A30 - Concevoir et développer des applications mobiles

Rapport de projet

Version 1 du vendredi, 18 juin 2021

Ruffieux Lucas

Module du 20.05.2021 au 18.06.2021

EMF – Fribourg / Freiburg
Ecole des Métiers / Berufsfachschule
Technique / Technik
Section EMF-Informatique

Introduction

Nom de code du projet

CertiPhone

Plateforme de développement

Nous allons partir sur Android studio car il nous permettra d'exploiter au maximum les fonctionnalités d'Android, de plus nous pensons déjà à l'avenir et peut être par la suite créer une version IOS.

Synopsis

Ce projet a pour but de créer une application Android qui nous permettra d'identifier les personnes nous appelant même si elles ne sont pas dans nos contacts. Ce projet sera un projet groupé avec Guillaume Gonin et sera distribué comme suivant :

Monsieur Ruffieux (moi) :

- Contact
- Ihm Contact
- Code contact
- Gestion sms

Guillaume Gonin :

- Envoie de l'appel
- Réception de l'appel
- Code général
- Ihm général

Ces tâches ne sont pas fixe et peuvent changer de responsable lors de l'implémentation du projet car j'ai de la peine à évaluer le temps que prendra chaque tâche.

Analyse et conception

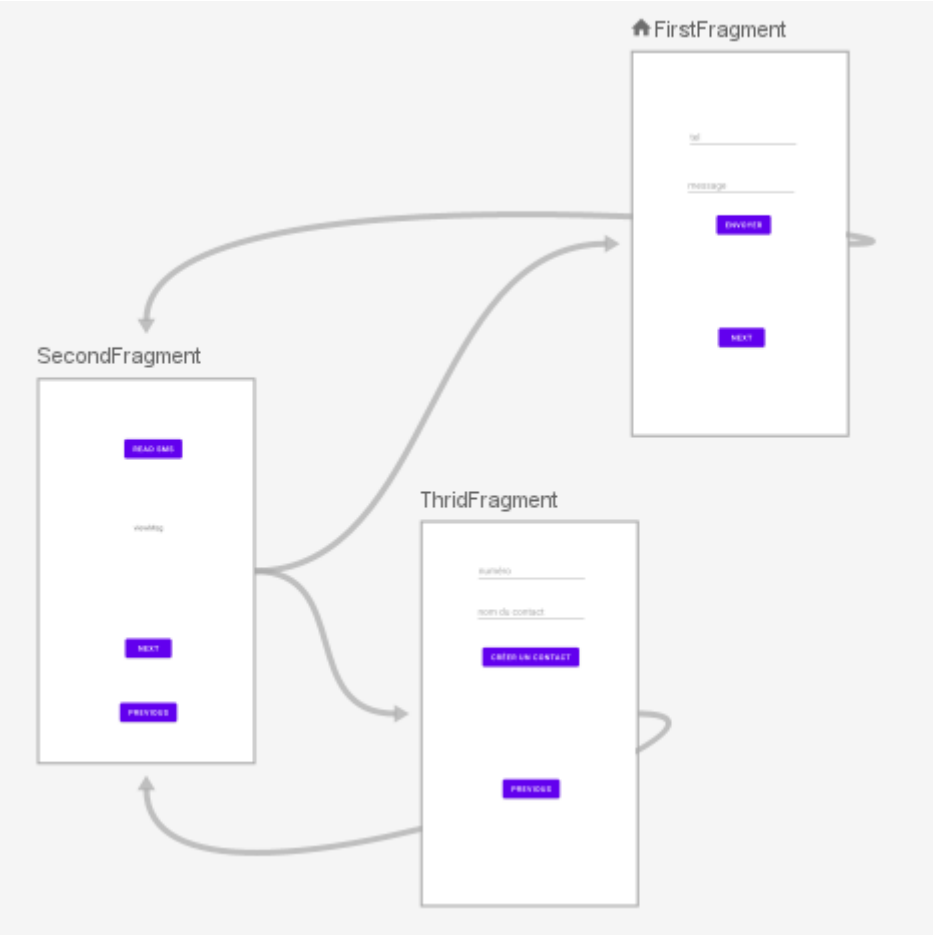
Description des fonctionnalités souhaitées

Je voudrais faire un projet qui a pour but d'identifier les utilisateurs émettant un appel en envoyant un sms contenant les informations de l'appelant. Pour ce faire je vais devoir envoyer et recevoir des sms, créer un nouveau contact.

Pour faire mon programme de démo j'imagine bien 3 fenêtre disposée ainsi :

De plus je voudrais créer des classes de type helper pour que je puisse facilement les utiliser dans d'autre projet.

Je veux pouvoir naviguer facilement entre les différentes vues, je veux que le programme soit facilement utilisable.



On a un mouvement d'une extrémité à l'autre.

Tests technologiques

Permets de définir si le projet est réalisable ou non.

Envois sms

But

Prouvé qu'on peut bien envoyer des sms avec les infos du contact. C'est la partie la plus importantes du projet.

Bout de code intéressant/important

Ajout des permissions dans le manifest :

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Demande des permissions :

```
send.setEnabled(false);
if(checkPermission(Manifest.permission.SEND_SMS)){
    send.setEnabled(true);
}else{
    ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.SEND_SMS}, SEND_SMS_PERMISSION_REQUEST_CODE);
}
```

```
public boolean checkPermission(String permission){
    int check = ContextCompat.checkSelfPermission(this,permission);
    return (check== PackageManager.PERMISSION_GRANTED);
}
```

onSend méthode :

```
    public void onSend(View v){
String phoneNumber= number.getText().toString();
String smsMessage= message.getText().toString();

if(phoneNumber==null || phoneNumber.length()==0 || smsMessage==null ||
smsMessage.length()==0){
    return;
}
if(checkPermission(Manifest.permission.SEND_SMS)){
    SmsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(phoneNumber, null, smsMessage, null,
null);
    Toast.makeText(this, "Message envoyé!", Toast.LENGTH_LONG).show();
} else {
    Toast.makeText(this, "Acces refusé!", Toast.LENGTH_LONG).show();

}
    }
```

Résultat de ce test

Il est très facile d'envoyer des SMS avec Android studio, de plus on peut assez facilement faire une liste de distribution de SMS en modifiant un tout petit peu le code.

Conclusions

L'envoi de SMS est entièrement réalisable et de ce fait

Lecture sms

But

Pouvoir lire les informations contact transmise pour pouvoir les traiter par la suite.

Bout de code intéressant/important

Ajout des permissions dans le manifest :

```
<uses-permission android:name="android.permission.READ_SMS"/>
```

Demande des permissions :

```
ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.READ_SMS},
PackageManager.PERMISSION_GRANTED);
```

Read_SMS méthode :

```
public void Read_SMS(View){

    Cursor = getContentResolver().query(Uri.parse("content://sms"), null,
null,null,null);
    cursor.moveToFirst();

    myTextView.setText(cursor.getString(12));

}
```

Résultat de ce test

On peut lire le dernier SMS reçu ou envoyé.

Conclusions

Il sera utile pour pouvoir récupérer les informations de l'appelant, il ne manque plus qu'un mettre en place une sécurité pour pouvoir être sur que le sms reçu nous concerne bien.

Affichage du contact

But

Pouvoir afficher le contact (le créer)

Bout de code intéressant/important

Création de l'ArrayList

```
ArrayList<ContentProviderOperation> ops = new
ArrayList<ContentProviderOperation>();
```

Ajout du type de compte

```
ops.add(ContentProviderOperation.newInsert(ContactsContract.RawContacts.C
ONTENT_URI).withValue(ContactsContract.RawContacts.ACCOUNT_TYPE,
null).withValue(ContactsContract.RawContacts.ACCOUNT_NAME,
null).build());
```

Ajout d'une information

```
ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE).withVa
lue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,
displayName).build());
```

Résultat de ce test

Conclusions

On peut facilement créer un contact une fois le code créer

Le code est puissant car il fait plusieurs choses comme ajouter un mail, un numéro de domicile.

Implémentation

Problèmes rencontrés

Interception du microphone

De basse nous devions faire un programme de chiffrement des appels mais nous n'avons jamais réussi à intercepter le flux du microphone pour le renvoyer modifier dans l'appel. De ce fait nous avons changé de projet dès la semaine une.

Lecture sms

Je n'ai pas réussi à trouver le moyen de mettre en place un listener ducoup ma méthode attend le fait que quelqu'un appelle la méthode pour lire le dernier sms envoyer où reçu

Résultat obtenu

Code de contacts

Le code a pour vocation de demandé les permissions pour avoir accès en lecture et en écriture au contact. Puis il créer un contact avec le/les paramètre spécifier, et ajoute le contact dans le répertoire de l'utilisateur.

```
package com.example.certiphonedemo.helper;

import android.Manifest;
import android.content.ContentProviderOperation;
import android.content.ContentResolver;
import android.content.Context;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.provider.ContactsContract;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.example.certiphonedemo.MainActivity;

import java.util.ArrayList;

public class AddContact extends AppCompatActivity {
    MainActivity;

    /**
     * Permet de vérifier si on a bien la permission de faire l'action
précisser
     *
     * @param context    le context de l'application qui a besoin des
permissions
     * @param permission la permission que l'on souhaite vérifier
     * @return true si la permission est accorder, sinon false
     */
    public boolean checkPermission(Context, String permission) {
        int check = ContextCompat.checkSelfPermission(context,
permission);
        return (check == PackageManager.PERMISSION_GRANTED);
    }

    /**
     * C'est le constructeur de la classe il permets de demander les
permission qu'il a besoin
     *
     * @param mainActivity le main activity pour les requêtes des
permissions
     */
    public AddContact(MainActivity) {
        this.mainActivity = mainActivity;
        if (!checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.READ_CONTACTS)) {
            ActivityCompat.requestPermissions(mainActivity, new
String[]{Manifest.permission.READ_CONTACTS,
Manifest.permission.WRITE_CONTACTS}, PackageManager.PERMISSION_GRANTED);
        }

    }

    /**
     * permets de créer un contact 2 informations, a condition que le
```

```
numéro de téléphone n'est pas déjà attribuer a un contact
     *
     * @param displayName  le nom du contact à créer
     * @param mobileNumber le numéro de téléphone du contact
     * @return true si le contact à été créer et false en cas d'erreur
     */
    public boolean add(String displayName, String mobileNumber) {
        return add(displayName, mobileNumber, null, null, null, null,
null);
    }

    /**
     * permets de créer un contact avec plusieurs informations, a
condition que le numéro de téléphone n'est pas déjà attribuer a un
contact
     *
     * @param displayName  le nom du contact à créer
     * @param mobileNumber le numéro de téléphone du contact
     * @param homeNumber   le numéro fix du contact (peut etre null si
inexistant)
     * @param workNumber   le numéro de travail du contact (peut etre
null si inexistant)
     * @param emailID      le mail du contact (peut etre null si
inexistant)
     * @param company      l'entreprise ou travail le contact (peut etre
null si inexistant)
     * @param jobTitle     l'emplois du contact, une compagny doit être
transmise (peut etre null si inexistant)
     * @return true si le contact à été créer et false en cas d'erreur
     */
    public boolean add(String displayName, String mobileNumber, String
homeNumber, String workNumber, String emailID, String company, String
jobTitle) {
        if (!checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.WRITE_CONTACTS)) {
            ActivityCompat.requestPermissions(mainActivity, new
String[]{Manifest.permission.WRITE_CONTACTS},
PackageManager.PERMISSION_GRANTED);
        }
        if (!checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.READ_CONTACTS)) {
            ActivityCompat.requestPermissions(mainActivity, new
String[]{Manifest.permission.READ_CONTACTS},
PackageManager.PERMISSION_GRANTED);
        }
        if (checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.READ_CONTACTS) &&
checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.WRITE_CONTACTS) &&
!contactExists(mainActivity.getApplicationContext(), mobileNumber)) {
            ArrayList<ContentProviderOperation> ops = new
ArrayList<ContentProviderOperation>();

ops.add(ContentProviderOperation.newInsert(ContactsContract.RawContacts.C
ONTENT_URI).withValue(ContactsContract.RawContacts.ACCOUNT_TYPE,
null).withValue(ContactsContract.RawContacts.ACCOUNT_NAME,
null).build());
            //------------------------------------------------------ nom
            if (displayName != null) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE).withVa
lue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME,
displayName).build());
            }
            //------------------------------------------------------
```

```
numéro mobile
            if (mobileNumber != null) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE).withValue(Conta
ctsContract.CommonDataKinds.Phone.NUMBER,
mobileNumber).withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE).build());
            }
            //------------------------------------------------------
numéro de domicile
            if (homeNumber != null) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE).withValue(Conta
ctsContract.CommonDataKinds.Phone.NUMBER,
homeNumber).withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_HOME).build());
            }
            //------------------------------------------------------
numéro de travail
            if (workNumber != null) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE).withValue(Conta
ctsContract.CommonDataKinds.Phone.NUMBER,
workNumber).withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_WORK).build());
            }
            //------------------------------------------------------
Email
            if (emailID != null) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE).withValue(Conta
ctsContract.CommonDataKinds.Email.DATA,
emailID).withValue(ContactsContract.CommonDataKinds.Email.TYPE,
ContactsContract.CommonDataKinds.Email.TYPE_WORK).build());
            }
            //------------------------------------------------------
Entreprise
            if (company != null && jobTitle != null) {
                if (!company.equals("") && !jobTitle.equals("")) {

ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_
URI).withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID,
0).withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Organization.CONTENT_ITEM_TYPE).withValu
e(ContactsContract.CommonDataKinds.Organization.COMPANY,
company).withValue(ContactsContract.CommonDataKinds.Organization.TYPE,
ContactsContract.CommonDataKinds.Organization.TYPE_WORK).withValue(Contac
tsContract.CommonDataKinds.Organization.TITLE,
jobTitle).withValue(ContactsContract.CommonDataKinds.Organization.TYPE,
ContactsContract.CommonDataKinds.Organization.TYPE_WORK).build());
                }
            }
            //Demande au contact provider de créer un contact
            try {

mainActivity.getContentResolver().applyBatch(ContactsContract.AUTHORITY,
```

```
ops);
                    return true;
                } catch (Exception e) {
                    e.printStackTrace();
                    Toast.makeText(mainActivity.getApplicationContext(),
e.toString(), Toast.LENGTH_SHORT).show();
                    return false;
                }
            } else {
                return false;
            }
        }

        /**
         * permets de vérifier si le numéro existe dans la liste des contacts
         *
         * @param context le contexte du mainActivity
         * @param number  le numéro de téléphone a vérifier
         * @return true si le numéro de téléphone est attribuer a un contact
         */
        public boolean contactExists(Context context, String number) {
            if (number != null) {
                ContentResolver cr = context.getContentResolver();
                Cursor curContacts =
cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null, null,
null, null);

                while (curContacts.moveToNext()) {
                    String contactNumber =
curContacts.getString(curContacts.getColumnIndex(ContactsContract.CommonD
ataKinds.Phone.NUMBER));
                    if (number.equals(contactNumber)) {
                        return true;
                    }
                }
                return false;
            } else {
                return false;
            }
        }
}
```

Code sms sender

Le code demande les permissions pour l'envois de SMS puis avec le sms manager il envois le sms au destinataire, avant d'envoyer il vérifie qu'il a bien les permissions de l'envoyer.

```
package com.example.certiphonedemo.helper;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.telephony.SmsManager;

import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.example.certiphonedemo.FirstFragment;

public class SmsSender {
    private FirstFragment firstFragment;
    public SmsSender(FirstFragment) {
        this.firstFragment=firstFragment;
        if (!checkPermission(firstFragment.getContext(),
Manifest.permission.SEND_SMS)) {

ActivityCompat.requestPermissions(firstFragment.getActivity(), new
String[]{Manifest.permission.SEND_SMS},
PackageManager.PERMISSION_GRANTED);
```

```
        }
    }

    public boolean onSend(String number, String sms){
        String phoneNumber= number;
        String smsMessage= sms;

        if(phoneNumber==null || phoneNumber.length()==0 ||
smsMessage==null || smsMessage.length()==0){
            return false;
        }
        if(checkPermission(firstFragment.getContext(),
Manifest.permission.SEND_SMS)){
            SmsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(phoneNumber, null, smsMessage,
null, null);
            return true;
        } else {
            return false;

        }
    }

    private boolean checkPermission(Context, String permission) {
        int check = ContextCompat.checkSelfPermission(context,
permission);
        return (check == PackageManager.PERMISSION_GRANTED);
    }
}
```

Code sms reader

Il lit le dernier sms qu'il a été envoyé ou reçu. Pour ce faire il utilise le contentResolver d'Android. Puis le filtre pour ne garder que les sms.

```
package com.example.certiphonedemo.helper;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.view.View;

import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.example.certiphonedemo.MainActivity;

public class SmsReader {
MainActivity;

    public SmsReader(MainActivity mainActivity) {
        this.mainActivity = mainActivity;
        if (!checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.READ_SMS)) {
            ActivityCompat.requestPermissions(mainActivity, new
String[]{Manifest.permission.READ_SMS},
PackageManager.PERMISSION_GRANTED);
        }
    }

    public String Read_SMS(){
        if (checkPermission(mainActivity.getApplicationContext(),
Manifest.permission.READ_SMS)) {
            Cursor cursor =
mainActivity.getContentResolver().query(Uri.parse("content://sms"), null,
```

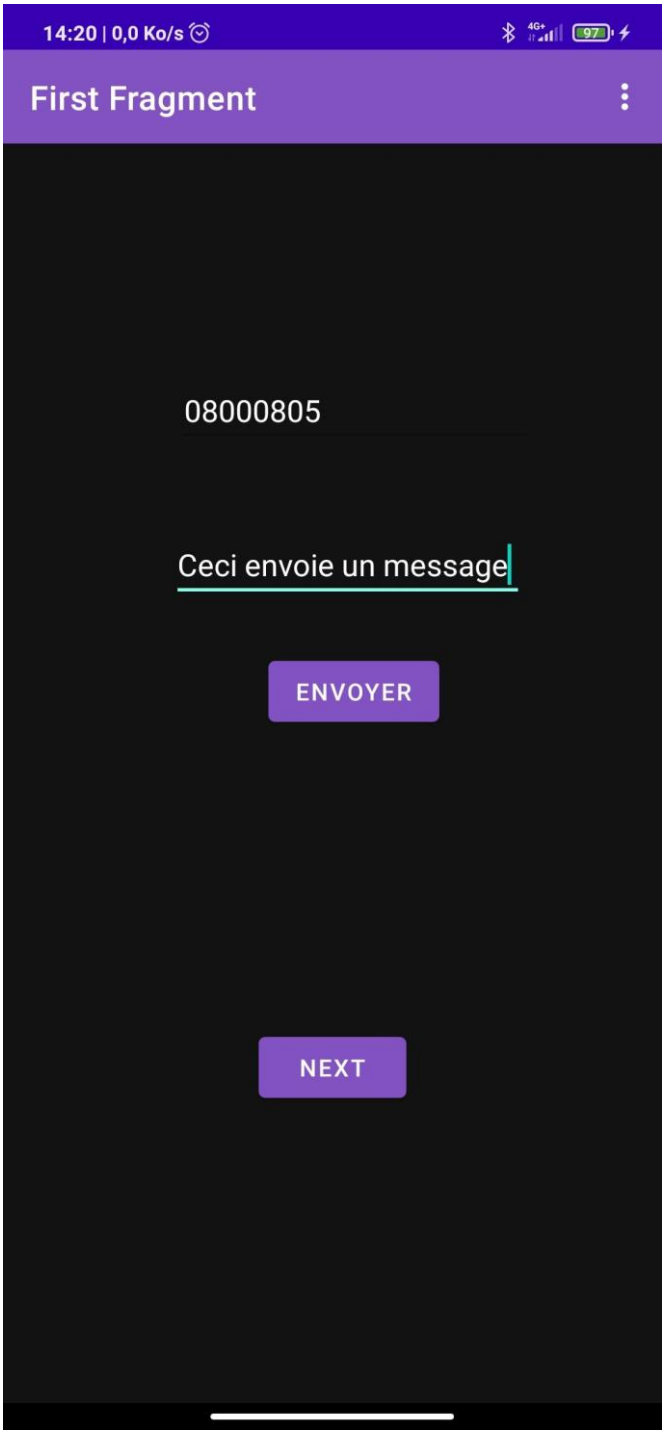
```
null, null, null);
            cursor.moveToFirst();

            return (cursor.getString(12));
        }
        return null;
    }
    private boolean checkPermission(Context context, String permission) {
        int check = ContextCompat.checkSelfPermission(context,
permission);
        return (check == PackageManager.PERMISSION_GRANTED);
    }
}
```

APK

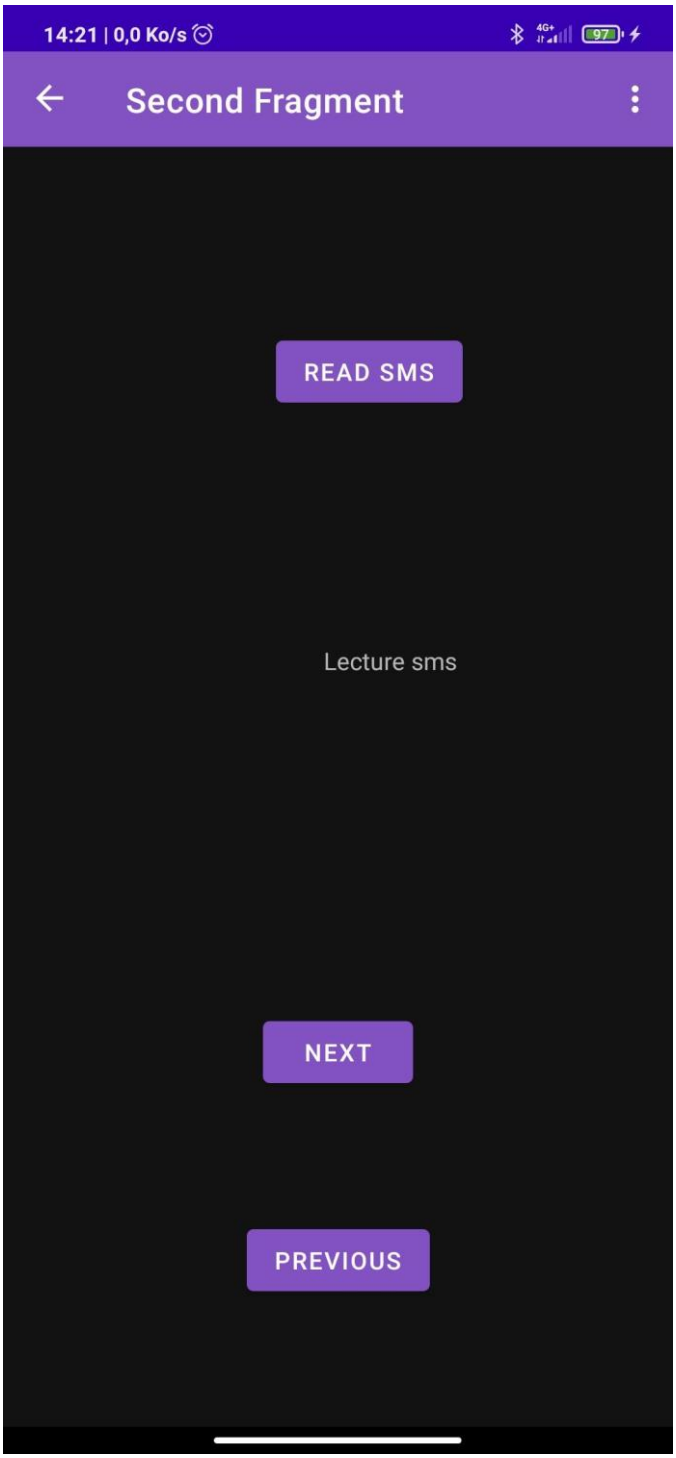
Voici le fichier APK :

app.apk

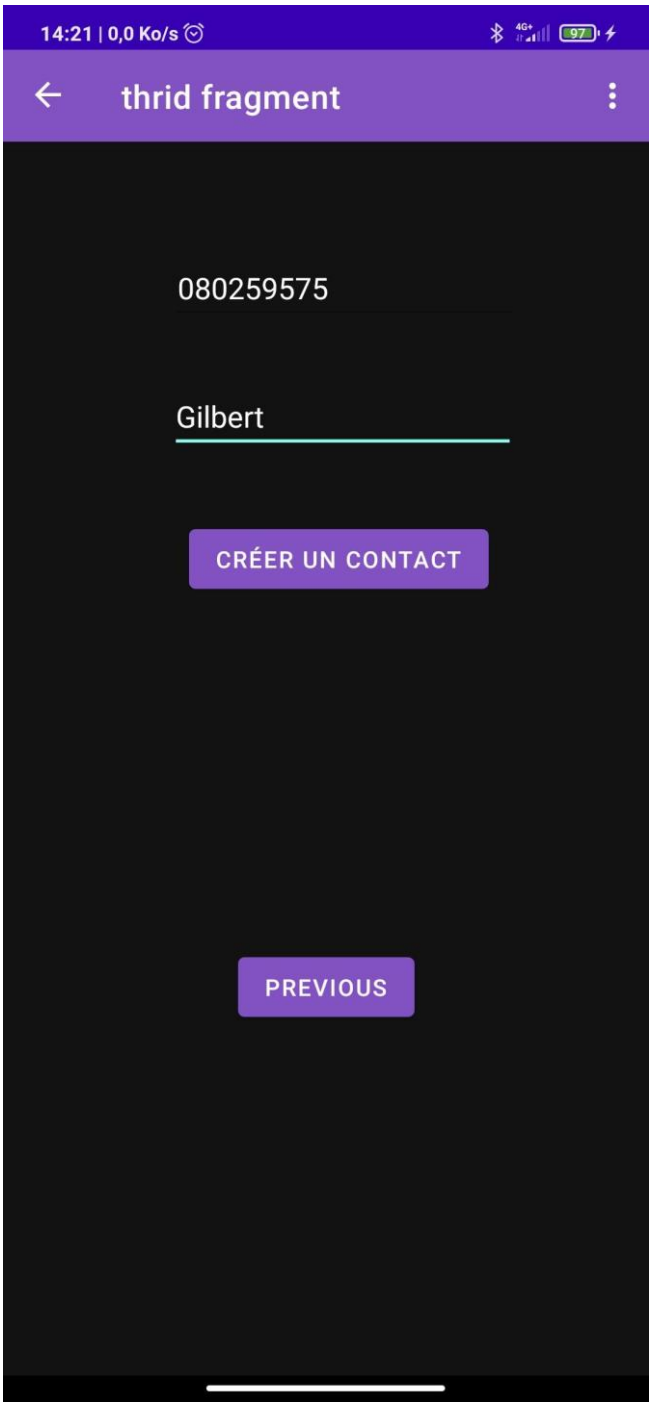
Voici la première fenêtre, celle qui envoie un sms :



Voici la deuxième fenêtre, celle qui lis un sms :



Voici la troisième fenêtre, celle qui ajoute un contact :



Tests

Tests fonctionnels

Envois sms (seul)	On peut envoyer un sms	OK
Réception sms (seul)	On peut lire le dernier sms	OK
Création de contact (seul)	Le contact est créer	OK
IHM	L'ihm est fonctionelle	OK
L'ensemble	Le tout fonctionne ensemble	OK

Conclusions

Ce que m’a apporté ce module

Il m’a apporté de nouvelle connaissance, j’ai découvert android Studio et j’ai pu expérimenter la programation mobile.

Ce que j’ai appris

J’ai appris la programation mobile, la recherche d’information fiable concernant un problème précis, j’ai ussi appris l’utilisation du mode collaboratif de github.

Ce que j’ai aimé

J’ai bien aimé faire un projet android, ça change des projet java standard. De plus le fait de travailler a deux nous a permis de se dépanner mutuellement en cas de bug.

Ce que je n’ai pas aimé

Les Jean-Michel codeur du dimanche qui donne des explications fausse sur les forums

Comment je me suis senti pendant ce module

Je me suis senti progressé, j’ai appris de nouvelle chose et j’ai envie d’en apprendre plus.

Auto-évaluation

Je trouve que j’ai bien travaillé , j’aurais pu faire mieux, mais pour une mise en bouche c’est bien. De plus je trouve que j’ai pleinement utilisé les capacités d’Android. J’airais pu mieux faire l’IHM mais je ne travaille pas à l’Eikon.