A30 – Concevoir et développer des applications mobiles

Documentation de projet

Version 1 du 12 juin 2023

Guillaume Dougoud

|  |  |
| --- | --- |
| S:\EMF\CPROF\Identite_EMF\2015-Identites\Logos des sections COULEURS\Logo_EMF-Informatique_FR_RVB_25.jpg | Module du 15.05.2023 au xx.xx.xxxx |

Table des matières

[1 Introduction 1](#_Toc137535368)

[2 Choix du projet 1](#_Toc137535369)

[2.1 Contexte 1](#_Toc137535370)

[2.2 Description 2](#_Toc137535371)

[2.3 Choix de technologie 2](#_Toc137535372)

[2.4 Schémas UML 2](#_Toc137535373)

[2.4.1 UseCase 3](#_Toc137535374)

[2.4.2 Maquette 3](#_Toc137535375)

[2.4.3 Entité relation 5](#_Toc137535376)

[2.4.4 Séquences Interactions 5](#_Toc137535377)

[2.5 Test Technologique 5](#_Toc137535378)

[2.5.1 Notification 5](#_Toc137535379)

[2.5.2 Photo 7](#_Toc137535380)

[2.5.3 Service REST 10](#_Toc137535381)

[3 Implémentation 12](#_Toc137535382)

[3.1 Dépendances 12](#_Toc137535383)

[3.2 Manifest 12](#_Toc137535384)

[3.3 Vue 12](#_Toc137535385)

[3.4 Descente de code 12](#_Toc137535386)

[4 Test 13](#_Toc137535387)

[5 Mise en production 13](#_Toc137535388)

[6 Conclusion 13](#_Toc137535389)

# Introduction

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

# Choix du projet

## Contexte

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## Description

Une application de gestion de bibliothèque qui permet d’accéder aux livres qui nous appartiennent (d’en ajouter, retirer et modifier), de voir les bibliothèques des autres utilisateurs et de faire des prêts de livres avec d’autres utilisateurs.

Une image contenant transport, texte, plein air, véhicule

Description générée automatiquement

## Choix de technologie

Liens utiles :

<https://openclassrooms.com/fr/courses/4517166-developpez-votre-premiere-application-android>

L’IDE **Android Studio** me permettra de faire la partie Client. La partie Server est déjà faite en php lors du projet du module 151 mais des adaptations vont être faites.

Pour les deux matériels du smartphone on peut utiliser l’appareil photo (ou la galerie) pour mettre une illustration du livre et une notification pour la demande d’emprunt.

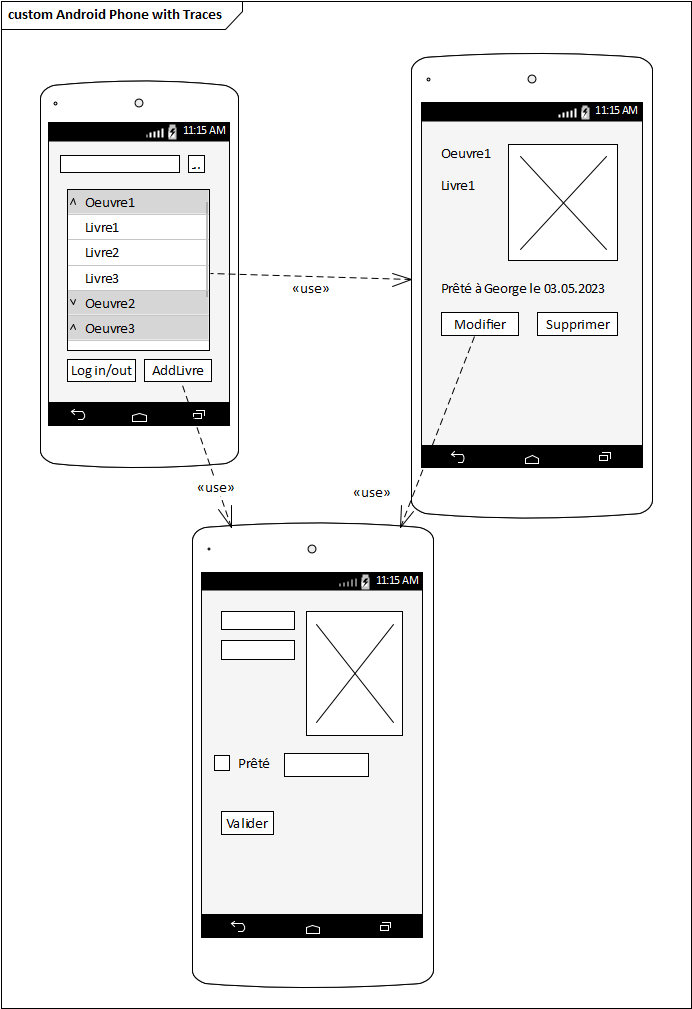
## Schémas UML

### UseCase

Une image contenant texte, diagramme, capture d’écran, ligne

Description générée automatiquement

### Maquette



### Entité relation



### Séquences Interactions

## Test Technologique

### Notification

Ajout d’un bouton dans le fichier activity\_main.xml

<Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/app\_name"  
 android:id="@+id/notify\_btn"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />

Ajout d’un écouteur sur le bouton et envoie d’une notification dans MainActivity.java

package com.example.testtechnonotify;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.content.pm.PackageManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
  
  
public class MainActivity extends AppCompatActivity {  
  
 Button notifyBtn;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 notifyBtn = findViewById(R.id.*notify\_btn*);  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 NotificationChannel channel = new NotificationChannel("My Notification", "My Notification", NotificationManager.*IMPORTANCE\_DEFAULT*);  
 NotificationManager manager = getSystemService(NotificationManager.class);  
 manager.createNotificationChannel(channel);  
 }  
  
 notifyBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.this, "My Notification");  
 builder.setContentTitle("My title");  
 builder.setContentText("Test Techno Notification");  
 builder.setSmallIcon(R.drawable.*ic\_launcher\_background*);  
 builder.setAutoCancel(true);  
  
 NotificationManagerCompat managerCompat = NotificationManagerCompat.*from*(MainActivity.this);  
  
 if (ActivityCompat.*checkSelfPermission*(MainActivity.this, android.Manifest.permission.*POST\_NOTIFICATIONS*) != PackageManager.*PERMISSION\_GRANTED*) {  
 // *TODO: Consider calling* // *ActivityCompat#requestPermissions* // here to request the missing permissions, and then overriding  
 // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
 // int[] grantResults)  
 // to handle the case where the user grants the permission. See the documentation  
 // for ActivityCompat#requestPermissions for more details.  
  
 return;  
 }  
 managerCompat.notify(1, builder.build());  
 }  
 });  
 }  
}

Résultat:

Une image contenant texte, Téléphone mobile, capture d’écran, gadget

Description générée automatiquement

### Photo

Ce test techno a été fait en suivant le tutoriel suivant :

<https://www.youtube.com/watch?v=LpL9akTG4hI&t=558s>

Ajout des droits sur l’appareil photo dans le manifeste :

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools" >  
 <!--Adding Camera, Write External Storage Permission-->  
 <uses-permission android:name="android.permission.CAMERA"  
 tools:ignore="PermissionImpliesUnsupportedChromeOsHardware" />  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

Ajout d’un bouton ainsi que d’une image view sur la vue :

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center\_horizontal"  
 tools:context=".MainActivity">  
  
 <ImageView  
 android:id="@+id/image\_view"  
 android:scaleType="centerCrop"  
 android:src="@drawable/baseline\_image\_24"  
 android:layout\_width="400dp"  
 android:layout\_height="400dp" />  
 <Button  
 android:id="@+id/capture\_image\_btn"  
 android:text="Capture Image"  
 style="@style/Base.Widget.AppCompat.Button.Colored"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
  
</LinearLayout>

Ajout des différentes méthodes appelées lors de la prise de photo :

public class MainActivity extends AppCompatActivity {  
  
 private static final int *PERMISSION\_CODE* = 1000;  
 private static final int *IMAGE\_CAPTURE\_CODE* = 1001;  
 Button mCaptureBtn;  
 ImageView mImageView;  
  
 Uri image\_uri;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 mImageView = findViewById(R.id.*image\_view*);  
 mCaptureBtn = findViewById(R.id.*capture\_image\_btn*);  
  
 mCaptureBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*N*){  
 if (checkSelfPermission(android.Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_DENIED* ||  
 checkSelfPermission(android.Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_DENIED* ){  
 String[] permission = {Manifest.permission.*CAMERA*, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
 requestPermissions(permission, *PERMISSION\_CODE*);  
 }  
 else {  
 //permission already granted  
 openCamera();  
 }  
 }  
 else {  
 //system os < Nougat  
 openCamera();  
 }  
 }  
 });  
 }  
  
 private void openCamera() {  
 ContentValues values = new ContentValues();  
 values.put(MediaStore.Images.Media.*TITLE*, "New Picture");  
 values.put(MediaStore.Images.Media.*DESCRIPTION*,"From the Camera");  
 image\_uri = getContentResolver().insert(MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*,values);  
  
 Intent cameraIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 cameraIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*,image\_uri);  
 startActivityForResult(cameraIntent,*IMAGE\_CAPTURE\_CODE*);  
 }  
  
 //handling permission result  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 //super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 switch (requestCode){  
 case *PERMISSION\_CODE*:{  
 if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*){  
 openCamera();  
 }  
 else {  
 Toast.*makeText*(this,"Permission denied...",Toast.*LENGTH\_SHORT*).show();  
 //*TODO "Take Picture with Camera - Android Studio - Java" 9:18* }  
 }  
 }  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 if (resultCode == *RESULT\_OK*){  
  
 mImageView.setImageURI(image\_uri);  
 }  
 }  
}

Résultat :

Une image contenant capture d’écran, Téléphone mobile, gadget, Appareil mobile

Description générée automatiquementUne image contenant capture d’écran, Rectangle, Ordinateur tablette, Téléphone mobile

Description générée automatiquement

### Service REST

Dans ce test techno nous allons envoyer des requêtes à notre service REST.php. Ce service a été fais lors du module 151 et est hébergé à l’adresse <http://dougoudg.emf-informatique.ch/151_personal-projet-DOUGOUD-Guillaume/Server/oeuvreManager.php>

La classe à ajouter à notre projet pour envoyer des requêtes :

package com.example.testtechnorest.rest;  
  
import com.loopj.android.http.AsyncHttpClient;  
import com.loopj.android.http.AsyncHttpResponseHandler;  
import com.loopj.android.http.RequestParams;  
  
public class HttpUtils {  
 private static AsyncHttpClient *client* = new AsyncHttpClient();  
 public static void get(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {  
 *client*.get(url, params, responseHandler);  
 }  
 public static void post(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {  
 *client*.post(url, params, responseHandler);  
 }  
 public static void put(String url, RequestParams params, AsyncHttpResponseHandler responseHandler){  
 *client*.put(url, params, responseHandler);  
 }  
}

La ligne à ajouter au Manifest pour pouvoir utiliser Internet :

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

Les deux dépendancies à ajouter dans build.gradle :

dependencies **{** implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
 implementation 'com.loopj.android:android-async-http:1.4.9'

Et un exemple de requête :

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 label = findViewById(R.id.*label\_REST*);  
  
 HttpUtils connection = new HttpUtils();  
  
 RequestParams params = new RequestParams();  
  
 params.put("fkutilisateur", 1);  
  
 connection.*get*("http://dougoudg.emf-informatique.ch/151\_personal-projet-DOUGOUD-Guillaume/Server/oeuvreManager.php", params, new TextHttpResponseHandler() {  
 @Override  
 public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwable) {  
 label.setText(statusCode+": "+responseString);  
 }  
  
 @Override  
 public void onSuccess(int statusCode, Header[] headers, String responseString) {  
 label.setText(responseString);  
 }  
 });  
  
}

Résultat (l’API nous retourne du xml) :

Une image contenant texte, capture d’écran, Appareil de communication, Appareil mobile

Description générée automatiquement

# Implémentation

Structure du projet et ajout :

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Les documents surlignés en bleu sont ceux qui ont été retouchés ou créé lors de ce projet.

## Dépendances

Dans le fichier build.gradle

//librairies permetant l'envoi de requêtes.  
implementation 'com.squareup.retrofit2:retrofit:2.9.0'  
implementation 'com.loopj.android:android-async-http:1.4.9'

## Manifest

Dans le fichier AndroidManifest.xml

<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.CAMERA"  
 tools:ignore="PermissionImpliesUnsupportedChromeOsHardware" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>  
<uses-permission android:name="android.permission.POST\_NOTIFICATIONS"/>

## Vue

Dans le fichier activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center\_horizontal"  
 tools:context=".MainActivity">  
  
 <ImageView  
 android:id="@+id/image\_view"  
 android:scaleType="centerCrop"  
 android:src="@drawable/baseline\_image\_24"  
 android:layout\_width="400dp"  
 android:layout\_height="400dp" />  
 <Button  
 android:id="@+id/capture\_image\_btn"  
 android:text="Capture Image"  
 style="@style/Base.Widget.AppCompat.Button.Colored"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>  
 <TextView  
 android:layout\_width="150dp"  
 android:layout\_height="300dp"  
 android:id="@+id/label\_REST"  
 />  
</LinearLayout>

Dans baseline\_image\_24.xml

<vector android:height="24dp" android:tint="#00FD3E"  
 android:viewportHeight="24" android:viewportWidth="24"  
 android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">  
 <path android:fillColor="@android:color/white" android:pathData="M21,19V5c0,-1.1 -0.9,-2 -2,-2H5c-1.1,0 -2,0.9 -2,2v14c0,1.1 0.9,2 2,2h14c1.1,0 2,-0.9 2,-2zM8.5,13.5l2.5,3.01L14.5,12l4.5,6H5l3.5,-4.5z"/>  
</vector>

## Descente de code

Une image contenant capture d’écran, texte, Police, Graphique

Description générée automatiquement

L’appui du bouton va appeler la méthode onClick

public void onClick(View v) {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*N*){  
 if (checkSelfPermission(android.Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_DENIED* ||  
 checkSelfPermission(android.Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_DENIED* ){  
 String[] permission = {Manifest.permission.*CAMERA*, Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
 requestPermissions(permission, *PERMISSION\_CODE*);  
  
  
 }  
 else {  
 //permission already granted  
 openCamera();  
 }  
 }  
 else {  
 //system os < Nougat  
 openCamera();  
 }  
}

La méthode onClick appel openCamera() qui vas ouvrir l’interface de l’appareil photo

private void openCamera() {  
 ContentValues values = new ContentValues();  
 values.put(MediaStore.Images.Media.*TITLE*, "New Picture");  
 values.put(MediaStore.Images.Media.*DESCRIPTION*,"From the Camera");  
 image\_uri = getContentResolver().insert(MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*,values);  
  
 Intent cameraIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 cameraIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*,image\_uri);  
 startActivityForResult(cameraIntent,*IMAGE\_CAPTURE\_CODE*);  
}

En validant la prise de photo, nous allons appeler la méthode onActivityResult() pour afficher l’image.

@Override  
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if (resultCode == *RESULT\_OK*) {  
  
 mImageView.setImageURI(image\_uri);  
 notify("Reussite","une photo a bien été prise");  
 getOeuvres();  
 } else notify("Echec","L'image n'a pas pu être affiché");  
  
}

Depuis là on vas créer une notification qui vas retourner l’erreur/réussite de la prise de photo.

public void notify(String title, String text) {  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.this, "My Notification");  
 builder.setContentTitle(title);  
 builder.setContentText(text);  
 builder.setSmallIcon(R.drawable.*ic\_launcher\_background*);  
 builder.setAutoCancel(true);  
  
 NotificationManagerCompat managerCompat = NotificationManagerCompat.*from*(MainActivity.this);  
  
 if (ActivityCompat.*checkSelfPermission*(MainActivity.this, android.Manifest.permission.*POST\_NOTIFICATIONS*) != PackageManager.*PERMISSION\_GRANTED*) {  
 requestPermissions(new String[]{Manifest.permission.*POST\_NOTIFICATIONS*},1);  
 // *TODO: Consider calling* // *ActivityCompat#requestPermissions* // here to request the missing permissions, and then overriding  
 //public void onRequestPermissionsResult(int requestCode, String[] permissions)  
 // int[] grantResults)  
 // to handle the case where the user grants the permission. See the documentation  
 // for ActivityCompat#requestPermissions for more details.  
 return;  
 }  
 managerCompat.notify(1, builder.build());  
}

Et toujours depuis onActivityResult on va appeler la méthode getOeuvres()

private void getOeuvres(){  
 HttpUtils connection = new HttpUtils();  
 RequestParams params = new RequestParams();  
  
 params.put("fkutilisateur",1);  
  
 connection.*get*("http://dougoudg.emf-informatique.ch/151\_personal-projet-DOUGOUD-Guillaume/Server/oeuvreManager.php", params, new TextHttpResponseHandler() {  
 @Override  
 public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwable) {  
 label.setText("Error "+statusCode+": "+responseString);  
 }  
  
 @Override  
 public void onSuccess(int statusCode, Header[] headers, String responseString) {  
 String[] rps = responseString.split("<oeuvre><pk\_oeuvre>");  
 Oeuvre oeuvre;  
 ArrayList<Oeuvre> oeuvres = new ArrayList<>();  
 for (int i = 1; i < rps.length; i++) {  
 oeuvre = new Oeuvre(rps[i]);  
 oeuvres.add(oeuvre);  
 }  
 String result = "";  
 for (Oeuvre o : oeuvres) {  
 result += o.getNom()+"\n";  
 }  
 label.setText(result);  
 }  
 });  
}

Dans getOeuvre() on va faire un appel à l’API distante grâce entre autre à la classe HttpUtils mais aussi créer des objets beans de la classe Œuvre pour ensuite les afficher.

package com.example.bibli0tech.rest;  
  
import com.loopj.android.http.AsyncHttpClient;  
import com.loopj.android.http.AsyncHttpResponseHandler;  
import com.loopj.android.http.RequestParams;  
  
public class HttpUtils {  
 private static AsyncHttpClient *client* = new AsyncHttpClient();  
 public static void get(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {  
 *client*.get(url, params, responseHandler);  
 }  
 public static void post(String url, RequestParams params, AsyncHttpResponseHandler responseHandler) {  
 *client*.post(url, params, responseHandler);  
 }  
 public static void put(String url, RequestParams params, AsyncHttpResponseHandler responseHandler){  
 *client*.put(url, params, responseHandler);  
 }  
}

Ceci est la classe HttpUtils qui rend l’utilisation AsyncHttpClient, AsycHttpResponseHandler et RequestParams plus facile.

Et Voici le résultat :

Une image contenant capture d’écran, texte, Rectangle, logiciel

Description générée automatiquement

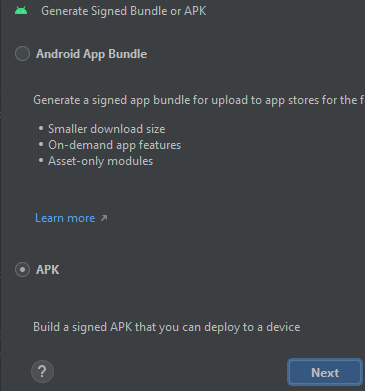
# Test



# Mise en production

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Après avoir créé le fichier APK, on doit le mettre sur le téléphone et télécharger l’application.Une image contenant texte, ligne, Police, Tracé

Description générée automatiquement

# Conclusion