Developing Application

2017/2018 - University of Cergy-Pontoise

FaceKey Projet

Gerard Quentin - L'Haridon Louis - Vilain Matthieu
quentin.gerard2@gmail.com - louislharidon56@gmail.com - mattvilain@gmail.com

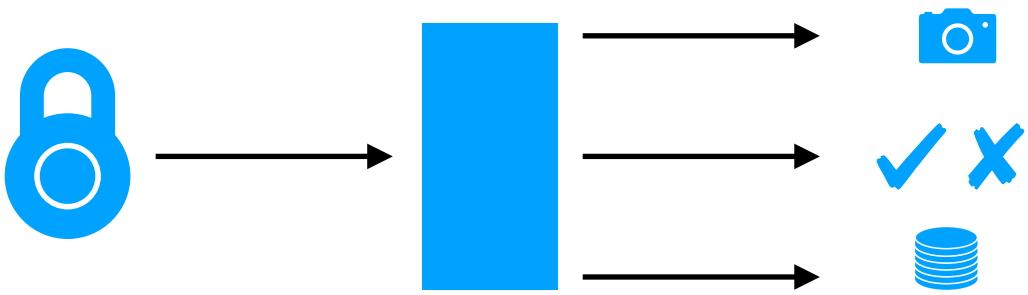
Context

This application is developed for the developing application course of the Cergy-Pontoise University teached by Dimitrios Kotzinos and Emmanouil Katsomallos. It is made for the project of this course and it also take part of the global project witch regroup all the project of the semester. For this project we choose to made a password manager witch use face recognition named Face Key. To do this we need a lot of picture of our users to teach our model. To make this process more user-friendly we decide to made an android application witch take picture of an user and save it on our database to train our model. So the application will first propose to the user to login, second it will propose him to choose if he want to take picture or look the picture he took. And, after having checked his picture, the use can upload his picture to our server.

Description of the application

The application is simple, in a first scree we can login, which conduct us to a second screen where we will be able to choose three options: '

- Taking a picture
- Selecting and deleting pictures
- Uploading pictures to mysql database

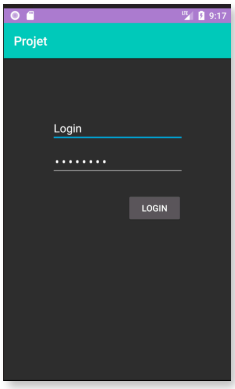


Activities

The app contains 3 different activities.

- Login Activity
- Choice Activity
- Sort Pictures Activity

Login Activity



This is the first activity we made. It contains 2 text input, one for the login and the other one for the password. By clicking on the button with the right log/password combinaison we can access to the second activity.

In FaceKey we use postgreSQL to store the login information of our users. As the database is in a university computer server we can't access it, so we are just going to check on a two dimensional array that contains login/password datas.

If we could have access to database from mobile phone we would have placed a php file on a web server, which, connected to the database would have returned login/ password data from our http request.

Here we just create the array

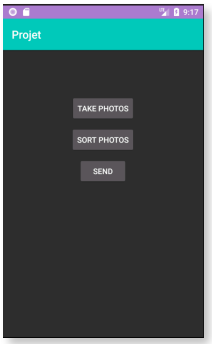
```
public static String[][] login = {{"admin",""},{"Login","p@ssw0rd"},{"louis.lharidon@etu.u-cergy.fr", "azerty"}};
```

and, while clicking on the submit button we check if the textedit couple login and password correspond to another couple present on the array.

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

It would have been the same method with postgreSQL method but, instead of creating an array, we would have get it from our database via httprequest.

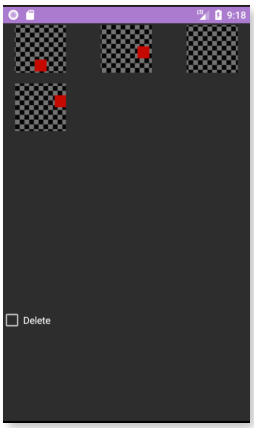
Choice Activity



This activity contains three buttons :

- one will launch the camera
- another one which launch our third activity, the sort picture one.
- one which launch the service that upload picture to mysql database

Sort pictures Activity



This activity is the most complex one.

It is composed of two elements, a gridview and a checkbox. The grid view is filled with imageviews. To do this we had to create an ImageAdapter.
This adapter will basically take a bimap image, calculate his dimension to make it fill in a view of the gridview and will put it into, keeping its id and position in memory.

The image adapter calculate proportions for imageviews to fill into views, store them into an array and allow some operations :

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

There are three interesting functions in the ImageAdapter, decodeSampledBitmapFromUri which will return a bitmap from a file path and a required height and width

```
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;
    }
```

Another one to calculate the required height and width

```
    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;
}
```

And an add function that will add an image to the grid view by passing file path.

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

We need in our onCreate to declare the ImageView as the gridview adapter

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

To add image from a folder to the grid view we need to list all files in a folder. The first step is getting our folder with the following lines :

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

Then we can list all files, browse them and add them to the grid view with our image adapter

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

To deal with all these file before upload we implemented file deletion. User need to be able to delete pictures that are not ok for him to upload. So we add an checkbox to allow him indicate he wants to delete file and an onclicklistener on the element he wants to delete.

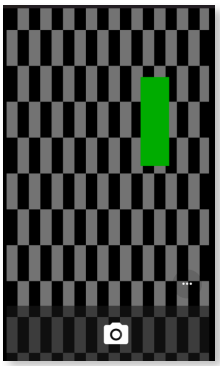
```
    gridview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View v,
                                int position, long id) {
            // if checkcbox is enabled
            if (ch.isChecked()) {
                // 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();
            }
        }
    });
```

reload function make an UI refresh by restarting the activity.

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

Intents

In the app we use an external intent, the camera one.



Camera intent

This intent is opened from the choice activity. By using permission, we can access the camera. Opening the intent will ask for user's camera application and open in. The user will take a photo, intent will ask him a confirmation and, then, forward the photo to the activity where the photo will be converted to png and saved into a specific folder.

To start the camera intent we need to declare permissions (**we also need permissions in our others activities and intent but we'll explain here how we deal with them taking the camera one**) .

To declare an activity we first need to add the use-permission in android manifest.

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

Then, in the activity where the permission is needed, we add the following line to the oncreate function to check if permission is granted and, if it is not, to ask for 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);
         }
     }
```

Then we can start the camera intent with the following lines :

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

After the user had took a picture we need to deal with the result of the camera activity to store the picture in our folder.
To do this we will use onActivityResult function by checking if we get camera result and, then, store compress the bitmap result and store it to the right directory with an unique filename.

```
    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();
            }
```

Background service

In our second activity we can launch a service that will upload all the photo in the specific folder to a mysql database.
To do this we will list all the pictures in the folder and, for each of them, convert them to bitmap and upload thew to our database.

```
            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();
```

To upload them we created a function that convert them in base64 them and launch an http request with contain a post request to a php page on a server.
To do this we use this bitmap to base64 conversion function

```
// 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);
}
```

and this Post to PHP one:

```
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());
    }
}
```

Which are both combined in the uploadImage function.

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

This php script will take the base base64 file and insert it into a mysql table in a blob type column.

```
                $conn = new mysqli(HOST,USER,PASS,DB) ;
                if ($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();
```

To notify the user of the uploading avancement we use Toast. But, as we are in a service, we can't simply use Toast, we have to get the main thread context to put toast in it :

```
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();
            }
        });
    }
}
```

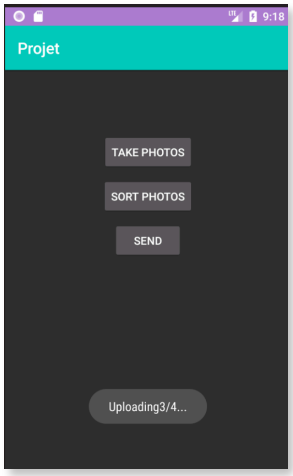
Then, with this function, we can simply make a toast to the user screen :

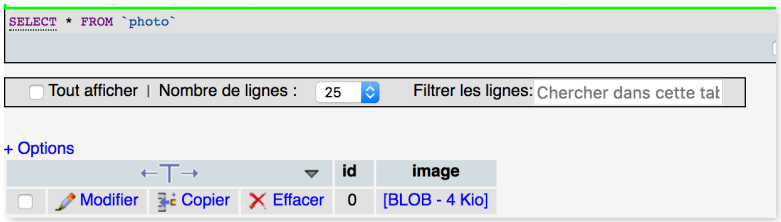
```
                String toToast = "Uploading" + (i+1) + "/" + filelist.length + "...";
                showToast(toToast);
```

After having uploaded an image we remove it from the folder by deletion :

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

After having uploaded all the files in the folder the service will simply stop.





Conclusion

In conclusion we have developed a robust application with a lot of different actions, a login activity, a service that make a httprequest to a php script, a displaying and deleting pictures from a folder activity and the use of a camera threw an intent.
To improve the application we could make our own camera taking activity, which would have allowed us to take burst pictures, and a more beautiful sorting activity.
The major minus of the application is that it's not really connected to the other parts of our projets as we can't upload image directly to the database.
Indeed we can't access a computer in the university network from the exterior and machine can't address themselves into the university network. This security functionality disallow us to make an android app really integrated to our FaceKey datas and ecosystem.