Face Key

Quentin GERARD, Louis L'HARIDON et Matthieu VILAIN
CMI Systèmes Intelligents et Communicants, Université de Cergy-Pontoise
mail@etu.u-cergy.fr

Résumé

abstract of the project

I. Présentation de Face Key

Dans le cadre de notre projet de synthèse de Licence 3 nous avons eu la possibilité

II. Gestion de projet

test

III. Base de Données

IV. Application Android

A. Introduction

Pour le projet FaceKey nous avons besoin d'un grand jeu de données pour entraîner nos modèles de reconnaissance faciale. Pour augmenter ce jeu de données nous eu avons besoin d'un processus plus orienté utilisateur. Nous avons donc décidé de créer une application Android qui va prendre des photos d'un utilisateur et les sauvegarder sur notre Base de Données pour pourvoir entraîner nos modèles. Nous voulons donc donc une application qui propose à l'utilisateur de so connecter à son compte Face Key et lui propose de prendre des photos de lui. Après vérification des photos par l'utilisateur, l'application lui propose de télécharger les photos sur notre Base De Données

B. Activitées

L'application contient 3 activités diférentes.

- Activité de connexion
- Activité de Choix
- Activité de tri des photos

1) Login Activity: Il s'agit de la première activié que nous avons codé. Elle contient 2 text input, une pour le login et une autre pour le mot de passe. En cliquant sur le bouton et en ayant le couple login/mot de passe correct on peut accéder à la seconde activité.

Dans Facekey, nous utilisons postgreSQL pour stocker les informations des utilisateurs. Comme la Base De Données est sur un ordinateur isolé par le réseau de l'Université, nous ne pouvons pas y acceder depuis un téléphone android. Nous allons donc juste stocker dans un simple tableau à deux dimensions des couples login/mdp pour tester l'application.

Si nous pouvions accéder à la BDD depuis le smartphone nous aurions placé un fichier PHP sur un serveur web qui, connecté à la BDD, aurait retourné vérifié via une http request l'exactitude du couple login/mdp testé.

Ici nous utiliserons simplement le tableau à 2 dimensions.

```
public static String[][] login = {{"admin",""},{"Login","p@ssw0rd"},{"louis.lharidon@etu.u-
    cergy.fr", "azerty"}};
```

Et, en cliqaunt sur le bouton soumettre on vérifie si le couple login/mdp correspond à un autre couple présent dans le tableau. present on the array.

```
        for(String[]s : login) {
                if ((name.equals(s[0])) && (pass.equals(s[1]))) {
                // login ok : starting intent
        }
    }
```

La méthode aurait été similaire avec postgreSQL mais, au lieu de créeer un tableau, on aurait vérifier la présence du couple dans la Base De Données via l'httprequest.

2) Choice Activity: Cette activité contient simplement 3 boutons.
— Un qui lancera la caméra
— Un autre qui lancera la troisième activité, celle du tri de photos
— Une dernière qui lancera le service qui uploadera les photos sur la Base de Données MySQL
3) Activité de tri de photos: Cette activité est la plus complexe. Elle est composée de deux éléments : une gridview et une checkbox. La gridview est remplie d'imageviews. Pour faire cela, nous avons du créer un ImageAdapter. Cet adapteur va simplement prendre une image au format bitmap. la redimensionner pour la faire rentrer dans l'imageview et la placer dans le griview tout en gardant sa position et son id en mémoire..

L'imageAdapter calcule les proportions pour que les imageview remplissent chacune de ses view, les stocke dans un tableau et autorise certaines opérations.

```
// Storing views into array
ArrayList<String> itemList = new ArrayList<>();
```

Il y a trois fonctions intéressantes dans l'imageAdapter : decodeSampledBitmapFromUri qui va retourner un bitmap depuis un chemin et un couple hauteur/largeur.

```
Bitmap decodeSampledBitmapFromUri(String path, int reqWidth, int reqHeight) {
            Bitmap bm ;
            final BitmapFactory.Options options = new BitmapFactory.Options();
            options.inJustDecodeBounds = true;
            BitmapFactory.decodeFile(path, options);
            // Calculate inSampleSize
            options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);
            // Decode bitmap with inSampleSize
            options.inJustDecodeBounds = false;
            bm = BitmapFactory.decodeFile(path, options);
            return bm;
        }
```

Une autre qui va calculer le couple hauteur/largeur requis.

```
int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int reqHeight) {
        //  height and width of image
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;
        if (height > reqHeight || width > reqWidth) {
                if (width > height) {
                inSampleSize = Math.round((float)height / (float)reqHeight);
                }
                else {
                        inSampleSize = Math.round((float)width / (float)reqWidth);
                }
        }
        return inSampleSize;
}
```

Enfin nous avons une fonction add qui va ajouter une image à la gridview via son chemin.

```
void add(String path){
        itemList.add(path);
}
```

Nous avons besoin de onCreate pour déclarer les imageview en tant qu'adaptateurs du gridview.

```
final GridView gridview = findViewById(R.id.gridview);
myImageAdapter = new ImageAdapter(this);
gridview.setAdapter(myImageAdapter);
```

Pour ajouter toutes les images depuis un dossier nous avons besoin de lister tous les fichiers d'un dossier. La première étape est l'accès au dossier avec les lignes suivantes :

```
String root = Environment.getExternalStorageDirectory().getAbsolutePath();
File myDir = new File(root + "/saved_images");
```

Ensuite nous pouvons lsiter tous les fichier, y accéder et les ajouter au griview via notre imageAdapter.

```
files = myDir.listFiles();
for(File f : files){
        // add to image adapter the file
        myImageAdapter.add(f.getAbsolutePath());
}
```

Pour gérer les fichiers avant l'envoi sur le serveur nous avons implémenté une suppression des fichiers. L'utilisateur peut être amené à décider si les photos prises sont bonnes à l'envoi. Nous avons placé une checkbox pour qu'il puisse indiquer si il veut supprimer une image et un onclicklistener pour qu'il appuie simplement sur l'image à supprimier.

```
gridview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View v,
                            int position, long id) {
        // if checkcbox is enabled
        if (ch.isChecked()) {
            // getting file from position
            File item = files[position];
            // deleting and checking deleting
            boolean deleted = item.delete();
            Log.d("Files", "Removed " + deleted);
            // reload page to delete from the grid
            reload();
        }
    }
});
```

La fonction reload raffraîchit l'UI en redémarrant l'activité.

```
// reload function, simply reload activity
public void reload(){
    finish();
    startActivity(getIntent());
}
```

C. Intents

Pour cette application nous utilisons une intent externe, celle de la caméra.

1) Camera intent: Cette intent est ouverte depuis l'activité de choix. en utilisant les permissions nous pouvons accéder à la caméra. En ouvrant cette intent, l'application demander la permission à l'utilisateur d'utiliser la caméra et de gérer les fichiers du téléphone et, après acceptation, ouvrira la caméra. L'utilisateur va ensuite prendre une photo, l'intent lui demandera sa confirmation et, ensuite, transfèrera la photo vers l'activité ou elle sera convertie en png et sauvegardée dans un dossier spécifique.

Pour démarrer l'intent de la caméra nous avons donc besoin de déclarer nos permissions (nous avons aussi besoin de permissions pour nos autres activités et intent mais nous expliquerons ici comment gérer les permissions pour utiliser la caméra).

Pour déclarer une activité nous avons tout d'abord besoin d'ajouter la use-permission spécifique dans le manifeste android.

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

Ensuite, dans l'activité ou la permission est requise, nous ajoutons les lignes suivante à la fonction oncreate pour vérifier si la permission est bien accordée et, si ce n'est pas le cas, demander la permission.

```
// CAMERA
 if( (ContextCompat.checkSelfPermission(SecondActivity.this, Manifest.permission.CAMERA)
      != PackageManager.PERMISSION_GRANTED)) {
     if (ActivityCompat.shouldShowRequestPermissionRationale(SecondActivity.this,
         Manifest.permission.CAMERA)) {
         ActivityCompat.requestPermissions(SecondActivity.this, new String[]{Manifest.
             permission.CAMERA}, PERMISSION_REQUEST_CODE);
     } else {
         ActivityCompat.requestPermissions(SecondActivity.this, new String[]{Manifest.
             permission.CAMERA}, PERMISSION_REQUEST_CODE);
     }
 }
```

Ensuite nous lançons l'intent caméra avec les lignes suivantes :

```
Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(cameraIntent, CAMERA_REQUEST);
```

Après que l'utilisateur est pris une photo nous avons besoin de gérer le résultat de l'activité caméra pour stocker l'image dans un dossier. Pour faire cela nous allons utiliser la fonction onActivityResult en vérifiant s i nous obtenons bien le résultat de la caméra et , ensuite, en stockant le bitmap résultant compressé au format png dans le dossier correct avec un nom de fichier unique.

```
if (requestCode == CAMERA_REQUEST && resultCode == this.RESULT_OK) {
    // getting photo bitmat
    Bitmap photo = (Bitmap) data.getExtras().get("data");
    FileOutputStream out = null;
    try {
        // Getting custom picture folder
        String root = Environment.getExternalStorageDirectory().getAbsolutePath();
        File myDir = new File(root + "/saved_images");
        if(!myDir.exists()) myDir.mkdir();
        Log.d("Photos", "directory " + myDir.toString());
        // Writing filename
        String fname = "img-" + System.currentTimeMillis() + ".png";
        // Creating file
        File file = new File(myDir, fname);
        Log.d("Photos", "file " + file.toString());
        try {
            out = new FileOutputStream(file);
            if (photo != null) {
                // putting bitmap photo to file after png compression
                photo.compress(Bitmap.CompressFormat.PNG, 100, out);
            }
            out.flush();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

D. Background service

Dans notre seconde activité nous lançons un service en arrière plan qui va uploader toutes les photos d'un dossier spécifique vers une Base De Données mySQL. Pour faire cela nous avons besoin de lister toutes les images d'un dossier et, pour chacune d'entre elles, les convertir en bitmap et les uploader sur notre Base de Données.

```
String root = Environment.getExternalStorageDirectory().getAbsolutePath();
File myDir = new File(root + "/saved_images");
// Listing files
File[] filelist = myDir.listFiles(IMAGE_FILTER);
// browsing files
int i = 0;
for (File f : filelist) {
    // Sleeping 0.01s between each files
    try {
        Thread.sleep(10);
    } catch (Exception e) {
        Log.e("Error", "exception");
    }
    if (isRunning) {
        Log.d("Files", "FileName:" + f.getName());
        // getting file path
        String filePath = f.getPath();
        // converting to bitmap
        bitmap = BitmapFactory.decodeFile(filePath);
        // starting upload fonction
        uploadImage();
```

Pour l'upload, nous créons une fonction qui convertit les images en base64 et qui lance une httprequest contenant une requête POST vers une page PHP stockées sur un serveur web. Pour cela nous utilisons cette fonction de conversion bitmap vers base64 :

```
// Converting bitmap to base64 function
public String getStringImage(Bitmap bmp){
    // new array
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    bmp.compress(Bitmap.CompressFormat.JPEG, 100, baos);
    byte[] imageBytes = baos.toByteArray();
    // return converted to base64 array
    return Base64.encodeToString(imageBytes, Base64.DEFAULT);
}
```

Et cette fonction qui effectuera la requête POST :

```
public void POST(String... params){
    String urlString = params[0]; // URL
    String data = params[1]; //data  POST
    try {
        // Opening connexion
        URL url = new URL(urlString);
        // URL
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000);
        conn.setConnectTimeout(15000);
        // POST METHOD
        conn.setRequestMethod("POST");
        conn.setDoInput(true);
        conn.setDoOutput(true);
        // Building request  with url and data
        Uri.Builder builder = new Uri.Builder().appendQueryParameter("image", data);
        Log.d("Updload", "POST : " + builder.toString());
        String query = builder.build().getEncodedQuery();
        // outputstream on connexion
        OutputStream os = conn.getOutputStream();
        // Buffered writer to write on outputsteam
```

```
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
            writer.write(query);
            writer.flush();
            writer.close();
            os.close();
            Log.d("Updload", "url : " + conn.getURL().toString());
            // Request
            conn.connect();
        }catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
```

Qui sont toutes deux combinées dans une fonction uploadImage :

```
    //upload image fonction
    private void uploadImage() {
        Log.d("Updload", "uploading ");
        // converting to base64
        String uploadImage = getStringImage(bitmap);
        // Posting to PHP
        POST(UPLOAD_URL, uploadImage);
    }
```

Ce script PHP va prendre le fichier base64 et l'insérer dans une table mySQL sur une colonne de type blob :

```
    $conn = new mysqli(HOST,USER,PASS,DB) ;
    if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
    }
    $image = $_POST['image'];
    $sql = "INSERT INTO photo (image) VALUES ('$image')";
    if ($conn->query($sql) === TRUE) {
        echo "New image successfully inserted";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
    $conn->close();
```

Pour notifier l'utilisateur de l'avancement de la mise ligne des photos nous utiliserons un Toast. Mais, comme nous sommes dans un service, nous ne pouvons pas simplement utiliser un Toast. Nous avons besoin de récupérer le contexte du Thread principal pour y placer le Toast.

```
    private Context appContext;
    // Function to toast on main thread
    void showToast(final String toShow){
        // checking app context
        if(null !=appContext){
            // geting main theard
            Handler handler = new Handler(Looper.getMainLooper());
            handler.post(new Runnable() {
                @Override
                public void run() {
                    // threading
                    Toast.makeText(appContext, toShow, Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
```

Ensuite, via cette fonction, nous allons simplement faire apparaître un Toast sur l'écran de l'utilisateur :

```
String toToast = "Uploading" + (i+1) + "/" + filelist.length + "...";
showToast(toToast);
```

Après avoir téléchargé une image sur la BDD nous la retirons du dossier en la supprimant : After having uploaded an image we remove it from the folder by deletion :

```
boolean del = f.delete();
```

Après avoir téléchargé toutes les images sur la BDD, le service s'arrête.

E. Conclusion

En conclusion, nous avons développé une application robuste qui effectue différentes actions : une activité de connexion, un service qui effectue une httprequest vers un script php, une activité qui affiche les images d'un dossier et permet de les supprimer et nous avons utilisé la caméra via une intent. Pour améliorer cette application nous aurions pu développer notre propre activité pour prendre des photos, ce qui nous aurait permis de prendre en rafale les photos de l'utilisateur. Nous aurions pu améliorer l'UI de l'activité de tri de photos. Le principal défaut de cette application est qu'elle n'est pas connectée aux autres parties de notre projet comme nous n'avons pas la possibilité de télécharger directement les images vers notre base de données. En effet le réseau de l'Université bloque les accès extérieurs aux machines et empêche les machines sur le même réseau de s'accéder. Cette mesure de sécurité nous empêche d'intégrer pleinement notre application android dans notre écosystème et nos données FaceKey

V. Conclusion

Acknowledgement