Rapport de projet de fin d'étude

Plan DAO en réalité augmentée

Elèves Ingénieurs :

Adel Aljane

Célia Amegavie

Encadrants :

M. Jean-Christophe Larrea

M. Laurent Grisoni

IMA 5

Année 2013-2014

Sommaire

Introduction

Dans le cadre de notre projet de fin d'étude, nous avons choisi de réaliser le projet « Plan DAO en réalité augmentée » en collaboration avec Oxylane. Ce projet en collaboration avec une entreprise est un bon moyen de nous préparer au métier d'ingénieur que nous devrons exercer dans quelques mois.

1. Le projet

1.1. Contexte

Le réseau Oxylane est un ensemble d'entreprises de la grande distribution spécialisées dans les articles de sport et de loisir. Il se définit comme créateur de produits sportifs et commerçant local et en ligne.
Afin de pouvoir répondre aux besoins de leurs clients en fonction des saisons, les magasins implantent leurs rayons quatre fois par an (selon les saisons de l'année). Les rayons en magasin sont faits selon un plan appelé « plan DAO ».
Traditionnellement, l'agencement et l'implantation des rayons se fait selon une procédure où les services centraux préconisent une certaine organisation des rayons qui sera effectuer par les magasins et en particulier les vendeurs en se basant sur des fichiers pdf à imprimer.
Ce projet essaie des trouver des solutions innovantes dans une optique d'innovation et numérisation du processus d'organisation.

1.2. Présentation

Le but initial de ce projet est de réaliser un prototype, permettant aux vendeurs des magasins de vérifier l'implantation des rayons du magasin à travers une application mobile et en utilisant le concept de la réalité augmentée.
Les discussions et les réunions de projet avec le responsable de projet chez Oxylane M. Jean-Christophe Larrea ainsi qu'avec notre encadrant de projet M. Laurent Grisoni ont permis d'élargir les réflexions pour proposer une solution innovante.

2. Cahier des charges

2.1. Cahier des charges initial

Nous avons exprimé une spécification des besoins initiaux conformément au sujet et à la préconisation du client (Oxylane).
Spécification : Développer un prototype d'une application Android permettant d'aider les vendeurs à organiser les rayons.
L'application comporte :

- Une page d'accueil
- Un mécanisme d'authentification
- Sélection d'un rayon ainsi qu'une partie de rayon à organiser.
- Une procédure d'aide à l'organisation avec double affichage:
  - Une liste de matériel nécessaire
  - Un affichage en réalité augmentée
- Validation d'une partie de rayon

Ce cahier de charge est présenté concrètement à travers la création d'une maquette de projet.

2.2. Redéfinition du cahier des charges

Le nouveau cahier de charge redéfinit le projet à travers 3 sous-parties:

- Un système d'identification des produits par RFID
  - C'est une raquette produite par Oxylane désignée initialement à l'inventaire des produits par communication RFID avec les tags RFID passifs implantés sur tous les produits et en adaptant une application android existante.
- Un système de localisation produit par rapport au rayon
  - Ce système est basé sur une Kinect fixé à côté du rayon permettant une analyse de l'espace rayon et en effet, détecter d'une manière globale la position de la raquette d'identification sur le rayon.
- Une comparaison entre le rayon théorique et le rayon implanté :
  - Produit mal positionné
  - Produit à la bonne position

Un système de communication entre toutes les parties sera mis en place.

3. Réalisations

3.1. Réflexion

Nous avons effectué une réflexion globale sur le sujet proposé par Oxylane afin de définir les objectifs et les contraintes du projet ainsi qu'un questionnement sur plusieurs aspects du projet tel que l'utilisation de la réalité augmentée ou les smartphones sous Android.

3.2. Premier prototype

3.2.1. Présentation

En basant sur le travail de réflexion initial, les préconisations du sujet et les réunions avec le responsable de projet chez Oxylane, nous avons décidé de commencer à mettre en place un prototype d'une application Android basé sur la réalité augmenté pour aider les vendeurs à organiser les rayons.

3.2.2. Outils

L'écosystème d'Android s'appuie sur deux piliers:

• Le langage Java

• Le SDK qui permet d'avoir un environnement de développement facilitant la tâche du développeur.

Pour commencer, nous avons commencé par installer l'environnement développement Android:

- Eclipse l'IDE (Environnement de développement intégré)
- Installation du JDK de java (Kit de développement java)
- Le SDK d'Android (Kit de développement)
- Plug-in ADT (Outils de développement Android)

3.2.3. Conception

Afin de réaliser une maquette démonstrative de l'application, nous avons utilisé Pencil Project comme un outil de design des différentes vues, avant le développement réel de l'application.
Nous avons ensuite présenté cette maquette à M. Larrea qui l'a validé. Nous avons donc commencé le développement.

Après l'installation des outils de développement Android et la réalisation d'une maquette de l'application, on arrive à l'étape de développement de l'application.
L'architecture de l'application se compose des plusieurs parties :

- Écran d'accueil :

Cette partie est développée à travers une simple classe MainActivity à partir de laquelle l'utilisateur peut accéder à la phase d'authentification pour accéder par la suite au contenu de l'application comme le montre la figure 1.

Figure 1 : Page d'accueil de l'application

- Page d'authentification

Comme on réalise un prototype, le mécanisme d'authentification est réalisé avec gestion d'erreur pour rendre le prototype réaliste. La page d'authentification est présentée à travers la figure 2.

Figure 2 : Page d'authentification de l'application

- Page plan du magasin

Par principe de simplicité, le plan du magasin est représenté sous forme d'une liste des rayons cliquables avec des données stockées dans un fichier xml. La figure 3, montre une liste des rayons.

Figure 3 : Page plan du magasin

- Page Rayon

On représente la page rayon choisi à l'étape précédente par des morceaux des mètres linéaires puisque la procédure d'implantation se fait partie par partie.
La galerie des images (visible sur la figure 4) est réalisée à travers une classe interne qui hérite d'une classe BaseAdapter depuis la librairie android.widget.

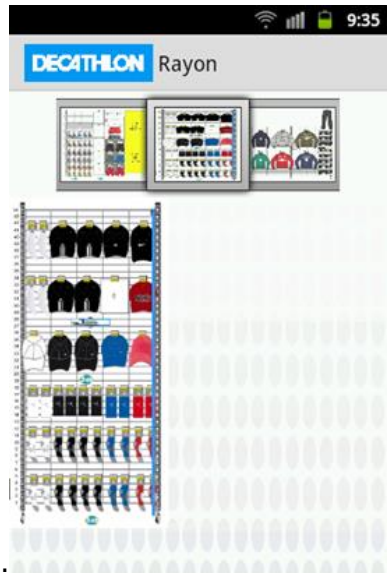

Figure 4 : Page rayon choisi de l'application

- Partie Procédure

Dans cette partie, On a essayé de prendre en compte l'ergonomie de l'application, puisque on est dans une procédure d'organisation des rayons où l'utilisateur doit avoir un mode navigation le plus simple possible. En effet, on a choisi le défilement horizontal entre 5 vues :
- Agencement Matériel
- Agencement RA
- Implantation Matériel
- Implantation RA
- Validation

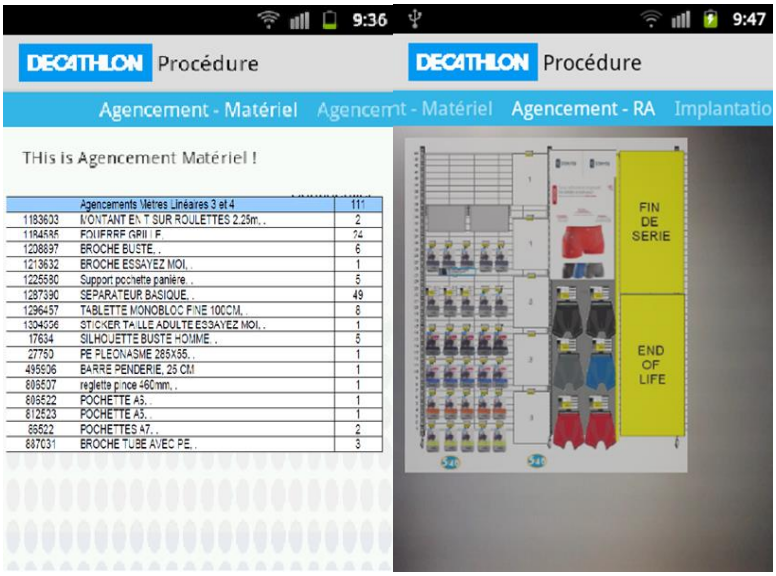

Figure 5 : Exemples de mode de navigation de l'application

La représentation du matériel nécessaire à l'agencement et à l'implantation se fait à l'aide d'une liste des outils et des produits nécessaires.

On a implémenté aussi la navigation à l'aide de 3 classes :

- Classe principale Procedure
- Classe PagerAdapter : affiche le layout en fonction de la position de navigation
- Classe Fragment : Elle permet de changer le contenu du fragment affiché.

Comme on utilise la camera android :

- Classe Previex camera: Elle permet de prendre en charge la camera android et de l'afficher comme un SurfaceView.

3.2.4. Critique du prototype

Le prototype d'aide à l'organisation des rayons DAO par des vendeurs perturbe la réalité métier d'un vendeur qui est censé effectuer plusieurs tâches manuelles dans son processus d'organisation. En effet, les vendeurs ne sont pas prêt à abandonner le processus actuel basé sur les papiers imprimés à partir de pdf.

M. Larrea nous a signalé qu'après un échange interne avec le responsable du processus implantation, que le projet est un outil de vérification et de validation et non pas un remplacement de processus d'implantation actuel.

L'aspect vérification peut être une solution qui s'intègre dans le processus actuel sans perturber le fonctionnement du processus implantation basé sur les pdfs surtout avec la disparité dans les « Template » des différentes marques.

3.3. Prototype final

Suite à une réunion avec nos deux responsables de projet, nous avons décidé de changer le cahier des charges car le prototype ne correspondait plus aux réalités métier des vendeurs.
Le prototype final devra permettre de vérifier si l'implantation du rayon correspond bien au plan d'implantation. Il faut donc arriver à détecter la position de chaque article et vérifier si elle correspond bien au plan d'implantation.

3.3.1. Architecture

Le système se compose de 3 parties comme le montre la figure 6:

Un système d'identification des produits par RFID :
C'est une raquette produite par Oxylane désignée initialement à l'inventaire des produits par communication RFID avec les tags RFID passifs implantés sur tous les produits.

Un système de localisation produit par rapport au rayon :
Ce système est basé sur une Kinect fixé en haut du rayon permettant une analyse de la géométrie du rayon et en effet, détecter d'une manière globale la position de la raquette d'identification sur le rayon.

Une comparaison entre le rayon théorique et le rayon implanté :

- Produit mal positionné
- Produit à la bonne position

Un système de communication entre les différentes parties est représenté par la figure suivante :

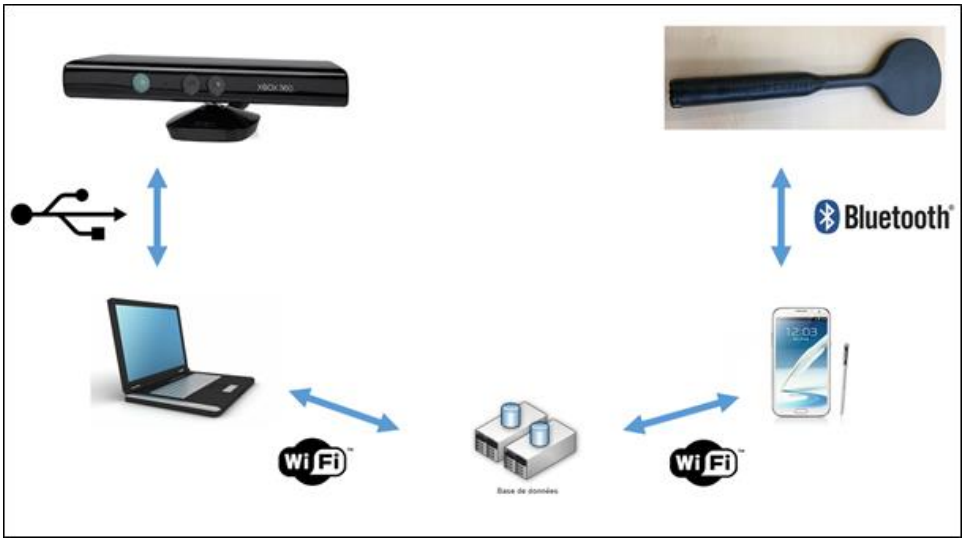

Figure 6 : L'architecture du système

3.3.2. Système de localisation

Ce système est basé sur une Kinect fixée à côté du rayon permettant une analyse de la géométrie du rayon et en effet, détecter d'une manière globale la position de la raquette d'identification sur le rayon.

3.3.2.1. Présentation de la kinect

Kinect initialement connu sous le nom de code Project Natal est un périphérique destiné à la console de jeux vidéo Xbox 360 permettant de contrôler des jeux vidéo sans utiliser de manette.

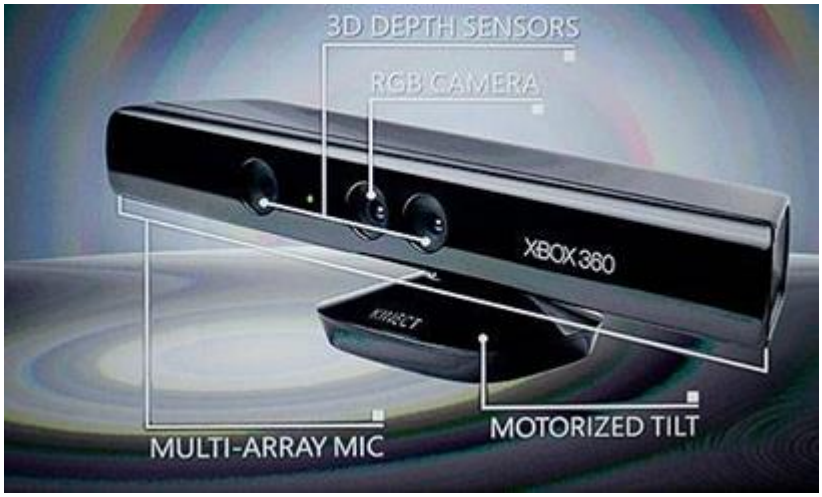

Figure 7 : Représentation d'une kinect

Le capteur Kinect envoie vers le système un ensemble de trois flux :

- Le flux image peut être affiché comme pour n'importe quelle caméra. Le capteur Kinect peut retourner le flux en 640x480 (à 30 images par seconde) et en 1280x1024 (à 15 images par seconde).
- Le flux de profondeur va donner à chaque pixel une profondeur depuis le capteur. Ainsi en plus de la position en 2D de chaque pixel (et de leur couleur) nous disposons désormais de leur profondeur. Cela va énormément faciliter les recherches de formes.
- Un troisième flux est également envoyé depuis le capteur : il s'agit du flux audio en provenance des quatre microphones du Kinect.

Le point clef ici concerne donc la capacité du capteur Kinect à nous donner des informations tridimensionnelles. En utilisant ces dernières, la librairie NUI (qui est fournie avec le SDK) est capable de détecter la présence d'humains en face du capteur. Elle peut ainsi "voir" jusqu'à 4 personnes et en suivre précisément deux.

3.3.2.2. Etude de l'architecture de Microsoft Kinect Sdk

Le capteur Kinect envoie vers le système un ensemble de trois flux.La communication entre le capteur et l'application se fait via une librairie NUI (*Natural User Interfaces)*

Figure 8 : Communication entre la kinect et les applixations

3.3.2.3. Recherches effectués

Lors du travail de réflexion et de recherche de solution nous avons évoqué deux approches de tracking de la raquette :

- Première approche : suivi d'objet

Cette première approche est basée sur le principe de suivre le déplacement d'un objet à l'aide d'une caméra en utilisant le flux rgb et la librairie opencv. Ensuite on pourra estimer la distance à travers le flux de profondeur selon l'axe z.
L'inconvénient principal de cette approche découvert lors de l'expérimentation est la non fiabilité de détection d'un objet par traitement d'image par principe de Slicing et Contour Finding.

- Deuxième approche : suivi des squelettes

Cette deuxième approche est basée sur la grosse force du capteur Kinect pour Windows SDK qui est sa capacité à retrouver le squelette de joints des humains présents devant le capteur. C'est un système de reconnaissance extrêmement rapide et ne nécessitant aucun apprentissage à l'utilisation.
Contrairement à la première approche, celle-ci a l'avantage d'être extrêmement fiable en terme de reconnaissance.

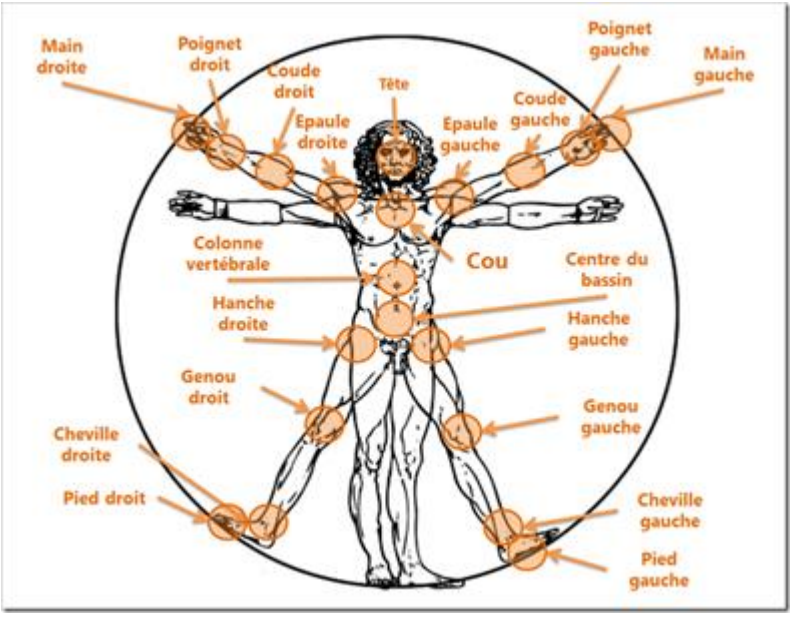

Figure 9 : Schéma de reconnaissance du capteur kinect

3.3.2.4. Réalisation de l'application de détection de la raquette

A l'aide de la librairie nui (*Natural User Interfaces*) fourni par Microsoft Kinect Sdk, nous avons réalisé un module permettant la détection de la position de la raquette à travers le suivi du squelette de l'utilisateur et récupération de la position de la main droite.

Cette application a été développée comme une application WPF à l'aide du langage C# très proche du Java.

Nous avons développé notre application avec les outils suivants :

- Visual studio
- Microsoft kinect sdk
- MySql Connect/NET

```
public MainWindow()...

public void setposition()...

private void WindowLoaded(object sender, RoutedEventArgs e)...

private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)...

private void SensorSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)...

private void DrawBonesAndJoints(Skeleton skeleton, DrawingContext drawingContext)...

private Point SkeletonPointToScreen(SkeletonPoint skelpoint)...

private void DrawBone(Skeleton skeleton, DrawingContext drawingContext, JointType jointType0, Joint

private void getZone(double z, double y)...
```

Figure 10 :Classe prinicipale

```
class Bd
{
    string ConnexionString = "server=db4free.net;userid=adel;password=flash30n;database=bdpfe";

    MySqlConnection connexion;
    MySqlCommand cmd;

    public Bd()...


    public bool getConnexion()...


    public bool update(double z, double y)...


    public bool CloseConnexion()...


}
```

Figure 11 : Classe de gestion de la base de données

- Acquisition de données Kinect

Comme on l'a vu dans la partie présentation, le Sdk permet l'acquisition de 3 types de trames (flux image RGB, flux profondeur, flux audio) mais aussi de traquer le squelette de l'utilisateur.

- Suivi de squelette

Pour tracker un squelette, on a développé les étapes suivantes :

- Initialisation du capteur :

Durant cette partie qui s'exécute lors du chargement de l'application, on initialise la librairie nui pour détecter le capteur et l'identifier

- Synchronisation sur un évènement :

Pour permettre de récupérer un type de trame donné, il faut l'indiquer à la librairie en se branchant sur l'évènement adéquat, ici on a choisi de se synchroniser sur l'évènement *SensorSkeletonFrameReady* permettant d'indiquer que la trame est disponible.

- Récupération des données d'un squelette

On implémente les traitements associés à l'évènement *SensorSkeletonFrameReady.* Quand l'évènement SkeletonFrameReadyEventArgs e se manifeste, on récupère et on traite les données du squelette tracké :

  - On ouvre le trame à l'aide de e.OpenSkeletonFrame()
  - On recopie les données par CopySkeletonDataTo(skeletons)

- Localisation de la main de l'utilisateur

Dans cette partie, on isole le joint de la main droite identifié par une énumération JointType.HandRight.
On visualise l'identification de la main droite par un changement de coloration du joint en rouge.
On récupère la position du joint stocké dans l'attribut position.
La position est représentée par 4 valeurs (x,y,z,w) : les cordonnées spatiales sont calculés selon l'orientation indiquée sur la figure suivante et la valeur de w compris entre 0 et 1 permettant d'indiquer la qualité et la fiabilité de la position.

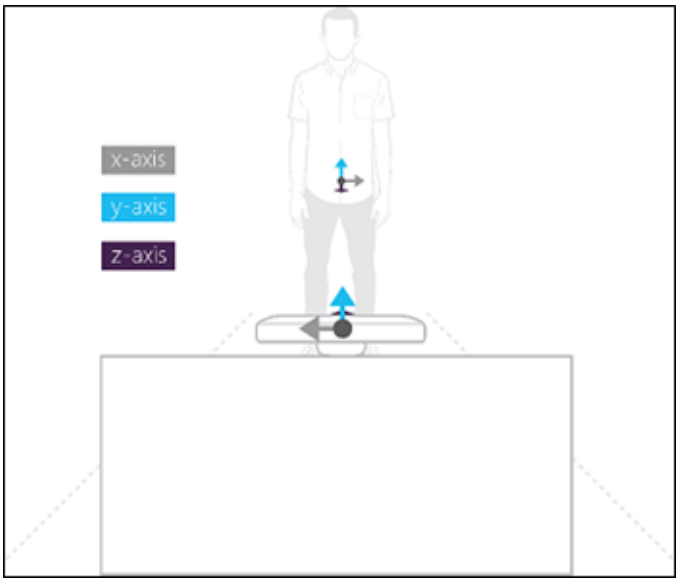

Figure 12 : Position de l'utilisateur par rapport à la kinect

➢ Visualisation des résultats

Un affichage sur PC permet de vérifier le fonctionnement de l'application kinect et de visualiser les résultats et les erreurs

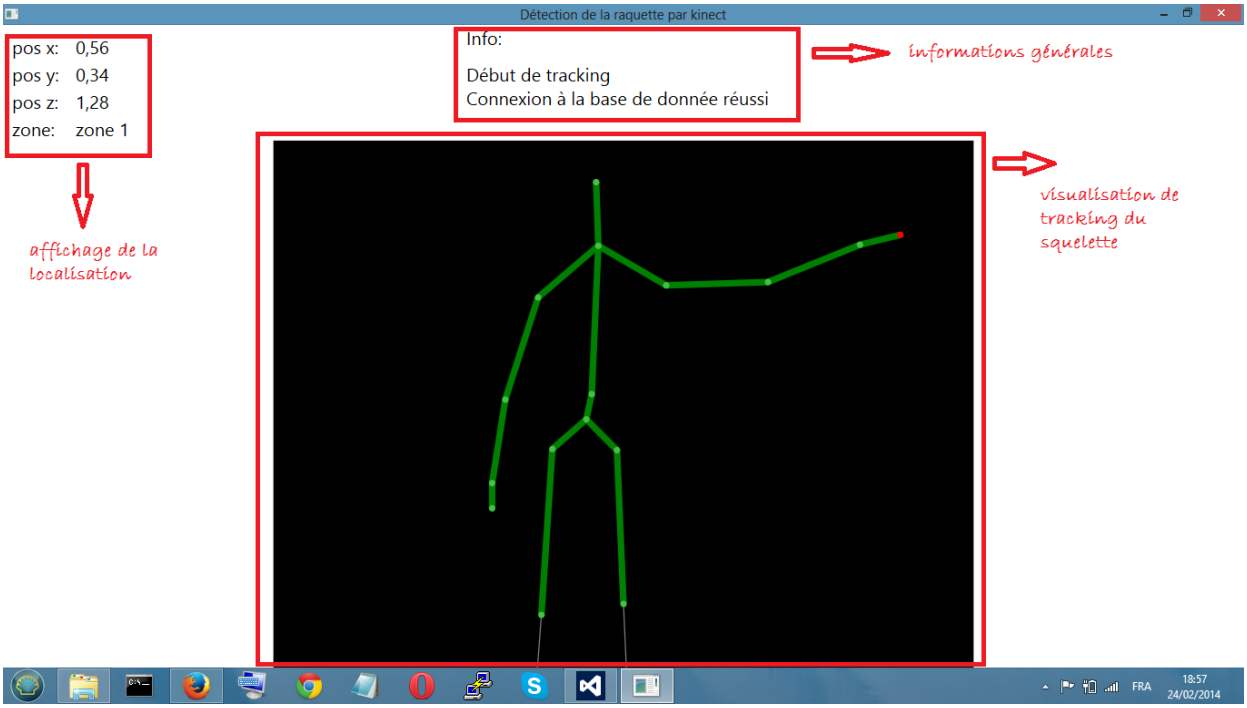

Figure 13 : Vérification de l'application kinect

➢ Mise à jour de la base de données

Pour alléger l'exécution de l'application et éviter le plantage de l'application, nous avons créé un processus léger thread permettant d'interagir avec la base de données pour mettre à jour les valeurs de z et y

Ce processus léger s'exécute à un intervalle temps fixe (500 ms) met à jour la base de données par :

- Connexion au serveur distant de la base de données
- Exécute une requête SQL update pour charger les valeurs de variables z et y dans la table position
- La déconnexion de la base se fait lors de la fermeture de l'application

➢ Gestion d'erreur

Notre application permet de capter et de contrôler les erreurs qui peuvent arriver lors de l'exécution. En effet, les erreurs suivantes sont contrôlées à travers les exceptions logicielles générées:

- Le non branchement de la kinect
- Problème de connexion à la base de données
- Problème dans la mise à jour de la base de données

3.3.2.5. Tests et ajustement

Après une vérification sur la sensibilité du capteur en fonction de la distance par rapport à l'utilisateur, à partir de la documentation et comme l'explique la figure suivante; il parait nécessaire de placer le capteur à une distance d'un mètre de l'extrémité du rayon pour garantir la fiabilité de la reconnaissance de squelette.

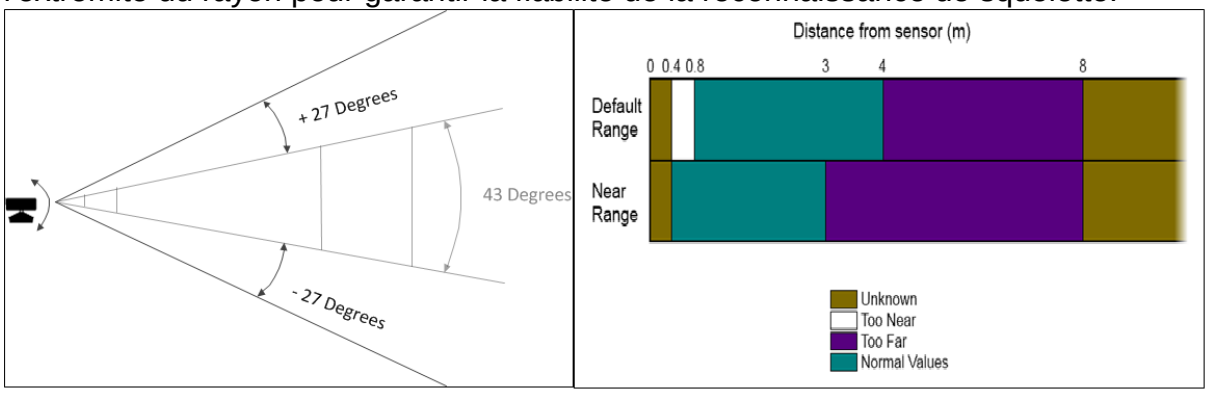

Figure 14 : Sensibilité du capteur en mode reconnaissance

Pour tester l'application, nous avons défini 9 zones de localisation des produits sur un rayon en fonction des coordonnes (z et y)
La figure suivante nous montre les différentes zones testées avec une bonne fiabilité.

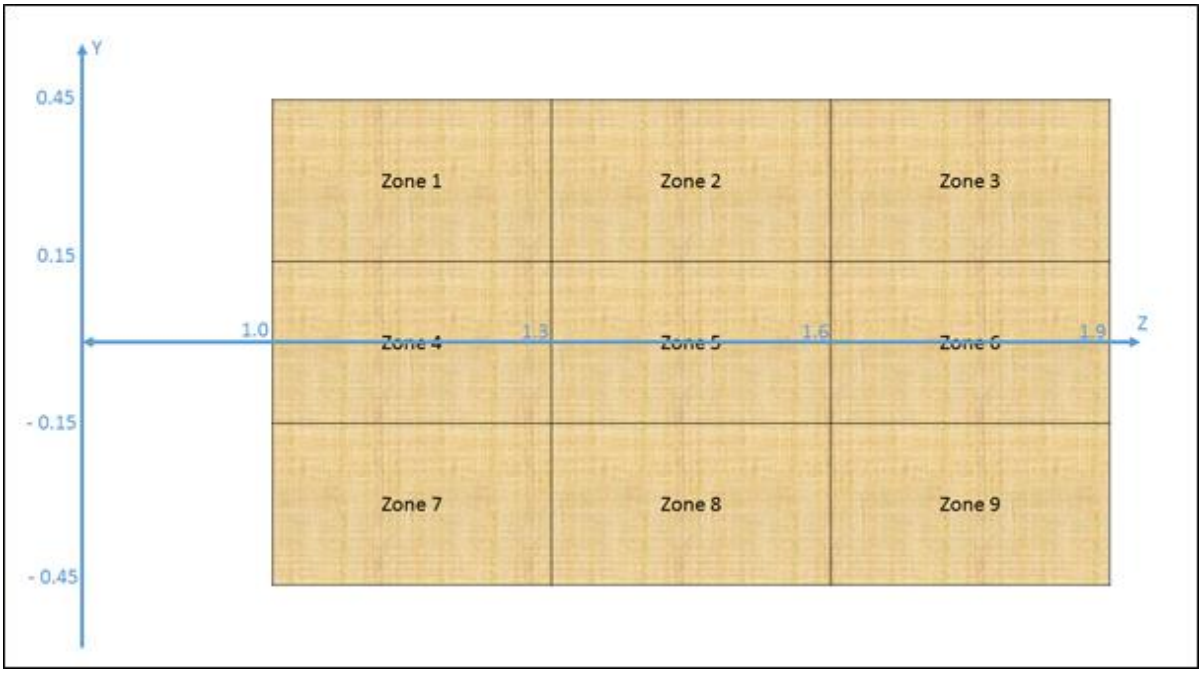

Figure 15 : Zones de test

3.3.3. Système d'identification

Dans cette partie nous expliquerons comment nous récupérons les informations de chaque article. Nous avons d'abord besoin de deux choses : Une raquette RFID et un téléphone android. La raquette RFID est un outil que possédait déjà Oxylane. Elle est composée de deux parties : la tête et le corps

Figure 16 : Raquette RFID

Cette raquette est utilisée initialement pour faire les inventaires des produits. Nous avons décidé de l'utiliser car elle s'intègre parfaitement dans les réalités métiers des vendeurs. Ils n'auront donc pas besoin d'une formation supplémentaire pour l'utiliser. Cette raquette s'utilise de façon assez simple. Tout d'abord il faut connecter par Bluetooth la raquette à un téléphone android. En effet Oxylane possède une application android qui permet de se connecter à la raquette et faire l'inventaire de tous les produits scannés. Ensuite en la tenant par le corps, il faut rapprocher la tête d'un tag RFID. La tête émet un rayonnement qui permet de scanner tout les tags RFID qui sont présent dans cette zone. Une fois ces tags sont scannés, les informations relatives au produit sont envoyées sur les serveurs d'Oxylane par le biais de l'application pour être traité.

Comme nous n'avons pas accès aux serveurs d'Oxylane, nous avons du trouver un moyen pour accéder aux informations des produits avant qu'ils ne soient envoyé sur

les serveurs d'Oxylane. Pour cela nous avons récupéré le projet «DécathlonMobility » permettant de faire l'inventaire et nous l'avons modifié.

C'est un projet assez conséquent comme on peut le voir figure 16. Ici le package qui nous intéresse est « com.oxylane.android.bluetooth.inventory »

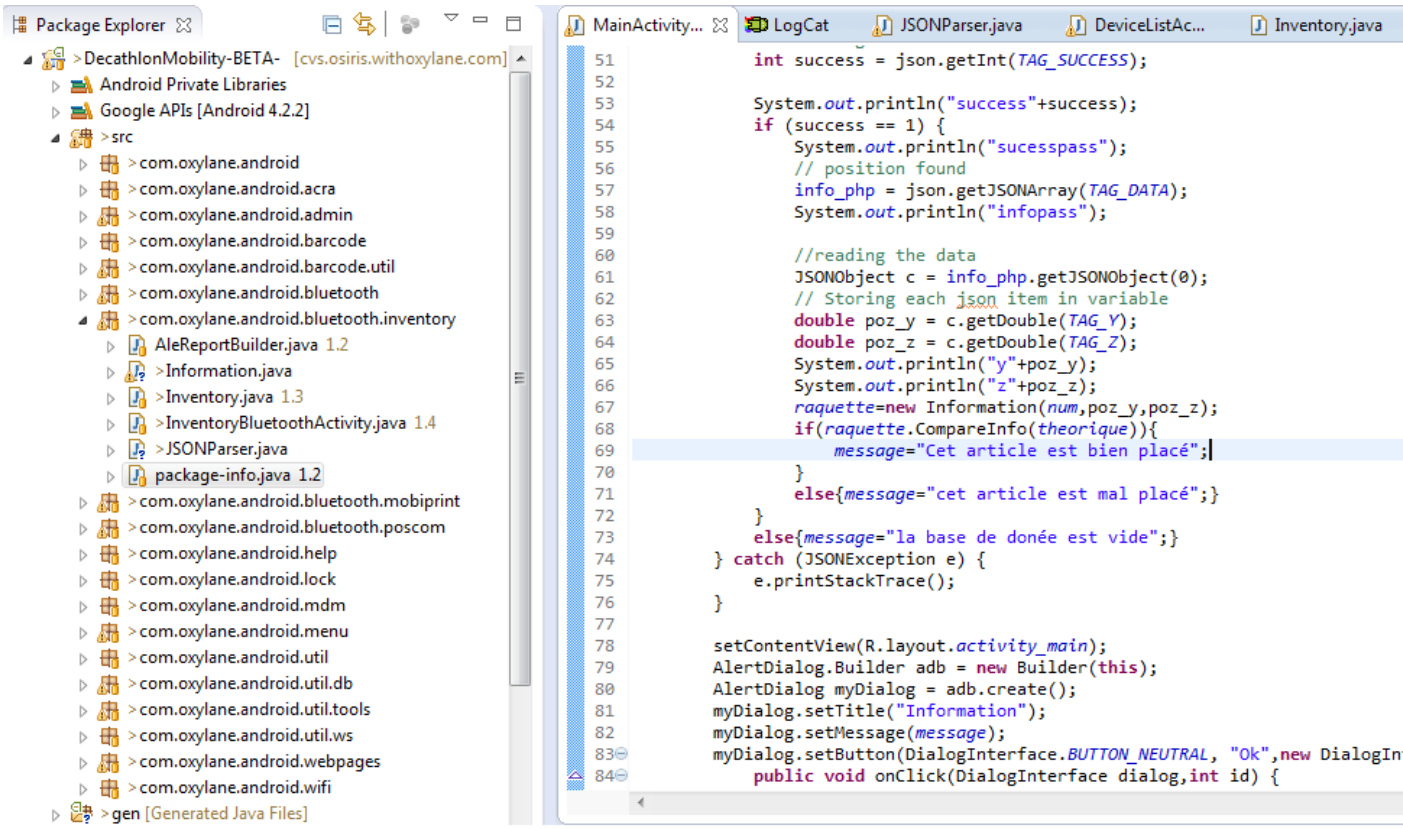

Figure 17 : Projet DécathlonMobility

Ce package était initialement composé de quatre classes : AleReportBuilder, Inventory, InventoryBluetoothActivity, et package-info.
Après avoir récupéré le projet, nous avons d'abord essayé de comprendre son architecture afin de savoir comment nous pourrions y intégrer notre solution. Nous avons alors remarqué que les informations des produits passaient par la fonction « manage » dans la classe « inventoryBluetoothActivity » avant d'être envoyé sur les serveurs. Nous allons donc les récupérer à cet endroit avant qu'ils ne soient envoyés. Avant de commencer à mmodifier cette fonction, nous avons créés la classe « Information » qui nous permet de stocker toute les informations relatives aux produit tel que son « epc (code modèle du produit + code article) » et ses positions. Nous avons aussi ajouté une méthode « CompareInfo » qui permet de comparer deux classes, ce qui facilitera la comparaison entre le rayon théorique et le rayon réel.

Cette classe a été définie comme suit :

```
package com.oxylane.android.bluetooth.inventory;

public class Information {
    private String epc;
    private double posy;
    private double posz;

    /**
     * constructor
     * @param message
     * @param positiony
     * @param positionz
     */

    public Information(final String message, final double positiony, final double positionz) {
        this.epc=message;
        this.posy=positiony;
        this.posz=positionz;
    }

    public Boolean CompareInfo(Information info){
        if (this.epc.equals(info.epc)){
            if(this.posy==info.posy){
                if(this.posz==info.posz){
                    return true;
                } else return false;
            }else return false;
        }else return false;
    }


}
```

Figure 18 : Classe information

Nous avons ensuite modifié la fonction « Manage » afin de récupérer les informations produit. Sachant que la fonction n'est appelée que si un tag est scanné.
La fonction a été modifiée de la manière suivante :

```
 private static String num;
 // Creating JSON Parser object
JSONParser jParser = new JSONParser();
 private static Information raquette;
 private static String message;
 private static Information theorique= new Information("epc",20,20);
  // url to get the position
private static String url_position ="http://adelaljane.netai.net/pfe/android.php";
// JSON Node names
private static final String TAG_DATA = "data";
private static final String TAG_SUCCESS = "success";
private static final String TAG_Y = "y";
private static final String TAG_Z = "z";
// position JSONArray
JSONArray info_php = null;


 private void manage(final Message msg) {
     // Building Parameters
     List<NameValuePair> params = new ArrayList<NameValuePair>();
         if (readTags == null) {
         readTags = new ArrayList<Inventory>();
     }
     String[] list = (String[]) msg.obj;
     for (int k = 0; k < (int) msg.arg1; k++) {
         boolean found = false;
         for (Inventory inv : readTags) {
             if (inv.getEpc().equals(list[k])) {
                 found = true;
             }
         }

         if (!found) {

             readTags.add(new Inventory(list[k], readTags.size()));

             //article id
             num=list[k];
              // getting JSON string from URL
             JSONObject json = jParser.makeHttpRequest(url_position, "GET", params);

             // Check your log cat for JSON reponse
             Log.d("parametre: ", json.toString());
             try {
                 // Checking for SUCCESS TAG
                 int success = json.getInt(TAG_SUCCESS);
```

Figure 19 : Fonction « manage »

- Tout d'abord nous avons déclaré toutes les variables dont nous aurons besoin dans la fonction.
- Ensuite nous récupérons les informations du tag RFID, puis on vérifie que ce tag n'a pas encore été scanné car cela ne sert à rien de traiter plusieurs fois les informations pour un même produit. S'il a déjà été scanné alors nous ne faisons rien, sinon on stocke l'information dans la variable "num" et on commence la partie communication et comparaison.

3.3.4. Communication et comparaison

Dans cette partie, nous exposons les réalisations effectuées pour permettre la communication entre la partie identification et la partie localisation, et nous détaillons les développements permettant la comparaison d'un rayon théorique à l'implantation réelle.

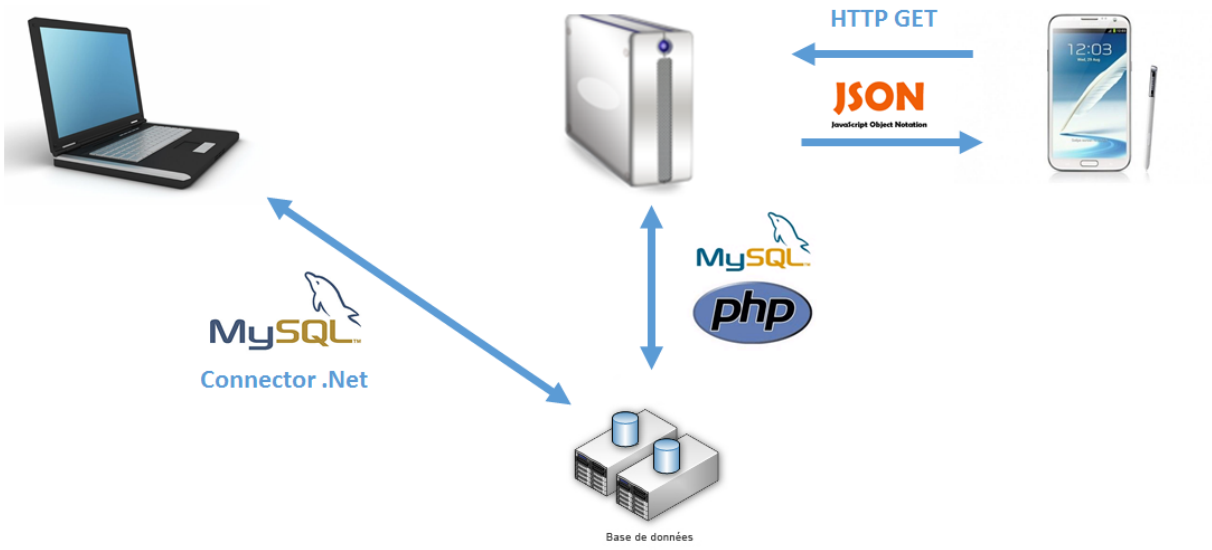

Figure 20 : Protocoles de communication

3.3.4.1. Communication via une base de données mysql

Nous avons créé une mini base de données mysql pour permettre de dialoguer entre les deux parties. Dans cette base de donné, nous stockons en continue les positions détecté par la kinect. Ces postions serviront ensuite de repère pour la partie comparaison.

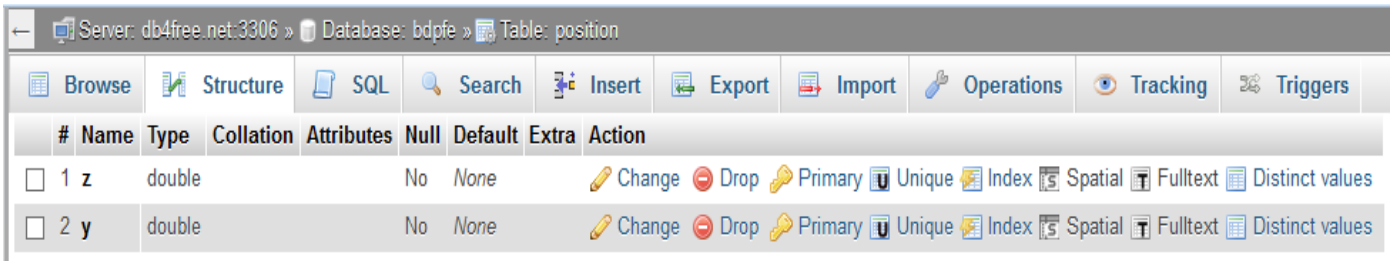

Figure 20 : Structure de la base de données

3.3.4.2. Comparaison entre le rayon réel et le rayon théorique

La comparaison entre le rayon réel et théorique se fera par le biais de l'application android. Initialement nous pensions la faire sur une application indépendante mais il aurait fallu mettre un place un système qui detecte qu'un tag a été scanné puis recupere les informations de la kinect. Nous nous somme rendu compte qu'il serait alors plus simple de faire la comparaison directement sur le téléphone car nous savons grâce à la fonction « manage » quand un tag est détecté alors nous pouvons directement le traiter.
Nous arrivions déjà à récupérer les informations des tags RFID et à les stocker dans la variable « num », à présent nous allons recuperer les informations de position de l'objet qui ont été stocké sur la base de donné.

D'abord nous allons créer un script php qui nous permettra de récupérer les informations de la base de données et ensuite de les envoyer à l'application android grace au JavaScript Object Notion (JSON).
Le Json est est un format de données textuelles, générique, dérivé de la notation des objets du langage ECMAScript. Il permet de représenter de l'information structurée. JSON est construit sur deux types d'éléments structurels

- une liste de paires
- une liste ordonnée de valeurs

Voici un exemple de format JSON:

Format JSON :

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New", "onclick": "CreateNewDoc()" },
                { "value": "Open", "onclick": "OpenDoc()" },
                { "value": "Close", "onclick": "CloseDoc()" }
            ]
        }
    }
}
```

Figure 22 : Format JSON

Le script php se présente de la manière suivante :

```php
<?php
// array for JSON response
$response = array();
//connexion to the database
$con=mysqli_connect("db4free.net","adel","flash30n","bdpfe");
if (mysqli_connect_errno($con))
{

   echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//resuparation of the position
$result = mysqli_query($con,"SELECT * FROM position" ) or die(mysql_error());
if (mysqli_num_rows($result) > 0){
$row = mysqli_fetch_array($result);
$data=array();
$data["z"] = $row['z'];
$data["y"]=$row['y'];
//success tag
$response["success"]=1;
$response["data"]=array();
array_push($response["data"], $data);
//sending
echo json_encode($response);}

else {
//error tag
$response["success"]=0;
$response["message"]="la base de donée est vide";
//sending
echo json_encode($response);}
mysqli_close($con);
?>
```

Figure 23 : Script php

- D'abord il faut se connecter à la base de données.
- Ensuite il faut récupérer les informations relatives aux positions et les stocker dans un tableau "data".
- Puis on initialise le tag de succès à 1 et on stocke le tout dans la variable "$response".
- Et pour finir on encode les informations en json grâce à la fonction "json_encode".
- Si le base de données est vide, on renvoi un message d'erreur et le tag succès est mis à 0.

Le format Json associé à notre script php est le suivant :
{"success":1,"data":[{"z":"0","y":"0"}]}

Ces informations seront récuperé puis décodé et enfin traité par l'application android. Pour faire cela, nous allons juste modifier la fonction "Manage".

- Tout d'abord, il faut récupérer et decoder les informations en JSON.

  Pour cela nous créons la classe "JSONPARSER" qui nous permet de nous connecter à la base grace à une URL, de recuperer les informations JSON et de les parser en un "JSONObjet"

Cette classe de definit comme suit:

```
public class JSONParser {

    static InputStream is = null;
    static JSONObject jObj = null;
    static String json = "";

    // constructor
    public JSONParser() {

    }

    public JSONObject getJSONFromUrl(String url) {
Log.d("JSON", "URL ="+url);

        // Making HTTP request
        try {
            // defaultHttpClient
            DefaultHttpClient httpClient = new DefaultHttpClient();
           // HttpPost httpPost = new HttpPost(url);
HttpGet http = new HttpGet(url);
            HttpResponse httpResponse = httpClient.execute(http);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                    is, "iso-8859-1"), 8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();
            json = sb.toString();
            System.out.println("json"+json);
            Log.i("json",json);
        } catch (Exception e) {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }

        // try parse the string to a JSON object
        try {
            jObj = new JSONObject(json);
        } catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }

    return jObj;

    }
}
```

Figure 24 : Class JSONParser

➢ Ensuite nous avons modifié la fonction « manage » :

```
//article id
num=list[k];
 // getting JSON string from URL
JSONObject json = jParser.getJSONFromUrl(url_position);

// Check your log cat for JSON reponse
Log.d("parametre: ", json.toString());
try {
    // Checking for SUCCESS TAG
    int success = json.getInt(TAG_SUCCESS);
    //reading the data
    JSONObject c = info_php.getJSONObject(0);
    if (success == 1) {
        // position found
        info_php = json.getJSONArray(TAG_DATA);

        // Storing each json item in variable
        double poz_y = c.getDouble(TAG_Y);
        double poz_z = c.getDouble(TAG_Z);
        raquette=new Information(num,poz_y,poz_z);
        if(raquette.CompareInfo(theorique)){
            message="Cet article est bien placé";
        }
        else{message="cet article est mal placé";}
    }
    else{message=c.getString(TAG_MESSAGE);}
} catch (JSONException e) {
    e.printStackTrace();
}

AlertDialog.Builder adb = new Builder(this);
AlertDialog myDialog = adb.create();
myDialog.setTitle("Information");
myDialog.setMessage(message);
myDialog.setButton(DialogInterface.BUTTON_NEUTRAL, "Ok",new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog,int id) {
        // if this button is clicked, just close
        // the dialog box and do nothing
        dialog.cancel();
    }
});
myDialog.show();


}
updateReadTags();
```

Figure 24 : Fonction « manage »

- Tout d'abord on test le tag succès pour savoir si la base de donné est vide.
- S'il est égal à 1 alors on a bien récupéré les informations de la base. On peut donc extraire les positions. Sinon on affichera le message d'erreur.
- On récupère les positions dans deux variables, "poz_y "et "poz_z".
- A partir de ces variables et de l'epc récupéré avant on initialise la variable "raquette" de type "Information".
- Ensuite on la compare avec les valeurs théoriques de l'article.
- Si elles ont les même alors l'article est bien placé sinon il est mal placé.

Après le traitement, on affiche une boite d'information qui nous permet de savoir le résultat du traitement, c'est à dire si la base de donné est vide ou si l'article est bien ou mal placé.

4. Bilan et perspectives

Dans cette partie, nous dressons un bilan de projet pour présenter l'avancement de nos travaux et les futures perspectives des résultats obtenues.

4.1. Bilan

Nous dressons un bilan du travail effectué pendant ce projet et nous évaluons le projet (atteintes des objectifs, raisons des écarts…), en comparant les dispositions initialement prévues avec le déroulement réel du projet, et en portant un regard critique sur tous les aspects du projet pour en tirer des voies d'amélioration.

4.1.1. Atteinte de l'objectif

- Bilan technique :

Les résultats obtenues ont abouti à obtenir un prototype fonctionnel à 90% permettant de projeter le principe de vérification des rayons à l'aide d'une raquette rfid, d'une kinect et un smartphone.
En effet, les exigences du cahier de charge tel qu'elle est établi est globalement réalisés.

- Respect des charges :

Nous n'avons pas imposé des charges importantes ni pour l'école ni pour l'entreprise.

- Respect des délais :

Le délai est une contrainte forte pour ce projet car nous avons l'obligation de proposer un prototype avant la date de fin de PFE les délais sont en effet respectés.

4.1.2. Méthode et outils de gestion de projet

La gestion de projet au sein du binôme a été basé sur le principe de division des taches et des parties avec une coordination forte pour éviter les incohérences et le mauvais intégration.

Les acteurs chargés du contrôle ont été suffisamment présents et vigilants. en effet, notre tuteur école nous a conseillé sur la prise de décision technique, cependant notre tuteur entreprise a suivi de près le déroulement de la réalisation de projet puisque on a longuement travaillé dans les locaux de l'entreprise Oxylane à btwin village.

4.2. Perspective

Les perspectives et les suites possibles pour ce projet sont très prometteuses car il permet d'exposer une première preuve de concept d'usage sans obligation d'implantation immédiate.
Cela permet aux ingénieurs d'Oxylane d'évoluer le système pour l'intégrer dans la réalité de terrain.

Conclusion

Ce projet est très instructif d'un part car c'est un projet industriel et aussi car nous participons à la conception total du prototype, en passant par les étapes de réflexion, de recherche et de développement. Après avoir redéfini le cahier des charge, nous avons pu nous concentrer sur la réalisation du nouveau prototype et le finir dans les temps malgré les quelques semaines perdu sur le premier prototype. Nous avons fait un réel travail d'ingénieur.