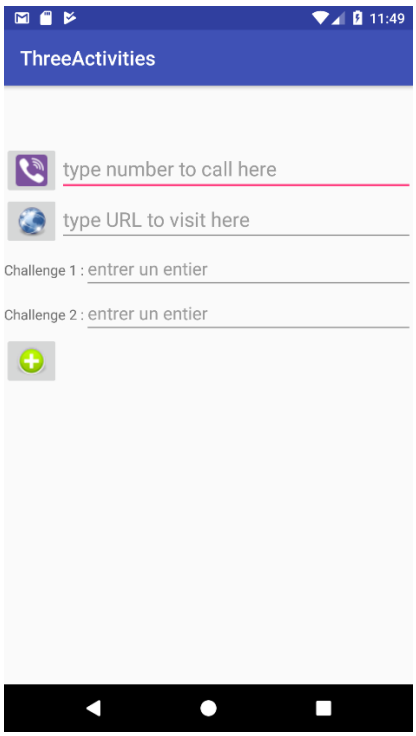
TP 2 Android
TI et IQL
S5 - 2021

Ce TP est le complément du TP 1. Il se base sur le code du TP 1

I- Suite TP 1 : Plusieurs activités

a- Modifier l'activité principale du TP1 pour ajouter deux TextView et deux EditText en-dessous de l'ImageBoutton (Internet). Les deux EditText doivent avoir la propriété suivante : `android:inputType="number"`

Le nom du premier TextView = challenge 1 et le nom du 2ème TextView = challenge 2



b- Modifier le comportement de l'image bouton 2 (Internet) de telle sorte à ce que lorsqu'on clique dessus une autre activité **Check** est lancée (objectif : protéger la navigation internet via un challenge).

Check contient 4 TextView, un EditText et deux boutons : OK, Cancel

TextView 1 : "Challenge 1 reçu : "

TextView 2 : "Challenge 2 reçu : " et les deux autres disposés devant TextView1 et TextView2 respectivement. Ils sont vides initialement mais seront remplis à l'exécution.

Dans l'activité principale il faudra désormais remplir les deux champs challenges avant de cliquer sur l'image bouton Internet. Lorsqu'on clique sur le bouton Internet, l'activité **Check** est appelée en lui transmettant les deux challenges (2 entiers). **Check** affiche les paramètres reçus (dans les TextView 3 et 4 correspondants).

Si on clique sur **Cancel**, on retourne à l'activité principale et rien ne se passe : un Toast est affiché à l'utilisateur pour l'informer que l'opération a été annulée.

Avant de cliquer sur **OK** l'utilisateur doit remplir le EditText avec la valeur de la somme des deux challenges. Quand on clique sur **OK** la valeur de l'EditText est récupérée puis renvoyée à l'activité principale.

L'activité principale vérifie la somme : si elle est correcte elle démarre le site sinon affiche un message d'erreur. Il faudra simuler le cas erroné (fausse somme).

c- Modifier l'activité principale de l'étape précédente pour que le clic sur l'ImageButton (ordi) appelle **implicitement** l'activité **Login**.

Il faut ajouter un Intent-filter personnalisé dans le Manifest. Par exemple :

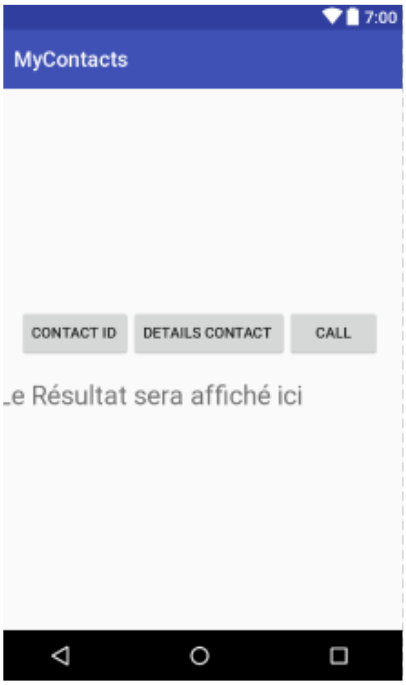
```
<intent-filter>
        <action android:name="login.ACTION"/>
        <category
android:name="android.intent.category.DEFAULT"/>
     </intent-filter>
```

d- Créer une autre activité **AutreLogin** avec le même intent-filter que **Login.**

Exécuter votre application et cliquer sur le bouton ordi. Quelle est l'activité qui démarre ?

Mots clés: Intent.putExtra(), getIntent(), Intent.getExtras(), intent.setAction(), startActivityForResult(), setResult(), onActivityResult(), RESULT_CANCEL, RESULT_OK, Integer.parseInt()

II- Interaction avec l'application contact



Réaliser l'interface suivante :

- Un TextView avec un hint
- 3 boutons : Contact ID, Détails contact et Call

Le clic sur les boutons **détails contact** et **call** est initialement désactivé

Le bouton **Contact ID** permet d'afficher la liste des contacts du téléphone pour en choisir un. (uri= "content://contacts/people" et ACTION_PICK)

Si l'utilisateur annule le choix (ne choisit aucun contact) alors le TextView affiche : opération annulée

Si on choisit un contact, l'application revient à notre interface et affiche dans le TexteView l'**id** du contact choisit sous forme : « content://contacts/people/**n »** avec **n** l'id du contact. (getDataString() ou getData( )).

Le bouton **détails contact** devient activé.

Ne pas oublier de demander la permission : **READ_CONTACTS**

```
        ActivityCompat.requestPermissions(this, new String[]
        {Manifest.permission.READ_CONTACTS}, Perm_CTC);
```

Et redéfinir la méthode correspondante :

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

    //check the permission type using the requestCode
    if (requestCode == Perm_CTC) {
        //the array is empty if not granted
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            //Toast.makeText(this, "GRANTED CALL",
Toast.LENGTH_SHORT).show();
        }

    }
}
```

Nous allons maintenant implémenter le comportement du bouton détails contact :

Après le clic sur le bouton **détails contact** on affichera le nom et le numéro de téléphone du contact. Le bouton Call placera un appel vers le numéro récupéré.

Pour cela :

a. Interroger le base Content provider contacts :

```
Cursor cursor = getContentResolver().query(uriContact, null, null,
null, null);
```

(uriContact = l'uri représentant **content://contacts/people/n** avec n l'id du contact)

b. Utiliser le cursor pour obtenir le nom du contact :

```
cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISP
LAY_NAME));
```

c. Pour Obtenir le numéro de téléphone correspondant il faut d'abord obtenir l'ID du contact et lancer une requête utilisant cet ID :

Pour obtenir l'ID :

```
Id =
cursorID.getString(cursorID.getColumnIndex(ContactsContract.Contacts._ID));
```

Pour Obtenir le numéro :

a. La requête :

```
Cursor cursorPhone =
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_U
RI,
        new String[]{ContactsContract.CommonDataKinds.Phone.NUMBER},

        ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ? AND " +
                ContactsContract.CommonDataKinds.Phone.TYPE + " = " +
                ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE,

        new String[]{contactID},
        null);
```

b. Le numéro :

```
cursorPhone.getString(cursorPhone.getColumnIndex(ContactsContract.CommonDat
aKinds.Phone.NUMBER));
```

Il ne reste qu'à afficher ces informations et au clic du bouton **Call** appeler le contact à partir de votre interface. Le bouton call ne devient activé que lorsqu'on récupère les détails du contact.