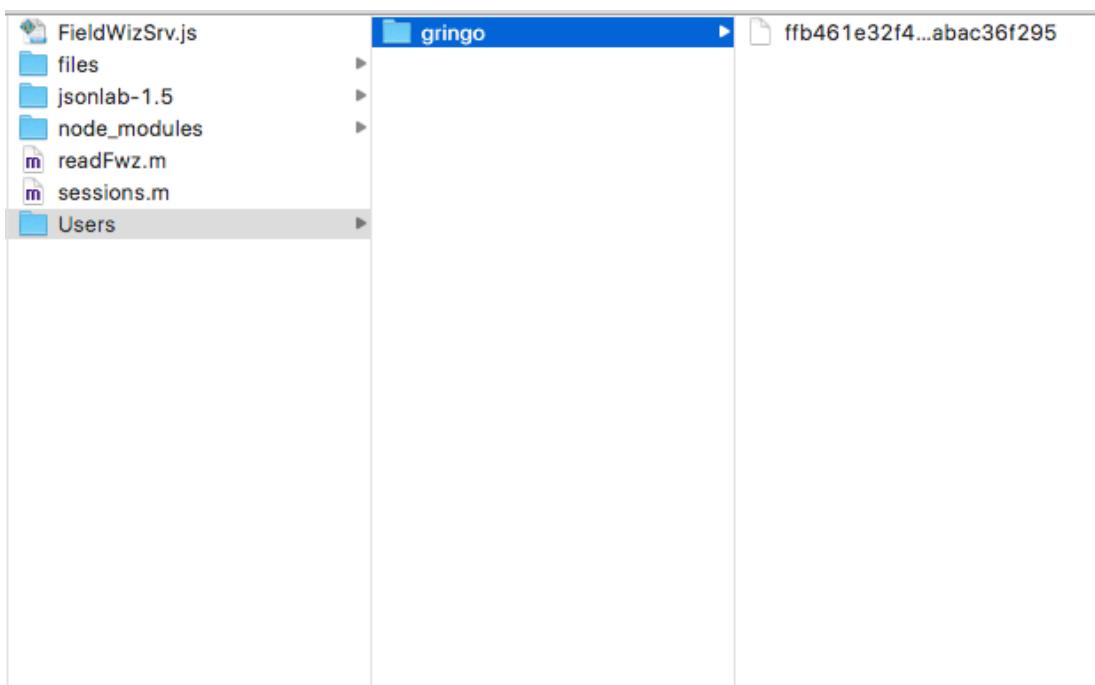


Figure 25: répertoire de l'utilisateur créé, avec l'avatar



\$ MATCH (n) RETURN n LIMIT 25

\*(1) Person(1)

Graph Rows Text Code

Person <id>: 3135 firstname: Daniel password: 1234 name: Garcia pathAvatar: Users/gringo/ffb461e32f44a38c7bd45cabac36f295

Figure 26: Noeud créé avec les bon paramètres

En appuyant sur le bouton « Create », cela va créer un répertoire de l'utilisateur et un nœud « Person » est créé dans la base de données.

#### Test n°2 : création avec le même nom d'utilisateur



Figure 27: Deux noeud identiques

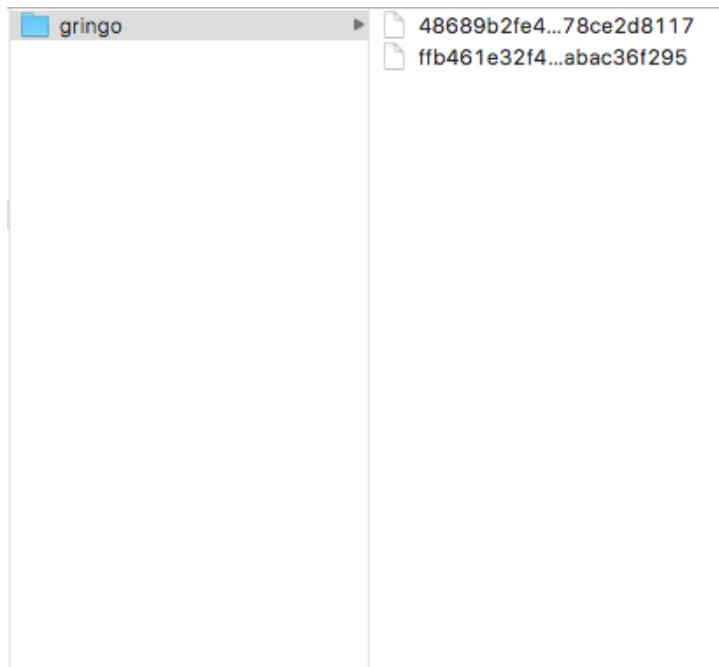


Figure 28: Deux avatars pour un utilisateur

Cela risque de créer de gros problèmes plus tard, lors de la connexion avec cet utilisateur. Le nom d'utilisateur n'est pas unique.

### Test n°3 : aucune insertion d'utilisateur

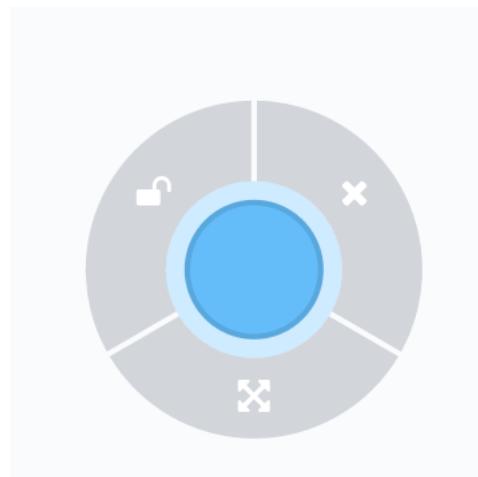


Figure 29: Noeud créé

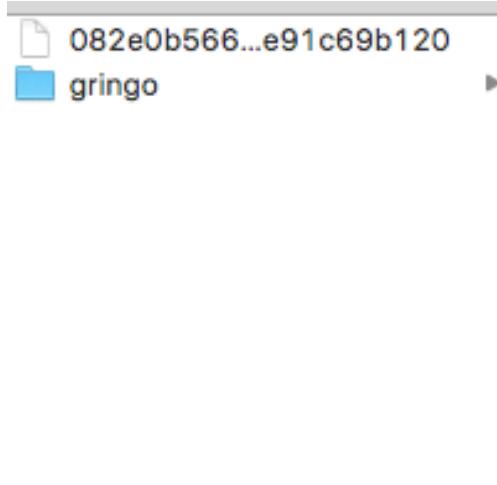


Figure 30: pas de dossier créé

### Récapitulation des tests

Test n°1	OK ✓
Test n°2	NOK ✗
Test n°3	NOK ✗

### 6.1.2.Connexion

#### Test n°1 : connexion normale :

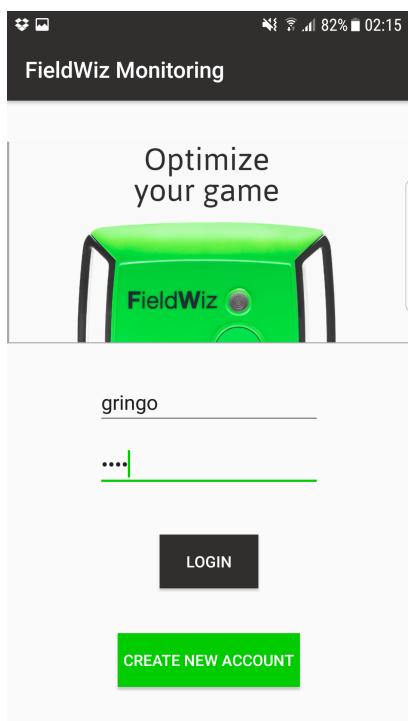


Figure 31: login

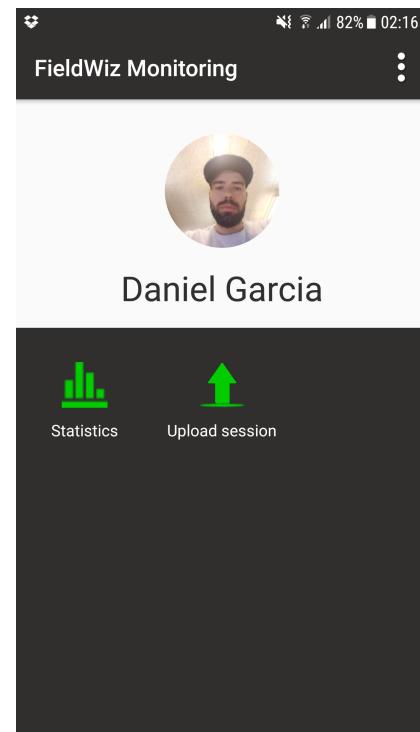
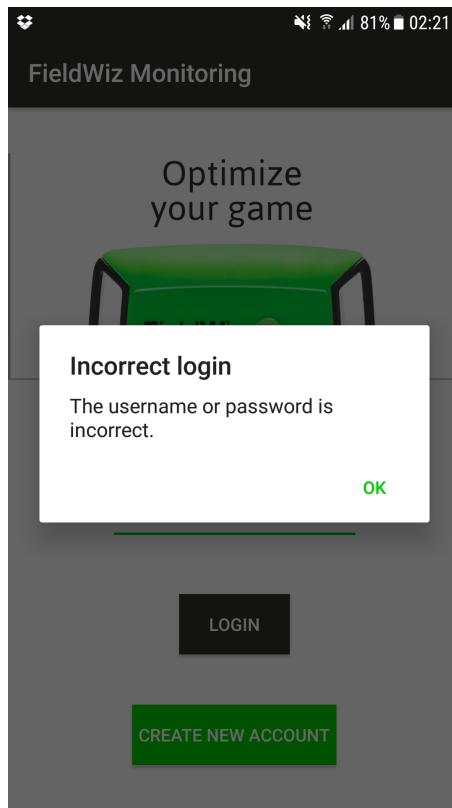


Figure 32: utilisateur logué



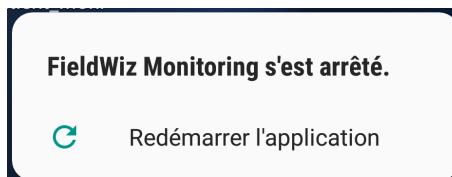
**Test n°2 : faux login :**



**Figure 33: faux login**

L'utilisateur insère un login qui n'est pas existant dans la base de données.

**Test n°3 : aucune information (suite à la création d'un login sans information) :**



**Figure 34: message du système**

**Récapitulation des tests**

Test n°1	OK ✓
Test n°2	OK ✓
Test n°3	NOK ✗

**Remarque :** le premier test est passé, bien qu'il y ait deux utilisateur du même nom.



### 6.1.3.Upload des sessions

#### Test n°1 : cas normal

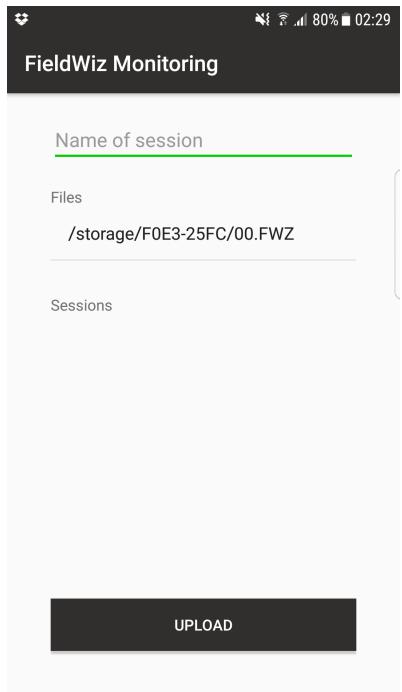


Figure 35: ouverture de la page

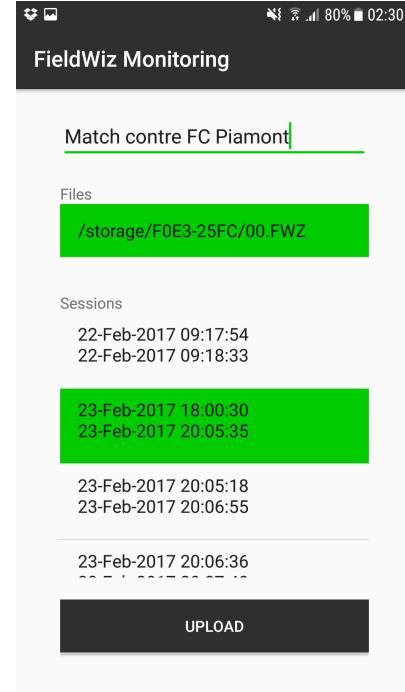


Figure 36 : recherche de sessions

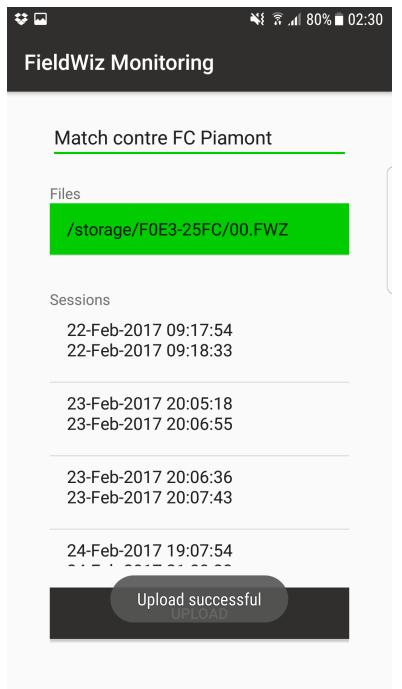


Figure 37: upload réussi



### Test n°2 : ajout d'une session avec le même nom :

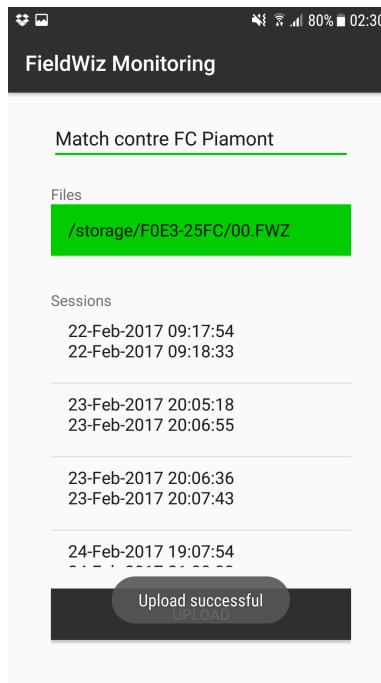


Figure 38: upload réussi

Malgré le même nom de session, l'application autorise l'upload.

### Test n°3 : ajout de toutes les sessions disponibles

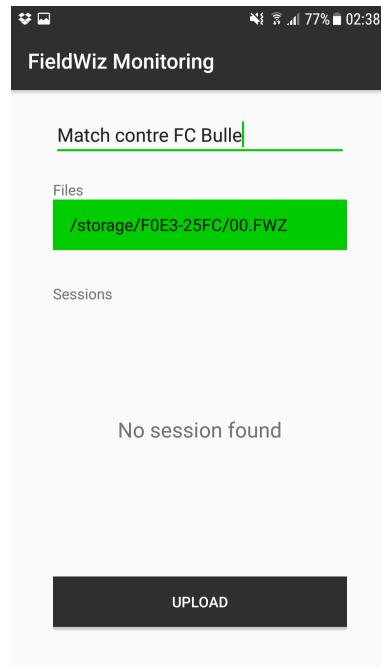


Figure 39: aucune session disponible

## Récapitulation des tests

Test n°1	OK ✓
Test n°2	NOK ✗
Test n°3	OK ✓

**Remarque :** ayant deux utilisateurs avec le même nom d'utilisateur, les sessions ont été créées deux fois. Cela va se répercuter lors de la visualisation des statistiques.

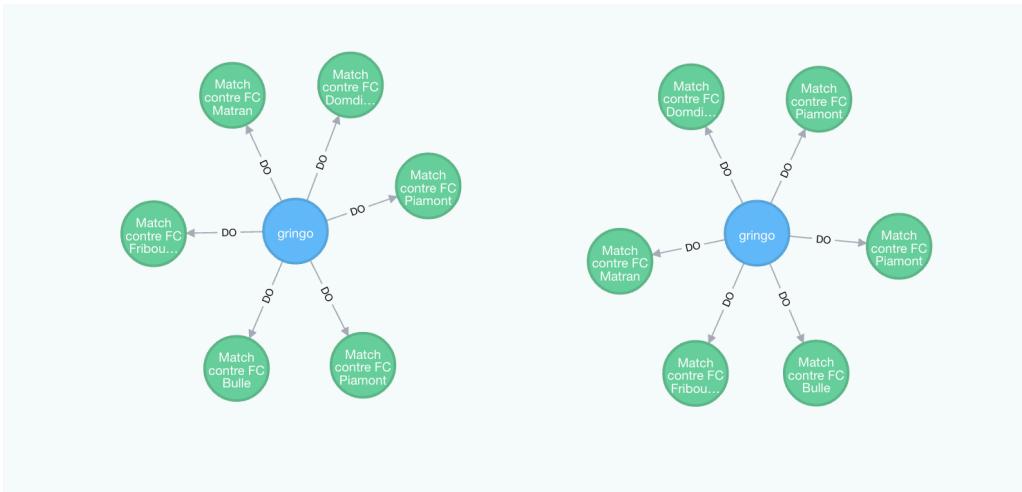


Figure 40: noeud avec les mêmes sessions

### 6.1.4. Visualisation des données

#### Test n°1 : cas normal :



Figure 41: liste des sessions

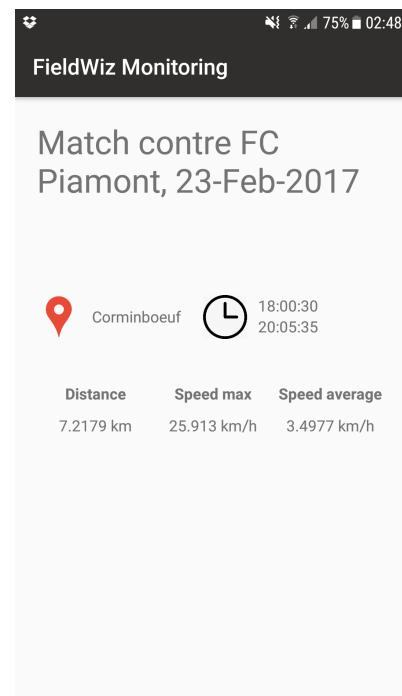


Figure 42: affichage d'une session

## Récapitulation des tests

Test n°1

OK ✓

**Remarque :** Le test passe, cependant, ceux qui n'ont pas passé auparavant ont eu des répercussions sur la liste des sessions.

En temps normal, avec un seul nom d'utilisateur par utilisateur, l'affichage de la liste des statistiques et des informations sur une statistique fonctionne parfaitement. Le seul souci est le formatage des nombres petits (écriture en puissance de 10) dans l'affichage des statistiques de vitesse ou distance.

### 6.1.5. Déconnexion

**Test n°1 : déconnexion depuis le menu des options :**

Il est possible de se déconnecter de deux manières : soit l'utilisateur se rend sur le menu des options, et choisit l'option « disconnect », soit l'utilisateur appuie sur le bouton « back » présent sur les smartphones Android.

Le cas ci-dessous ne teste que si l'utilisateur appuie sur le bouton « Yes », le bouton « No » n'a aucun effet, ce qui est l'effet voulu.

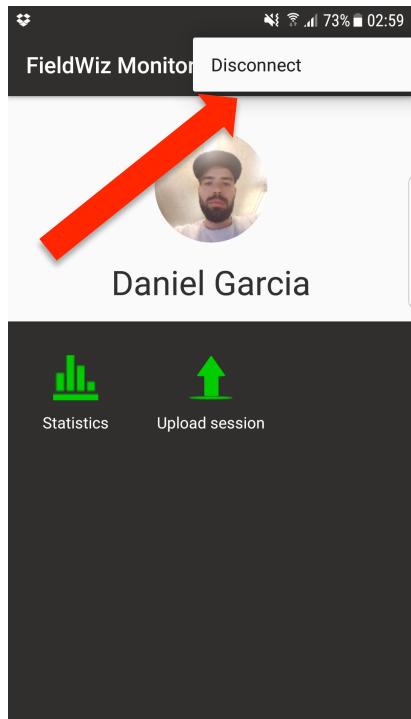


Figure 44: menu pour se déconnecter

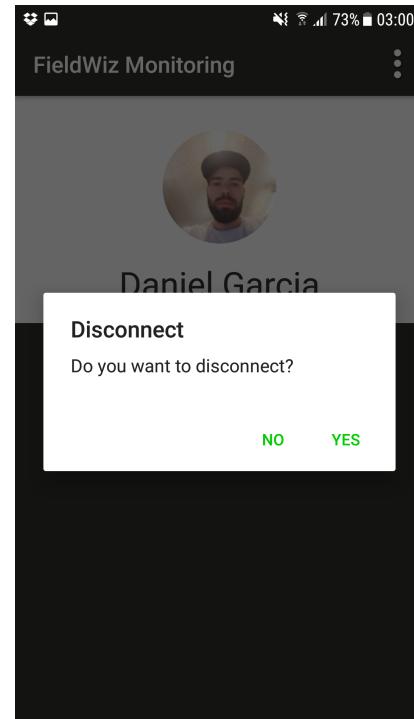


Figure 43: Message d'information



## Récapitulation des tests

Test n°1	OK ✓
----------	------

### 6.1.6. Synthèse des tests

On peut constater que les tests ne fonctionnant pas ou partiellement sont dû aux saisies des données par l'utilisateur. En effet, il n'y a pratiquement aucun contrôle de saisie. L'utilisateur a le choix de mettre ce qu'il veut, ou ne rien mettre. Le fait de ne rien mettre ou de mettre deux fois le même nom va repousser les problèmes ailleurs.

Sinon, en temps normal avec des données normales, la totalité des fonctionnalités fonctionnent parfaitement.

## 6.2. Tests utilisateur

Les tests utilisateurs permettent de faire tester l'application directement à des personnes physiques. Ces personnes donnent leur feedback du système. Voici la donnée :

*« Vous êtes joueur d'un club de football local, et vous avez deux sessions à sauvegarder : le 23 février 2017, de 18:00:30 à 20:05:35 et le 24 février, de 19:07:54 à 21:00:33. Les matchs joués sont contre le FC Domdidier respectivement contre le FC Corminboeuf. »*

Ce test a été effectué sur trois personnes dont l'anonymat est conservé : Damien<sup>3</sup>, 24 ans, Pedro<sup>4</sup>, 24 ans et Sandrine<sup>5</sup>, 25 ans

### 6.2.1. Feedback

Les trois personnes ayant testé l'application *FieldWiz Monitoring*, voici leur feedback

#### Damien :

Selon Damien, l'application est bien réussie. Le fil rouge de l'application bien respecté (création login – connexion – upload – visualisation des données). La photo de profil n'a pas été prise.

Il n'a eu aucun problème concernant le test, tout s'est bien déroulé. Aucun bug n'est à déplorer.

#### Pedro :

Selon Pedro, il fallait lui dire qu'il fallait créer un compte. En effet, il n'a pas tout de suite compris qu'il devait créer un login afin d'utiliser l'application. Il a été guidé afin de créer un login. La suite n'a pas été meilleur : en allant sur l'upload des sessions, il a directement appuyé sur le bouton « upload » sans information. Cela a fait planté l'application. Il en avait marre et a décidé d'arrêter le test.

<sup>3</sup> Nom d'emprunt

<sup>4</sup> Nom d'emprunt

<sup>5</sup> Nom d'emprunt



Le test n'a pas été un grand succès avec Pedro.

#### Sandrine :

Selon Sandrine l'interface est bien réussie, les couleurs de l'application sont belles. Pour la création de login, pas de soucis. Par contre, une fois loguée, elle ne savait pas quoi faire (ourtant la donnée est claire). Elle a d'abord essayé de voir sur les statistiques. Bien évidemment, la liste est vide. Ensuite elle va sur l'upload des sessions, et c'est là qu'elle a compris le fil rouge. Là aussi, comme Pedro, elle a appuyé sur le bouton upload, ce qui a fait crashé l'application.

Une intervention a dû faire face afin de lui guider la suite des opérations. Elle se logue, elle se rend sur l'upload des sessions, et sauvegarde les sessions en question. Ensuite elle va consulter ses statistiques.

### 6.2.2. Synthèse

Malgré le peu de personnes avoir testé l'application, on peut déjà tirer des conclusions et relever les points négatifs de l'application, les points à améliorer.

Ce qui a été constaté :

- Aucune des personnes n'a eu l'idée de se prendre en photo lors de la création du compte.
- Pas mal de perte de temps sur la page principale : les bouton « upload session » et « statistics » ne paraissent pas « cliquable ».
- Le bouton « upload » peut faire planter l'application lorsqu'aucune donnée n'est fournie.
- Personne ne s'est déconnecté.
- Lors des saisies des champs texte, il faut constamment appuyer sur le bouton « back » afin de quitter le clavier, ce qui ont agacés les trois testeurs.

## 6.3. Améliorations

Suites aux tests, plusieurs améliorations peuvent être apporté au système actuel

### 6.3.1. Saisie du clavier

Une solution doit être trouvé afin que le clavier disparaît lorsque l'utilisateur du système à finit la saisie d'un champ

### 6.3.2. Contrôle des saisie

Un contrôle sur les saisies doit être fait, sur tous les champs de l'application

### 6.3.3. Unicité

L'utilisateur doit être unique dans la base de données. Il faut qu'il ne soit pas possible d'avoir deux fois le même nom d'utilisateur



Il faut également que le nom de la session soit unique pour un utilisateur.

#### 6.3.4. Blocage des boutons

Il faut impérativement bloquer les boutons tant que tous les champs ne sont pas remplis, ou trouver une autre solution afin que tous les champs soient remplis.

#### 6.3.5. Rendre cliquable ce qui est cliquable

Les boutons « statistics » et « upload session » ne paraissent pas cliquables, il faut faire en sorte qu'ils le soient. Il faut également trouver une solution pour la capture de la photo de profil. Lors des tests, il a été constaté qu'aucun des participants n'a pris de photo.

#### 6.3.6. Affichage de messages

Il manque un message pour informer à l'utilisateur qu'il n'y a pas de session enregistrée dans la base de données, lorsque l'utilisateur se rend dans les statistiques.



## 7. Problèmes rencontrés

Durant ce projet de semestre, plusieurs problèmes ont été rencontrés durant la phase d'implémentation.

### 7.1. Application Android

#### 7.1.1. Impossibilité de lire le fichier .FWZ

Les systèmes Android ne permettent pas de lire les fichiers MATLAB (extension .m). Il y avait plusieurs choix pour pallier ce problème. Deux solutions ont été retenues :

- conversion du fichier .m en un fichier exécutable par le système Android
- exécution du script côté serveur

La deuxième solution a été choisie pour des raisons de gain de temps, et surtout parce c'est la solution que FieldWiz opte pour le chargement d'une session depuis le navigateur Web.

#### 7.1.2. Format JSON

Le format JSON envoyé depuis le serveur Web doit exactement convenir au « DataModel ». Le JSON créé par les réponses provenant Neo4J n'étaient pas convenable au « DataModel ».

Afin de résoudre ce problème, un formatage des données a dû être fait, côté serveur.

#### 7.1.3. Gestion des thread

Il faut bien distinguer le thread qui est lancé par l'application principale, et ceux qui sont lancés par les callback. En effet, les callback ne peuvent pas effectuer de modifications sur le GUI.

## 7.2. Node.js

### 7.2.1. Exécution des commandes

Un problème au niveau de l'exécution des commandes octave a été rencontré. En effet, le module utilisé était « execa ». Ce module proposait de lancer des commandes avec la syntaxe suivante :

```
execa('octave-cli-4.2.1', '--eval 'readFwz files/' +
req.file.filename + " " + startSession + " " + endSession + "'"
).stdout.pipe(process.stdout);
```

La commande a été passé en premier lieu, puis la suite de la commande. Il s'avère qu'avec cette méthode, il n'était pas possible d'exécuter octave. La solution était d'utiliser un autre module : « exec » qui lui permet de mettre directement dans un format « String » (Javascript pas typé) la commande tout entière. Mais là aussi, il y avait des problèmes d'exécution. Aucun retour sur la commande. Pourtant, la

commande était correcte (testé sur l'environnement Octave). De plus, il n'était pas possible de passer des paramètres.

Par la suite, MATLAB a été installé. Les commandes fonctionnaient. Cependant, lors de l'affichage des résultats, il fallait formater un texte assez long (618 caractères). De plus, MATLAB est bien plus long à exécuter des programmes qu'Octave. En effet, MATLAB est bien plus lourd, mais plus complet qu'Octave, mais les fonctionnalités d'Octave pour ce projet de semestre sont suffisantes.

De retour avec Octave, les commandes passent sans problème. Suite aux consultations des commandes des premiers essais et ceux actuellement, il y avait en effet des erreurs d'échappement des guillemets simples et des doubles guillemets. Cela est étrange qu'il n'y a aucun retour des erreurs des commandes sur Node.js.

### 7.2.2. Retour des résultats Neo4j

Le problème est mineur mais mérite son attention : le format JSON des résultats avec Neo4j est d'une façon peu convenable pour l'utilisation sur le client. Il ne faut pas oublier de formater les données avant de les envoyer au client.

## 7.3. Script MATLAB

### 7.3.1. Retour des session

Lors de la recherche des indexées pour délimiter la session, la commande « find » ne suffit pas. En effet, les sessions renvoyées par le premier script n'apparaissent pas forcément dans la liste de toutes les données. C'est très étrange mais en effet, la liste des sessions possède un bug. Il y a des chevauchements de dates :

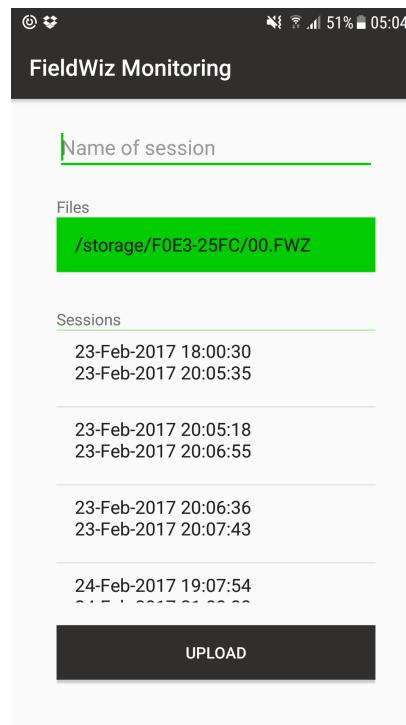


Figure 45: chevauchements de dates



On peut constater que sur la liste des sessions, la date de fin de la première session a une date plus récente que la date de début de la deuxième session. Pareil entre la deuxième session et la 3<sup>ème</sup> session.

### 7.3.2. Format des dates

Les types de formats de dates sur MATLAB sont différents. Le vecteur « time » fournit ce type de dates basées sur un « float » à 5 décimales. Il fallait trouver un moyen de convertir cette date en un format de date du type DD-MMM-YYY HH :MM :SS. les fonctions **datestr()** et **datenum()** permettent de convertir.



## 8. Conclusion

Globalement, ce projet de semestre est une réussite. Malgré quelques petits soucis dans l'implémentation, l'application fonctionne très bien, les objectifs principaux ont été accomplis.

Dès le départ du projet, le problème de la lecture du fichier .FWZ a été relevé. Des solutions ont rapidement été trouvées par les superviseurs.

Le planning n'a, malheureusement, pas toujours été suivi à la lettre. La cause est bien sûr la surcharge de travail dans les autres cours, ce qui a eu un bouleversement dans les semaines P7 et P8. Peu de travail pour ce projet de semestre n'a été fourni. Malgré ces problèmes, il n'y a pas eu de retard au final.

Les compétences utilisées dans ce projet de semestre ont été acquises et sont maîtrisées, à savoir :

- Environnement Android
- Environnement Node.js
- Script MATLAB
- Base de données Neo4j

Cette application a un réel potentiel devant lui, et mérite d'être continuée. Un des projets Bachelor proposé par la Haute école d'ingénierie et d'architecture de Fribourg est la suite de ce projet.

*« Je, soussigné, Daniel Garcia Barros, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteurs ont été clairement mentionnées. »*

Daniel Garcia Barros



## 9. Références

### 1. Documentation

- [1] Jean-François Pillou, « *USB* », 2015,  
<http://www.commentcamarche.net/contents/773-usb#normes-usb>
- [2] Vito, « *Tuto : Qu'est-ce que l'USB OTG ?* », 2015,  
<http://www.frandroid.com/blogs/vitostech/2015/06/118-tuto-usb-otg>
- [3] <https://developer.android.com/guide/topics/connectivity/usb/host.html>
- [4] Andr.oid Eric, « *Android code sample : usb, UBS Host Mode* », 2013,  
<http://android-er.blogspot.ch/2013/10/list-attached-usb-devices-in-usb-host.html>
- [5] OkHttp, <http://square.github.io/okhttp>
- [6] Mathieu Nebra, « *Des applications ultra-rapides avec Node.js* »,  
<https://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js>
- [7] Module Node.js, <https://www.npmjs.com/>
- [8] Florent Champigny, « *Maitriser Gradle* » , <http://www.tutos-android.com/maitriser-gradle-partie-1>
- [9] Throrin, « *Mon retour sur le passage de PHP à Node.js* » ,  
<http://www.throrinstudio.com/dev/web/mon-retour-sur-le-passage-de-php-a-node-js/>
- [10] Houda Chabi Drissi & Benoit Perroud, « *Introduction au NoSQL* », PDF du cours NoSQL
- [11] Android Developer, « *USB HOST* » ,  
<https://developer.android.com/guide/topics/connectivity/usb/host.html>
- [12] JakeWharton, « *A Java serialization/deserialization library to convert Java Objects into JSON and back* » , <https://github.com/google/gson>
- [13] Android Developper, « *Geocoder* » ,  
<https://developer.android.com/reference/android/location/Geocoder.html>
- [14] MATLAB Developper, « *datenum* » ,  
[https://ch.mathworks.com/help/matlab/ref/datenum.html?searchHighlight=datenum&s\\_tid=doc\\_srchtitle](https://ch.mathworks.com/help/matlab/ref/datenum.html?searchHighlight=datenum&s_tid=doc_srchtitle)



## 2. Figures

Figure 1 : Architecture générale de l'application .....	8
Figure 2: un FieldWiz et un gilet.....	9
Figure 3: Exploration du Pod FieldWiz sur un smartphone (Samsung S7).....	10
Figure 4: Logo USB .....	10
Figure 5: exemple d'un graphe Neo4j .....	13
Figure 6: Ouverture de Neo4j (Mac).....	14
Figure 7: interface de Neo4j.....	15
Figure 8: Ajout de librairies .....	15
Figure 9: Cas d'utilisation .....	17
Figure 10: diagramme de séquence "se loguer" .....	18
Figure 11: diagramme de séquence "créer compte".....	19
Figure 12: Diagramme de séquence "uploader une session" .....	21
Figure 13: Diagramme de séquence "se déconnecter" .....	23
Figure 14: Diagramme d'état-transition .....	24
Figure 15: Maquette "login" .....	25
Figure 16: Maquette "création d'un compte" .....	26
Figure 17: Maquette "page principale".....	26
Figure 18: Maquette "upload" .....	27
Figure 19: Maquette "Statistiques" .....	27
Figure 20: Maquette "info statistique" .....	28
Figure 21: création de comptes      Figure 22: ouverture de l'application.....	44
Figure 23: champs complétés      Figure 24: Prise de photo .....	45
Figure 25: répertoire de l'utilisateur créé, avec l'avatar .....	45
Figure 26: Noeud créé avec les bon paramètres .....	46
Figure 27: Deux noeud identiques.....	46
Figure 28: Deux avatars pour un utilisateur.....	47
Figure 29: Noeud créé .....	47
Figure 30: pas de dossier créé .....	48
Figure 31: login      Figure 32: utilisateur logué.....	48
Figure 33: faux login .....	49
Figure 34: message du système .....	49
Figure 35: ouverture de la page      Figure 36 : recherche de sessions.....	50
Figure 37: upload réussi.....	50
Figure 38: upload réussi .....	51
Figure 39: aucune session disponible .....	51
Figure 40: noeud avec les mêmes sessions .....	52
Figure 41: liste des sessions      Figure 42: affichage d'une session.....	52
Figure 45: chevauchements de dates.....	58