

Borne NFC pour le commerce

By Zhibin LIN



PLAN

1,Présentation de projet -----	3
2,Préparation avant le projet -----	3
3,Diagramme de réalisation -----	4
4,Android -----	5
5,Base de donnée SQLite -----	6
6,QRCode -----	9
7,NFC -----	13
7,1NFC Android	
7,2,Avant de faire le programme	
7,3,foreground dispatch system	
8,NFC Sheild Arduino -----	17
9,Difficulté et Resultat -----	19
10,Annex -----	19

1,Présentation de projet

Encadrants : Thomas Vantroys

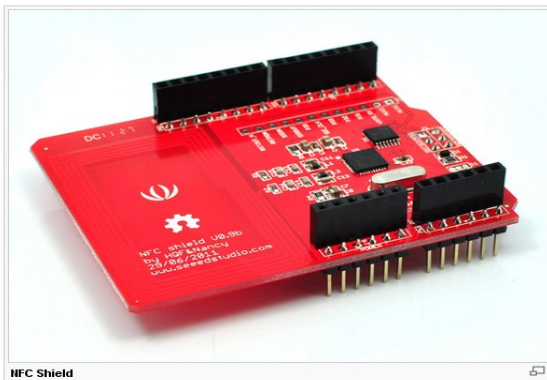
Objectif : Réaliser une application mobile permettant de faciliter l'achat de produits.

Description : Le but de ce projet est de concevoir un dispositif permettant à un client de sélectionner des articles à son domicile sur le site web du vendeur puis de venir les examiner en magasin. Les articles sont enregistrés sur un mobile puis communiqués à une borne sur place via le protocole sans fil NFC. Ce projet est porté par Oxylane, département recherche et développement de Décathlon.

2,Préparation avant le projet

Matériel requis

- Shield NFC



- Téléphone NFC (Google Nexus S)



Connaissances prérequis

- Base de données
- JAVA
- PHP

3,Diagramme de réalisation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae metus ac leo ultricies placerat. Curabitur vel ipsum vitae urna ultricies rhoncus. Proin tempus felis nec turpis feugiat vitae vestibulum arcu faucibus. Donec dapibus mi at nisl mattis a malesuada lectus feugiat. Etiam ultricies adipiscing nisi, sed scelerisque mauris ultricies sagittis. Proin blandit dignissim magna, id consectetur justo molestie in. Nullam id nunc et sem dictum malesuada ac et libero. Integer vitae mattis est. Nulla sodales dolor nec metus tempor a vestibulum nisl

Fonction utilisé(class TextView):

```
final void setText(CharSequence text)  
CharSequence getText()
```

5,Base de donnée SQLite

Réaliser une petite application laquelle nous permette de gérer les bases de données par SQLite .

Cette petite application qui nous permette de créer , insérer , supprimer , chercher les bases de données ce qu'on veut .

On peut regarder le contenu de base de données par entrer dans le terminal de Windows (Ctrl+R)



Taper la commande adb shell , qui nous permette d'entrer dans le terminal de téléphone (laquelle est constituée de LINUX).

on va aller dans le chemin ./data/data , ce répertoire est pour stocker les fichiers de Base de Données de chaque app.

On peut choisir notre base de données et entrer par shell -3

puis on peut taper les commandes de SQL pour faire la manipulation ce qu'on veut

comme .schema nous permette d'afficher la structure de notre base de données

select * from user ; nous permette de regarder les contenus de cette table

Class utilisé:

Class SQLite Helper

- [getReadableDatabase\(\)](#)
- [getWritableDatabase\(\)](#)
- [onCreate\(SQLiteDatabase db\)](#)
- [onOpen\(SQLiteDatabase db\)](#)
- [onUpgrade\(SQLiteDatabase db, int oldVersion, int newVersion\)](#)
- [close\(\)](#)

public class SQLiteDatabase

Class Overview

Exposes methods to manage a SQLite database.

SQLiteDatabase has methods to create, delete, execute SQL commands, and perform other common database management tasks.

See the Notepad sample application in the SDK for an example of creating and managing a database.

Database names must be unique within an application, not across all applications.

On peut utiliser cette class de faire la manipulation par langage **SQL**

[insert\(String table, String nullColumnHack, ContentValues values\)](#)

[Cursor](#)

[query\(String table, String\[\] columns, String selection, String\[\] selectionArgs, String groupBy, String having, String orderBy\)](#)

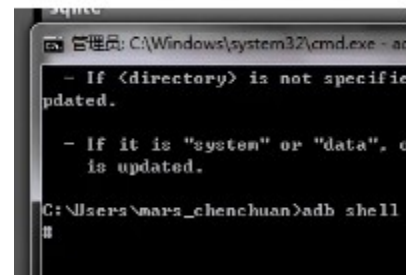
D'abord je testais un exemple

On peut constater qu'on a créé cinq boutons poussoirs qui nous permet de faire les manipulation SQLite

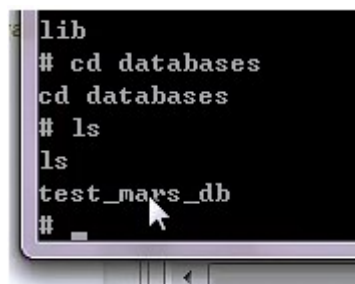


on tape adb shell qui nous permette d'entrer dans le terminal de téléphone android

Après on crée les base de donnée par createDatabase



on trouve que



dans le répertoire est a déjà eu dataBase qu'on a créé,

on tape le bouton poussoir **Insert**

et dans le terminal de WINDOWS , on a vu que qu'il y a une liste créé


```
select * from user;
1!zhangsan
sqlite> select * from user;
select * from user;
1!zhangsanfeng
sqlite>
```

Note:Le programme je vais laisser dans l'annex 1

WAMP:On installe le serveur Apache par Wamp, puis on peut mettre le fichier web dans le répertoire www , on utilise le navigateur qui nous permette d'afficher les contenus

6,QRCode

On a testé la communication entre le téléphone mobile et l'ordinateur par le socket (annexe 2), mais grâce à la rappel de prof , ce n'est pas la meilleure solution , la meilleure solution est on utilise le QR Barcode de donner les infos vers le téléphone mobile .



Sur la génération de QR Barcode , il est déjà existé sur le site

<http://phpqrcode.sourceforge.net/>

on va aller sur ce site de télécharger le code qui peut nous laisser d'utiliser PHP de générer le QR Code.

De la part de téléphone mobile , on va aller sur le site ci-dessous de télécharger une application BarcodeScanner qui nous permette de scanner le QRCode de l'écran de l'ordinateur.

<http://code.google.com/p/zxing/downloads/detail?name=BarcodeScanner4.0.apk>

On a testé , ça marche très bien , l'app va utiliser la caméra , quand on cible la caméra vers QRCode barre , les données(tous forme de texte) va être scanné .

Maintenant ce qu'on va faire est de créer une app qui nous permette de récupérer les données de QRCode , on utilise "intent" de faire la transmission de données .

D'abord il nous faut aller sur ce site de télécharger les 4 fichiers(les fonctions sont bien défini dedans) et les mettre dans la source de code sur eclipse

<http://code.google.com/p/zxing/source/browse/trunk/android-integration/src/com/google/zxing/integration/android/>

on va trouver

```
import android.support.v4.app.Fragment;
```

il y a erreur .

c'est parce qu'on a pas importé dans notre projet

on cherche android-support-v4.jar sous le répertoire SDK

puis taper alt+Enter ->java Build Path->Add External JARS on peut insérer ce jar dans eclipse , l'erreur va disparaître .

on peut commencer d'écrire notre programme.

le programme n'est pas difficile

on intancie

```
IntentIntegrator integrator = null;
```

```
integrator = new IntentIntegrator(CodeBarreActivity.this);
```

```
integrator.initiateScan();
```

on crée une fonction

```
public void onActivityResult(int requestCode, int resultCode, Intent intent) {  
    IntentResult scanResult =  
IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);  
    if (scanResult != null) {  
        Log.i("info", scanResult.getContents());  
        // handle scan result  
    }  
    // else continue with any other code you need in the method  
}
```

on peut obtenir le résultat par scanResult.getContents().

Note :.le code est sur l'annexe3

quand on exécute ce programme , d'abord le programme va lancer l'app Barcode Scanner , quand on a réussi de scanner , l'app va revenir laquelle qu'on a lancé et afficher le résultat .

On générè un QRCode



Data: ECC: Size:

note:virgule est séparation nom marchandise et prix , point virgule est la séparation de marchandise , on utilise la fonction split qui nous permet de faire la séparation de chaine de caractères

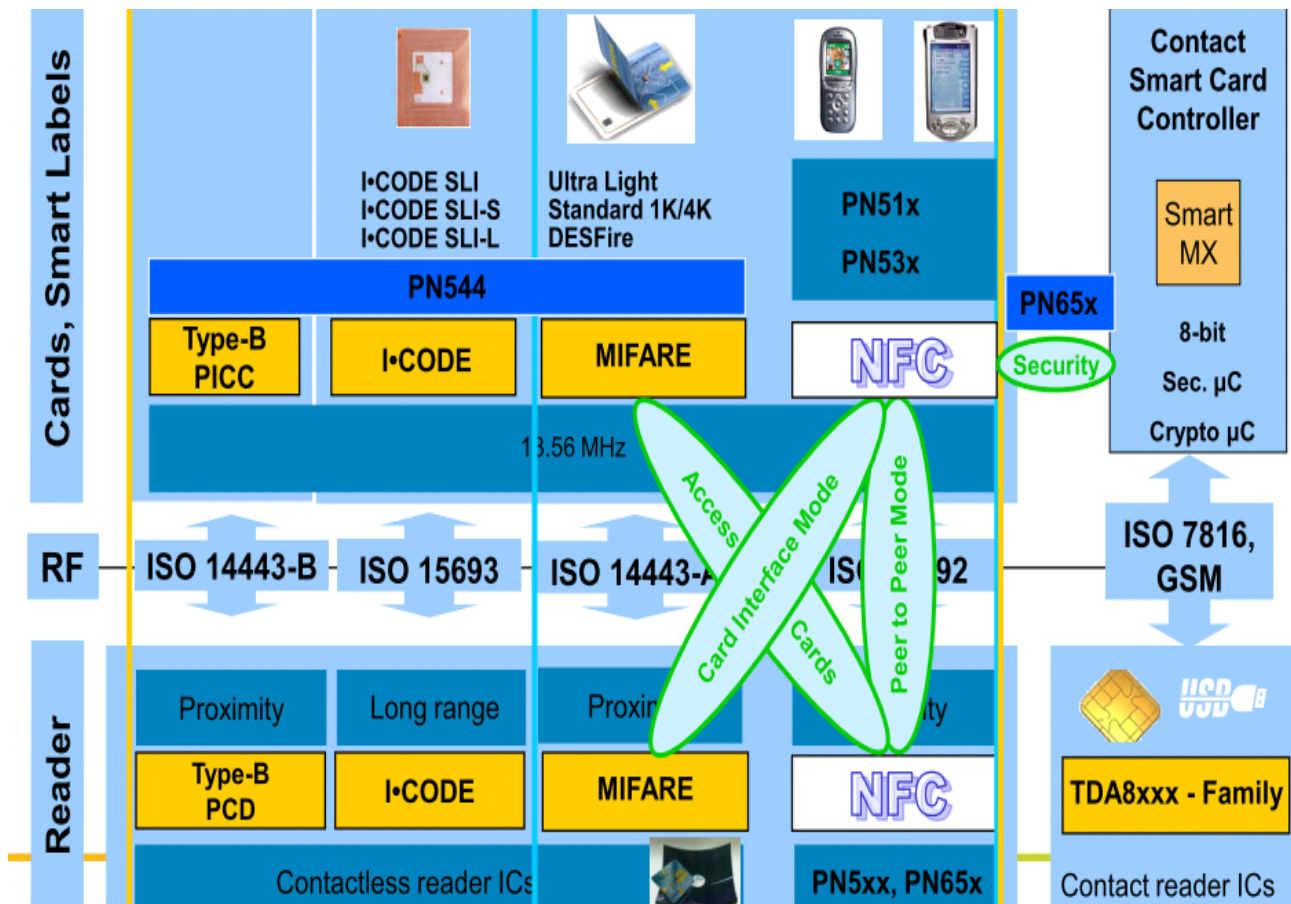


qui peut nous aidé d'afficher

Puis on stocke les infos dans database par le programme ce qu'on a écrit avant

7,NFC

La communication en champ proche (*Near Field Communication*), habituellement appelée **NFC**, est une technologie de communication sans-fil à courte portée et haute fréquence, permettant l'échange d'informations entre des périphériques jusqu'à une distance d'environ 10 cm.



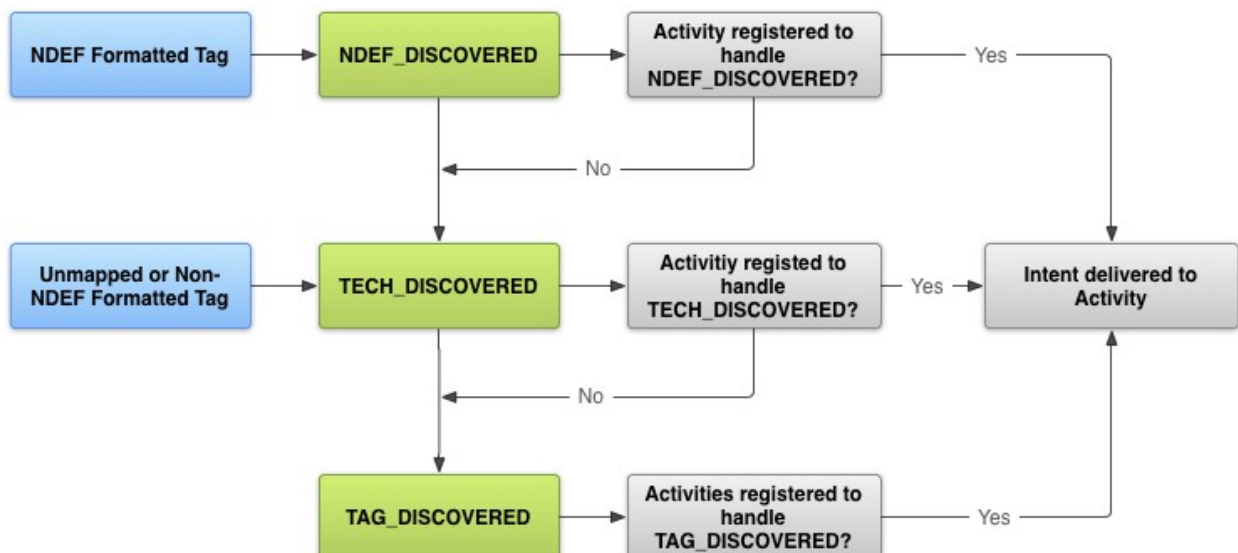
La référence de standard

Pour différents appareils, on doit utiliser différents standards de communication.

7,1 NFC Android

On peut regarder sur le SDK Google il nous a bien fourni les fonction qui nous permettant de faire la communication avec l'autre appareil NFC ,

Maintenant on doit apprendre un peu le façon de dispatch qu'il nous a fourni



Lorsque le système de répartition tag est fini de créer une intention qui encapsule le tag NFC et ses informations d'identification, il envoie l'intention d'une demande intéressée que les filtres pour l'intention. Si plus d'une application peut gérer l'intention, le Sélecteur d'activité est présenté afin que l'utilisateur peut sélectionner l'activité. Le système de répartition balise définit trois intentions, qui sont énumérés par ordre décroissant de priorité la plus basse:

- 1, ACTION_NDEF_DISCOVERED: Cette intention est utilisé pour démarrer une activité où une étiquette qui contient une charge utile NDEF est numérisé et est d'un type reconnu. C'est l'intention la plus haute priorité, et le système de répartition tag tente de démarrer une activité avec cette intention avant toute autre intention, chaque fois que possible.
- 2, ACTION_TECH_DISCOVERED: Si aucune activité enregistré pour gérer l'intention ACTION_NDEF_DISCOVERED, le système de répartition tag tente de démarrer une application avec cette intention. Cette intention est également directement commencé (sans en commençant d'abord ACTION_NDEF_DISCOVERED) si la balise qui est balayé contient des données NDEF qui ne peuvent pas être mappés à un type MIME ou d'une URI, ou si l'étiquette ne contient pas de données NDEF mais il est d'une technologie connue tag.
- 3, ACTION_TAG_DISCOVERED: Cette intention est démarré si aucune activité gérer les intentions ACTION_NDEF_DISCOVERED ou ACTION_TECH_DISCOVERED.

La méthode de base du système de répartition balise fonctionne comme suit:

- 1, Essayez de démarrer une activité avec l'intention qui a été créé par le système de répartition tag

lors de l'analyse du tag NFC (soit ACTION_NDEF_DISCOVERED ou ACTION_TECH_DISCOVERED).

2, Si aucune activité de filtres pour que l'intention, essayez de démarrer une activité avec l'intention priorité la plus basse suivante (soit ACTION_TECH_DISCOVERED ou ACTION_TAG_DISCOVERED) jusqu'à ce qu'un des filtres d'application pour l'intention ou jusqu'à ce que le système de répartition tag tente toutes les intentions possibles.

3, Si aucune des applications de filtrage pour l'une des intentions, ne rien faire.

7,2 Avant de faire le programme NFC

Avant de faire le programme, il nous faut donner le droit vers android

<uses-permission android:name="android.permission.NFC" />
dans Manifest.xml

**une app qui peut détecter la présence de Mifare carte que le prof fourni ,
l'app est vraiment facile mais qu'il y a des choses qu'il faut faire attention .**

L'étape de réalisation est suivant

1. sous le logiciel eclipse , dans le projet , AndroidManifest.xml , on met un filtre sous l'activity correspondant

```
                <intent-filter>
                    <action
android:name="android.nfc.action.TECH_DISCOVERED" />
                </intent-filter>

                <meta-data
android:name="android.nfc.action.TECH_DISCOVERED"
                android:resource="@xml/nfc_tech_filter"
                />
```

ici quand l'action a reçu une action TECH_DISCOVER , il va le comparer si cette action est l'action de démarrage de APP ou pas , si oui , l'app va être démarré automatiquement .

sous le répertoire res->xml, on crée un fichier de nfc_tech_filter.xml

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <tech-list><tech>android.nfc.tech.IsoDep</tech></tech-list>
    <tech-list><tech>android.nfc.tech.NfcA</tech></tech-list>
    <tech-list><tech>android.nfc.tech.NfcB</tech></tech-list>
    <tech-list><tech>android.nfc.tech.NfcF</tech></tech-list>
    <tech-list><tech>android.nfc.tech.NfcV</tech></tech-list>
    <tech-list><tech>android.nfc.tech.Ndef</tech></tech-list>
    <tech-list><tech>android.nfc.tech.NdefFormatable</tech></tech-list>
    <tech-list><tech>android.nfc.tech.MifareClassic</tech></tech-list>
    <tech-list><tech>android.nfc.tech.MifareUltralight</tech></tech-list>
</resources>
```

attention , la forme que sdk nous donne est

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
```

```
<tech-list>
  <tech>android.nfc.tech.IsoDep</tech>
  <tech>android.nfc.tech.NfcA</tech>
  <tech>android.nfc.tech.NfcB</tech>
  <tech>android.nfc.tech.NfcF</tech>
  <tech>android.nfc.tech.NfcV</tech>
  <tech>android.nfc.tech.Ndef</tech>
  <tech>android.nfc.tech.NdefFormatable</tech>
  <tech>android.nfc.tech.MifareClassic</tech>
  <tech>android.nfc.tech.MifareUltralight</tech>
</tech-list>
```

</resources>

dans ce cas , ça ne marche pas , ici ça veut dire que le device ici qu'il doit pouvoir de supporter tous les standard simultanément.

mais le premier cas dit que si il y a un de ces standards est détecté , il va réveiller app correspondant .

et après cette configuration , ça peut marcher pour le téléphone de détecter le mifare carte et récupérer les infos.

Note:Le code est dans l'annexe

7,3 foreground dispatch system

ce qui permet une activité de premier plan à avoir la priorité quand un tag NFC est découvert. Avec cette méthode, l'activité doit être au premier plan pour remplacer RAA et le système de répartition intention.

Note:Le code est dans l'annexe

8,NFC Shield Arduino

De la part de NFC shield , il y a déjà eu les programme d'exemples sur le site officiel , mais il a dit qu'ils n'ont pas testé avec téléphone mobile?

http://www.seeedstudio.com/wiki/index.php?title=NFC_Shield

En fait , le Arduino il nous a déjà fourni des codes pour faire la communication NFC P2P et ils ont réussi de faire le test entre 2 arduino

J'essai de chercher en ligne si il y a des personnes qu'ils sont en train de faire cette

projet ou pas , et ils ont tous disent qu'il n'a pas reussi . Et il y a des personnes
donnenet des propositions et je testais ça ne peux pas marcher non plus .

À la fin , je trouve sur le code,google,com il dit que pour faire la commencation P2P
Il vaux mieux d'utiliser le

NppFromACR122ToPhone



il y a des codes sources qui sont déjà testés et je regarde le programme il est le
programme ce que j'écrit avant ça veux dire que cette appaerille ACR122 peux bien
être en compatible avec android phone

site de code source

<http://code.google.com/p/ismb-npp-java/>

19,1 Difficulté rencontré

Ce projet , le problème le plus difficile est de chercher la solution pour faire la communication NFC et Android , comme en ligne , il y a trop peu de ressource , et les fonctions qu'il a fourni sur site officielle n'est pas suffisant(pas pareil avec doc API) , Et un autre problème est au-début on a pas encore touché le JAVA , pour SC , il me dépense beaucoup du temps de faire la recherche de JAVA puis Android.

19,2 Résultat:

Etablir le serveur par apache (fait)

base de données SQLite(fait)

QRCode (fait)

Android NFC avec milfare carte(fait)

Android NFC avec Arduino NFC Shield(échec)

Annex :

Le code au-dessous j'ai déjà le testé

I,Le code au-dessous nous permet de initialiser une base de donnée avec prix et nom de marchandise

```
package DB;

import android.app.Activity;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;

public class DatabaseHelper extends SQLiteOpenHelper {

    public static final int VERSION = 1;

    public DatabaseHelper(Context context, String name, CursorFactory factory,
```

```

        int version) {
            super(context, name, factory, version);
            // TODO Auto-generated constructor stub
        }

    public DatabaseHelper(Context context, String name, CursorFactory factory) {
        this(context, name, factory, VERSION);
    }

    public DatabaseHelper(Context context, String name) {
        this(context, name, null, VERSION);
    }

    public DatabaseHelper(Context context, String name, int version) {
        this(context, name, null, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub
        System.out.println("create a Database");
        db.execSQL("CREATE TABLE IF NOT EXISTS user (id integer primary key autoincrement,
product varchar(20), prize varchar(20))");
    }

    @Override
    //quand on insere une talbe, update une table , on a besoin de le faire
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub
        System.out.println("update a Database");
    }
}

```

Le code au-dessous est pour lancer le Qrode scanner puis rendre les

données dans une base de données

```
package com.codebarre.lin;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import DB.DatabaseHelper;
import android.app.Activity;
import android.app.ListActivity;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.util.Log;
import android.widget.SimpleAdapter;
import android.widget.TextView;
//il doit installer l'app Barcode Scanner(ou les app semblables)
public class CodeBarreActivity extends ListActivity {

    IntentIntegrator integrator = null;
    String result;
    /** Called when the activity is first created. */
    String products;
    String[][] product;
    String prize;
    TextView tv1;
    TextView tv2;
    TextView tv3;
    List<Map<String, Object>> list;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        list = new ArrayList<Map<String, Object>>();

        //initiate attente la resultat de Scan
        integrator = new IntentIntegrator(CodeBarreActivity.this);

        Log.i("info", "1");
        //initateScan va appelle la app QRCode barre
        integrator.initiateScan();

    }

    //This is where you will handle a scan result
    public void onActivityResult(int requestCode, int resultCode, Intent intent)
    {
        // else continue with any other code you need in the method
        Log.i("info", "2");
        IntentResult scanResult =
        IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);
        if (scanResult != null) {
            // handle scan result
            String[][] d;
            String s = scanResult.getContents();
        }
    }
}
```

```

Log.i("info",s+"");
String[] sFirst = s.split(";");
d = new String[sFirst.length][];

for(int i=0; i<sFirst.length; i++) {
    String[] sSecond = sFirst[i].split(",");

    d[i] = new String[sSecond.length];
    for(int j=0; j<sSecond.length; j++) {

        d[i][j] = new String(sSecond[j]);

    }
}

for(int i=0; i<d.length; i++) {
    Map<String, Object> map = new HashMap<String, Object>();
    for(int j=0; j<d[i].length; j++) {
        if(j == 0) {products = d[i][j];map.put("title",
d[i][j]);Log.i("info", "product" + d[i][j]);}
        if(j == 1) {prize = d[i][j];map.put("info", d[i]
[j]);Log.i("info", "prize" + d[i][j]);}

    }
    list.add(map);
    DatabaseHelper dbHelper = new
DatabaseHelper(this,"NFC_DB",2);
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    db.execSQL("insert into user(product, prize)
values(?,?)", new Object[]{products, prize});
    db.close();
}

}
SimpleAdapter adapter = new SimpleAdapter(this,list,R.layout.vlist,
    new String[]{"title","info"},
    new int[]{R.id.title,R.id.info});
setListAdapter(adapter);
}
}

```

FOREGROUND DISPATCHES SYSTEM

```

package com.NFC.lin;

```

```

import java.io.IOException;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.IntentFilter.MalformedMimeTypeException;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.MifareClassic;

```

```

import android.nfc.tech.MifareUltralight;
import android.nfc.tech.Ndef;
import android.nfc.tech.NdefFormatable;
import android.nfc.tech.NfcA;
import android.nfc.tech.NfcF;
import android.nfc.tech.NfcV;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

/**
 * An example of how to use the NFC foreground dispatch APIs. This will
intercept any MIME data
 * based NDEF dispatch as well as all dispatched for NfcF tags.
 */
public class ItsANFCActivity extends Activity {
    private NfcAdapter mAdapterer;
    private PendingIntent mPendingIntent;
    private IntentFilter[] mFilters;
    private String[][] mTechLists;
    private TextView mText;
    private int mCount = 0;
    private NfcAdapter nfcAdapter;
    private PendingIntent pendingIntent;
    private IntentFilter[] intentFiltersArray;
    private String[][] techListsArray;
    Button myButton;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);
        mText = (TextView) findViewById(R.id.text);
        mText.setText("Scan a tag");
        myButton = (Button) findViewById(R.id.MyButton);
        try{
            mAdapterer = NfcAdapter.getDefaultAdapter(this);
        }catch (NullPointerException ex){
            System.out.println("mAdapterer est null pointer");
        }

        setUpForegroundDispatchSystem(this);

        // Create a generic PendingIntent that will be deliver to this activity.
The NFC stack
        // will fill in the intent with the details of the discovered tag before
delivering to
        // this activity.
        mPendingIntent = PendingIntent.getActivity(this, 0,
            new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
    }

    @Override
    public void onResume() {
        super.onResume();
        if (mAdapterer != null){ mAdapterer.enableForegroundDispatch(this,
mPendingIntent, mFilters,

```

```

        mTechLists);
        mText.setText("foreground adapted");
    }
}

@Override
protected void onNewIntent(Intent intent) {
    Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
    NfcA myTag = NfcA.get(tag);

    try{
        myTag.connect();
        if (myTag.isConnected()) {
            byte[] data = myTag.transceive(new byte[]{ (byte)0x30,
(byte)0x00 });
        }
        myTag.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

@Override
public void onPause() {
    super.onPause();
    if (mAdapter != null) {
        mAdapter.disableForegroundDispatch(this);
        mText.setText("foreground disables");
    }
}

public void setUpForegroundDispatchSystem(Activity activity) {
    this.nfcAdapter = NfcAdapter.getDefaultAdapter(activity);

    this.pendingIntent = PendingIntent.getActivity(activity, 0, new
Intent(activity, activity.getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP),
0);

    IntentFilter ndef = new IntentFilter(NfcAdapter.ACTION_NDEF_DISCOVERED);
    this.intentFiltersArray = new IntentFilter[] { ndef };
    this.techListsArray = new String[][] {
        new String[] { MifareUltralight.class.getName(),
            Ndef.class.getName(), NfcA.class.getName() },
        new String[] { MifareClassic.class.getName(),
            Ndef.class.getName(), NfcA.class.getName() },
        new String[] { MifareUltralight.class.getName(),
            NdefFormatable.class.getName(), NfcA.class.getName() },
        new String[] { Ndef.class.getName(), NfcV.class.getName() },
        new String[] { NfcF.class.getName() } };
    mText.setText(Ndef.class.getName()+"");
    mText.setText("Ndef~~~~~");
}
}

```

