Aymane Bouziane, Basile Dyment

RAPPORT DE PROJET

Chef de projet: Gérald Dherbomez

Sommaire

Remerciements

Nous souhaitons remercier Mr. Gérald Dherbomez de nous avoir donné ce projet et de nous avoir encadré durant ce projet. Nous souhaitons remercier Mr. Olivier Auverlot de nous avoir apporté son aide notamment sur les parties Web-Service en Pharo.

Introduction

Le projet Flexin se déroule au sein du laboratoire CRIStAL. Les différents intervenants sont Aymane Bouziane, Basile Dyment, étudiants en Master 1 informatique à Lille 1. Le responsable de projet est Gérald Dherbomez, Olivier Auverlot apporte son aide au projet. Le projet s'effectue sur une période de 4 mois environ, et une réunion avec les personnes s'occupant du projet s'effectue toutes les deux semaines.

Le projet Flexin consiste à développer une solution pour mieux ordonner le matériel du laboratoire CRIStAL. Pour cela, l'objectif est de créer une base de données qui référencera tout le matériel et de créer une application mobile Android pour le personnel de CRIStAL afin de pouvoir gérer tout ce matériel. Cette application mobile permettra d'interagir avec les différents équipements (au moyen de puce NFC ou de code barre) et mettra à jour les informations de position grâce à la puce GPS du téléphone de l'élément afin de plus facilement le retrouver. Elle inclura un système d'emprunt de matériel et un système d'état de matériel.

Les différentes technologies utilisées pour créer le projet sont Android Studio pour créer l'application, PostGreSQL pour la base de données et Pharo pour le web-service.

Une première partie de ce rapport sera dédiée au cahier des charges et à l'explication du déroulement du projet, puis dans une deuxième partie seront présentées les différents choix de conception du projet puis dans une troisième partie, l'application sera présentée et enfin dans une dernière partie les perspectives et des idées d'amélioration du projet seront présentées.

1. Description du projet

1. Cahier des charges

Au sein des différents laboratoires du campus de l'université de Lille 1, le personnel se trouve souvent confronté à des problèmes liés à la gestion de matériels comme les robots.

Quelques problèmes :

1. Chercher un matériel dans toutes les plateformes et bâtiments du campus pour le trouver.

2. Un matériel peut être emprunté et dégradé par l'emprunteur sans pénalisation car on ne connaît pas le responsable de la dégradation.

3. L'inventaire manuel du matériel peut s'avérer inefficace à cause d'oublis et d'autres erreurs humaines.

4. Un personnel peut perdre du temps en cherchant un matériel dont il a besoin qui est jeté à la poubelle ou qui est oublié quelque part lors d'un déménagement par exemple.

Solution :

Pour résoudre ce type de problèmes, l'application mobile Flexin, permet de gérer l'organisation des matériels. Toute personne voulant emprunter ou rendre un matériel devra utiliser cette application. L'utilisateur pourra en scannant un matériel, l'emprunter, le rendre, ou consulter simplement les informations associées, dont la localisation géographique où celui-ci doit être remis et la plateforme à laquelle il appartient. Cela permettra de sauvegarder dans une base de donnée bien organisée toutes les informations concernant les emprunts ou les retours dont: la date d'emprunt et de retour, l'état du

matériel lors de l'emprunt et lors du retour, sa localisation géographique courante et la personne qui l'a emprunté.

Cependant l'application mobile ne permettra d'effectuer que les manipulations principales tels que: consulter les informations, emprunter et retourner un matériel. Une plateforme web qui sera développée dans le futur permettra d'effectuer des opérations qui sont plus liées à la gestion tels que la consultation de la liste des matériels empruntés, et la consultation des informations d'un emprunteur pour le contacter par exemple.

Objectifs :

Les sections précédentes ont démontré le besoin de développer l'application Flexin. La majeure partie du projet s'est concentrée sur la conception et le développement des fonctionnalités clés de cette application.

Nous rappelons brièvement la solution à implémenter dans le cadre de ce projet. Il s'agit d'implémenter 3 fonctionnalités clés : consulter , emprunter , rendre. Une simple procédure de sécurité et d'authentification est à développer aussi. Celle-ci sera améliorée plus tard par d'autres développeurs.

L'application que nous développerons est une application mobile Android, nous utiliserons donc Android Studio comme environnement de travail et Java comme langage de développement.

En arrivant dans ce projet, nous avions également comme objectif personnel d'améliorer nos compétences dans le développement d'applications mobiles, et plus particulièrement dans le développement d'applications réparties qui reposent sur un système client/serveur/base de données.

Enfin le sujet de projet va nous permettre également de valider les acquis de notre formation à travers la conception et le développement de l'application Flexin, et de valider l'UE PJI du master 1 Informatique.

2. Déroulement du projet

Etapes de la réalisation du projet

1ère étape: Janvier/Mi-Février:

- Création du schéma de la base et réflexion sur les moyens pour la relier à la base de données APPSI (Voir Olivier Auverlot pour réunion le 27/01)

- Etude de la documentation sur les différentes technologies à utiliser sur Android: service de localisation, utilisation du NFC, librairie code barre/QR code.

2ème étape; Mi-Février/Fin Avril:

- Développement du web-service et de l'application Android.

3ème étape: Mai

- Tests et déploiement de l'application

Jalons

- 6 Mars: rapport de projet 50%

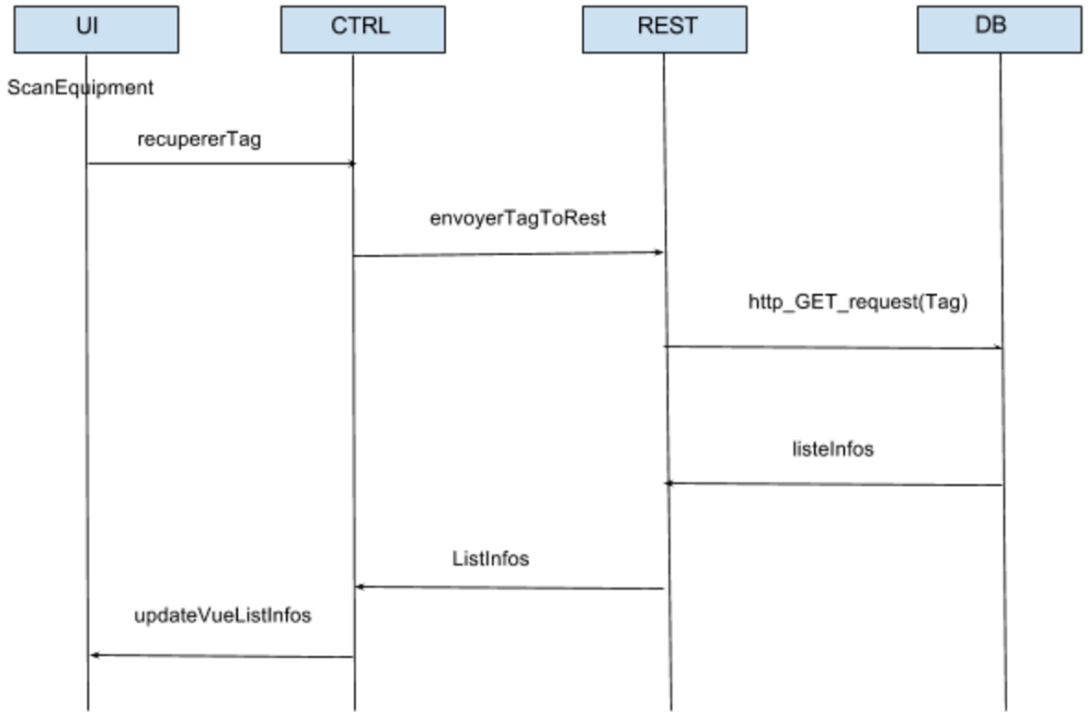
- 2 Mai: rapport à 90%

- Mi-Mai: rapport final

- Fin-Mai: soutenance

2. Conception

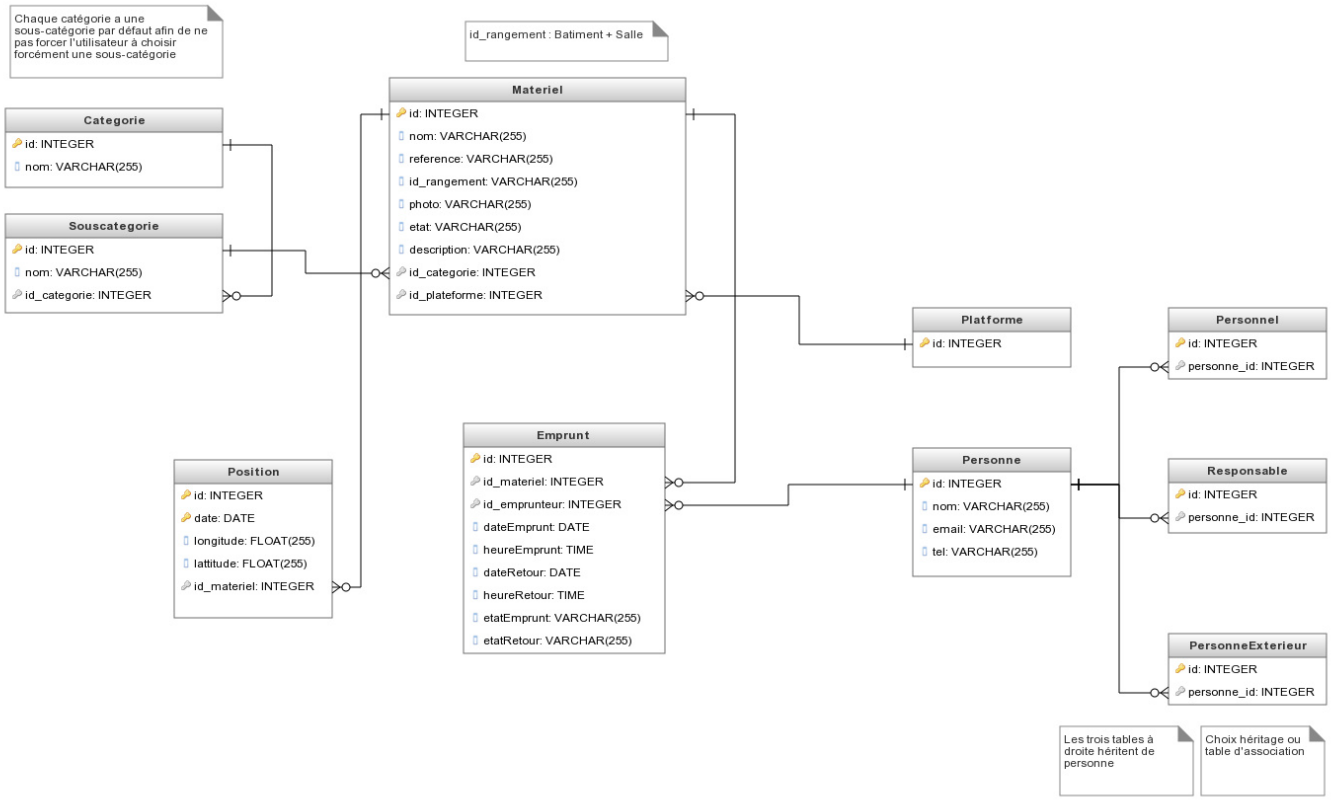
Le schéma suivant permet d'avoir une vue d'ensemble sur le fonctionnement du système :



Nous allons décrire le fonctionnement des différentes parties du projet.

1. Base de données

Contenu de la base de données:



Une table **Matériel** : un matériel contient plusieurs informations nécessaires comme la plateforme à laquelle il y est associé et qui est représentée par la table Plateforme, ainsi que la catégorie à laquelle il appartient et qui est représentée par la table Categorie.

Une table **Categorie** : catégorie d'un matériel

Une table **SousCategorie** : sous catégorie d'un matériel. Une sous catégorie appartient à une catégorie.

Une table **Plateforme** : une plateforme du laboratoire (le laboratoire a plusieurs plateformes).

Une table **Emprunt** : cette table rassemble toutes les informations associées à l'emprunt d'un matériel, comme son identifiant, la date d'emprunt, la date de retour, et la personne qui l'a emprunté représenté par la table Personne.

Une table **Personne** : une personne empruntant le matériel. Cette personne peut être un responsable du labo (table Responsable), un personnel du laboratoire (table Personnel) ou une personne extérieure (table PersonneExterieur).

Une table **Responsable** : un responsable du laboratoire. Cette table hérite de la table Personne.

Une table **Personnel** : un personnel du laboratoire. Cette table hérite de la table Personne.

Une table **PersonneExterieur** : une personne extérieure au laboratoire (un étudiant, une entreprise etc). Cette table hérite de la table Personne.

2. Serveur REST

Le serveur REST aura pour but de lier l'application à la base de donnée. Il permettra à l'application de consulter ou ajouter des données à la base de données facilement grâce à des requêtes HTTPs (GET, PUT, DELETE etc).

Le serveur écoute les demandes de l'utilisateur, et il s'occupe d'ajouter un nouveau matériel à la base de donnée avec les informations fournies par l'utilisateur, de récupérer des informations demandées par l'utilisateur de la base de donnée ou de supprimer des informations de la base de données.

3. Application Mobile

L'application mobile permet de scanner un tag NFC ou un code barre qui se trouve sur un matériel quelconque. Elle récupère l'information codée dans le tag ou dans le code barre, et la fourni au serveur REST. Ce dernier récupère de la base de données toutes informations désirées et correspondantes à l'information fournie par l'application, puis renvoie la réponse de la base de donnée à l'application pour qu'elle soit affichée.

De la même manière, l'application gère plusieurs fonctionnalités, tels que l'authentification d'un utilisateur, emprunter un matériel et rendre un matériel.

3. Présentation de l'application

1. Prérequis

Afin de faire fonctionner l'application, il faut avoir un périphérique Android qui dispose d'un appareil photo pour le scan de code barre et de la technologie NFC pour se connecter à son profil utilisateur et pour scanner les objets. Il faut aussi avoir une connexion internet quand on effectue un scan de matériel.

Pour avoir un profil utilisateur, il faut amener sa carte services Lille 1 à un administrateur pour qu'il enregistre dans la base de données le profil utilisateur et permettre d'utiliser au mieux toutes les fonctions de l'application Flexin.

2. Description de l'application Android

L'application est divisée en plusieurs activités. Nous allons décrire les différentes activités et leurs utilités:

Connexion

L'activité connexion permet à l'utilisateur au préalable enregistré par un administrateur de se connecter avec sa carte de service Lille 1. Pour ce faire il devra juste effectuer un contact NFC entre sa carte et son téléphone. Il pourra choisir aussi de ne pas se connecter en appuyant sur le bouton *"Ne pas se connecter"* mais il n'aura ensuite pas accès à toutes les actions dans ce mode utilisateur.

Accueil

L'activité accueil propose à l'utilisateur deux boutons: un bouton scan NFC et un bouton scan code barre. Elle permet donc à l'utilisateur de se diriger vers ces deux fonctions.

Scan NFC

L'utilisateur dans cette activité peut scanner la puce NFC sur un matériel. Il sera ensuite dirigé vers une activité matériel.

Scan code barre

L'utilisateur peut scanner le code barre d'un matériel. Il sera ensuite dirigé vers une activité matériel.

Matériel

L'utilisateur voit dans cette activité les informations du matériel qu'il a scanné. Ces informations sont:

- Référence
- Nom
- Rangement
- Plateforme
- Catégorie
- Description
- Etat

Si l'utilisateur s'est connecté au démarrage de l'application il pourra emprunter l'object. En fonction de l'état d'emprunt du matériel, différentes actions sont disponibles:

- Si le matériel est disponible, un bouton *"Emprunter"* permettra de l'emprunter. L'utilisateur connecté sera donc l'emprunteur du matériel.

- Si le matériel est emprunté par un autre utilisateur, un texte s'affichera: *"Emprunté par Basile Dyment"* par exemple.

- Si le matériel est emprunté par l'utilisateur, un bouton *"Rendre"* sera disponible. En appuyant sur ce bouton une activité *"Rendu"* s'ouvrira.

Rendu

L'activité rendu permet à l'utilisateur de rendre un matériel. On lui demande de sélectionner l'état dans lequel sera rendu le matériel et il pourra ensuite en appuyant sur le bouton *"Rendre"* changer le statut du matériel en disponible.

3. Transferts entre application mobile et REST

Côté client:

Dans cette partie nous allons expliquer comment se déroulent les choses au niveau du code, lorsqu'un utilisateur scanne un matériel et consulte simplement ses informations. Ce que nous allons expliquer pour ce scénario d'utilisation s'applique exactement de la même manière sur tous les autres scénarios tels que emprunter et rendre un matériel.

Quand l'utilisateur de l'application scanne un matériel, le scan se fait sur une activité indépendante, le tag est parsé et décodé, puis l'information est

récupérée. Cette information est un identifiant du matériel. Cet identifiant est envoyé à MaterielActivity.

Dans MaterielActivity, l'identifiant du matériel est récupéré et passé en paramètre d'une méthode displayMateriel() qui fait elle appel à la classe MaterielManager, qui s'occupe de la communication avec le serveur rest.

```
public class MaterielActivity extends HomeActivity {
    public static final String TAG = "tag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //ici on récupere le tag nfc scanne par la classe de scan viewTag
        Intent intent = getIntent();
        String message = intent.getStringExtra(TAG);
        displayMateriel(message);
    }

    public void displayMateriel(final String id){
        final MaterielManager materielManager = new MaterielManager(id);
    }
```

MaterielManager envoie au serveur REST, grâce à la méthode getMateriel(), l'identifiant du matériel scanné et récupère la réponse dans un CallBack. Les informations du matériel envoyés par le serveur sont enregistrées dans un objet Java Materiel.

```
public final void getMateriel(final GetMaterielCallBack getMaterielCallBack){
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder().url(URL +
"materiel/"+this.id ).build();
        client.newCall(request).enqueue(new Callback() {

                @Override
                public void onFailure(Call call, IOException e) {
                        System.err.println("Getting materiel failed Callback failure in
MaterielBis class");
                }

                @Override
                public void onResponse(Call call, Response response) throws
IOException{
                        String result = response.body().string();
                        Gson gson = new Gson();
                        Type collectionType = new
TypeToken<Collection<Materiel>>(){}.getType();
                        Collection<Materiel> enums = gson.fromJson(result,
collectionType);
                        Materiel[] materiels = enums.toArray(new
Materiel[enums.size()]);
                        getMaterielCallBack.onSuccess(materiels);
                }
        });
}
```

```
public interface GetMaterielCallBack {
        public void onSuccess(Materiel[] materiel);
        public void onFail();
}
```

Quand la réponse du serveur est récupérée grâce à la méthode getMateriel() du materiel_manager, l'activité **MaterielActivity** récupère le callBack du MaterielManager et récupère donc la réponse, puis l'affiche sur l'écran.

Côté serveur:

Le serveur REST a été codé en Pharo afin d'assurer son intégration sur les serveurs CRIStAL. Le web service utilise deux frameworks Pharo existant: Teapot qui permet de créer un web service et de programmer les redirections et Garage qui permet de récupérer les informations dans la base de données en lui passant des commandes PostGreSQL.

Nous avons crée deux fonctions en Pharo afin de pouvoir faciliter au maximum l'utilisation et l'ajout de requêtes SQL à ce web-service:

- sqlQuery: *sqlCommand*
  Cette fonction prend en entrée une chaine de caractère qui est la commande SQL qu'on veut envoyer. Elle permet au web-service de se connecter à la base de données, d'effectuer la requête SQL et de renvoyer une liste contenant les différents éléments

- restRoute
  Cette fonction effectue les routages de chacun des liens afin de pouvoir accéder aux données des différentes tables SQL en JSON.

Exemple de récupération d'informations de la base de données :

```
restRoute
        "route the method"
        lemprunts teapotl

        emprunts := Dictionary new.
        teapot := Teapot configure: {
                #defaultOutput -> #json. #port -> 8080. #debugMode -> true }.
        teapot
                GET: '/materiels' -> ((WebService new) sqlQuery: 'SELECT m.nom,
reference, photo, etat, description, p.nom AS plateforme FROM "Materiel" m LEFT JOIN
"Plateforme" p ON m.id_plateforme = p.id'). start;
```

Nous avons rajouté à notre serveur REST plusieurs endpoints utilisant uniquement la méthode http GET, voici quelque exemples :

```
"Récupérer les informations d'un matériel : "

        GET: '/materiel/<id>'

"Mettre à jour la base de donnée quand un emprunt est effectué :"

        GET: '/emprunt/<id_materiel>/<etat_emprunt>/<id_emprunteur>'

"Mettre à jour la base de donnée quand un retour est effectué :"

        GET: '/rendre/<id>/<etat>'
```

4. Améliorations et perspectives du projet

Idées de fonctions à ajouter

Durant ce projet, nous avons implémenté les fonctions principales mais nous avons aussi pensé à d'autres fonctions que nous n'avons pas eu le temps d'implémenter.

Amélioration du système de connexion

Afin d'assurer une meilleure sécurité, il faudrait ajouter une demande d'authentification par code utilisateur à chaque fois qu'on veut effectuer une action importante (emprunt, rendu). Ce code utilisateur serait donné au moment de la création du profil utilisateur auprès de l'administrateur.

Ajout de la gestion de position

Pour mieux retrouver l'emplacement des objets, il faudrait connaitre leur position approximative. Pour la connaitre, a chaque emprunt et scan d'un matériel, la position actuelle du matériel devrait être mis à jour grâce au GPS du périphérique Android. L'emplacement sur une carte pourrait être ajouté à la description du matériel.

Ajout de la gestion des matériels pour lesquels on est responsable.

Pour une meilleure gestion des matériels, chaque utilisateur aurait accès à sa liste de matériel dont il est responsable et au statut d'emprunt des matériels ainsi qu'à l'historique d'emprunt comportant les informations de chaque emprunt effectué sur un matériel. Un responsable pourra demander la restitution d'un matériel via l'application ce qui enverra une notification à l'emprunteur.

Ajout de la liste des matériels empruntés.

Pour se souvenir des différents matériels qu'on a emprunté et à qui les rendre, il faudrait ajouter une liste avec les différents matériels empruntés.

Amélioration du code

Transformer certaines activités en fragments

Au niveau du code de l'application, nous avons choisi de faire des activités mais ce serait surement mieux pour les performances de l'application d'utiliser des fragments.

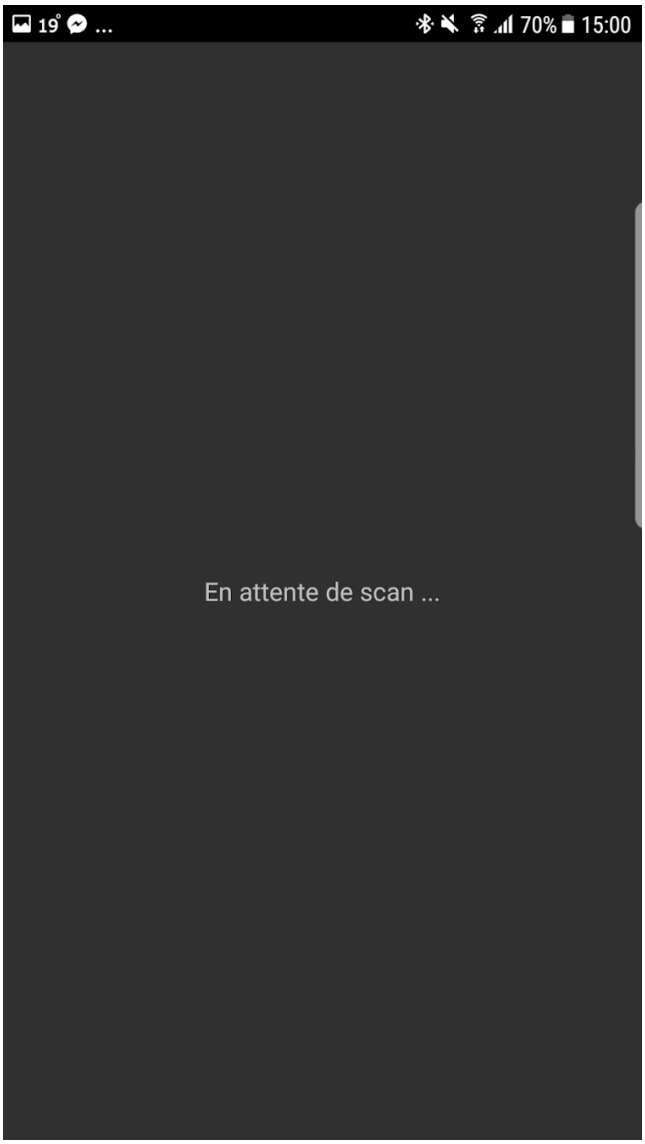
Conclusion

Nos objectifs de départ ont été atteint, d'autres fonctions peuvent être ajoutées par ailleurs ou même améliorées afin de faciliter la gestion des matériels.

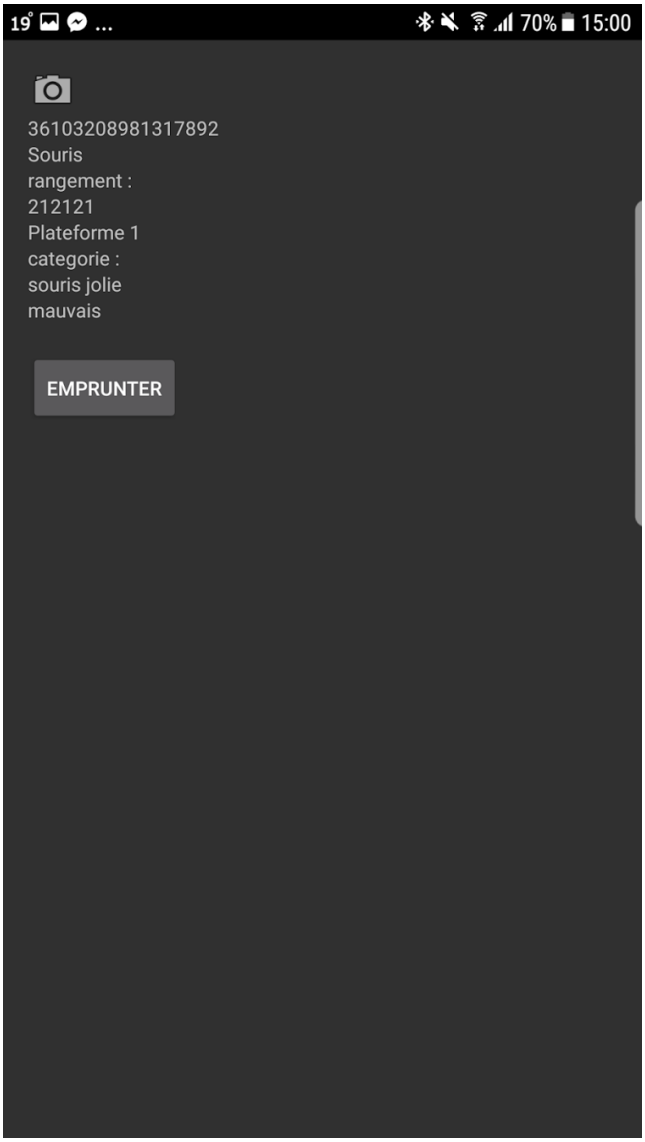
Ce projet nous a permis d'apprendre à connaitre l'univers Pharo et de programmer un Web Service dans ce language, d'améliorer nos compétences en Android et d'apprendre à utiliser le NFC et nous avons pu avoir des rappels en PostGreSQL.

Nous avons pu travailler en groupe sur le même projet et revoir les différentes étapes de développement d'une application et confirmer notre choix d'orientation vers un master IAGL.

Annexes



Activité de scan (L'application attend un contact NFC)



Activité affichant les informations d'un matériel