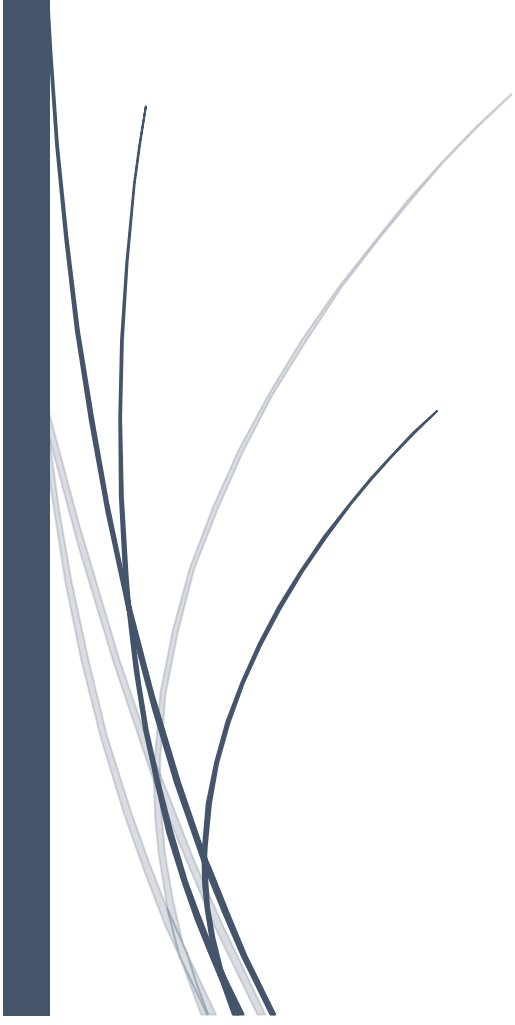
10/04/2016

Introduction à la programmation Android

DAWIN 2017



Stéphane Fossé
IUT INFORMATIQUE DE BORDEAUX

TABLE DES MATIERES

PRESENTATION RAPIDE

Android est un système d'exploitation à destination des dispositifs mobiles (téléphones, tablettes, téléviseurs, montres, lunettes, voitures).

Ses principales caractéristiques sont :

- Open Source[1] (licence Apache), gratuit, flexible, basé sur un noyau linux.
- Inclut les applications de base (téléphone, sms, carnet d'adresse, navigateur, etc.)
- Un ensemble important d'API[2] (OpenGL, media, etc …)
- Un SDK basé sur un sous-ensemble de JAVA[3] (autres langages disponibles : C, C++, …)
- Une machine virtuelle (Dalvik) qui exécute la majorité des applications remplacée par ART depuis la version 5.0 d'Android [4]

Historiquement, Android a été créé en 2005 par la société Android, puis rachetée en 2007 par Google.

Plus de 20 versions depuis la 1.0 (Apple Pie) en 2008 jusqu'à la 7.0 (Nougat) au 08/2016 ont été développées.

Sa part de marché en 2016 est de 80% (18% pour IOS)

LES CONTRAINTES DE LA PROGRAMMATION MOBILE

Il faut toujours penser à ceci lorsqu'on fait du développement mobile :

1 https://source.android.com/

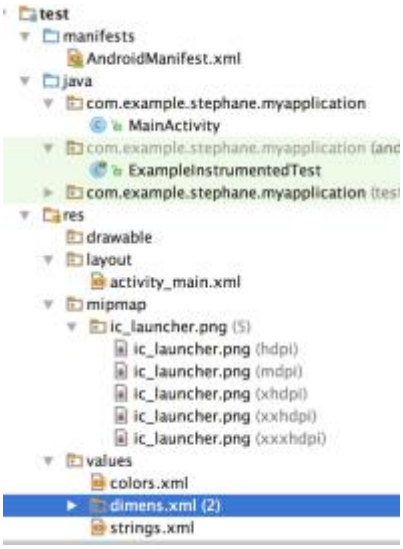
2 https://developer.android.com/reference/packages.html

3 https://developer.android.com/index.html

4 https://source.android.com/devices/tech/dalvik/index.html

- Hétérogénéité du matériel, Processeurs …
- Puissance et mémoire limitées
- Interface tactile
- Taille de l'écran
- Connectivité à internet (disponibilité, rapidité, ...)
- Développement (souvent) extérieur au périphérique

ARCHITECTURE D'UN PROJET SOUS ANDROID STUDIO



- Source Java
- Les ressources
  - Mises en page
  - Menus
  - Icônes
  - Données constantes

LES COMPOSANTS : PRESENTATION GENERALE

- Les activités (Activity)[5]
  Un écran avec une interface utilisateur et un contexte
- Les services (Service)[6]
  Composant sans écran, qui tourne en fond de tâche (lecteur de musique, téléchargement, ...)
- Les fournisseurs de contenu (ContentProvider) [7]
  Entrée/Sortie sur des données gérées par le système ou par une autre application
- Des récepteurs d'intentions (BroadcastReceiver)
  Récupération d'informations générales, arrivée d'un sms, batterie faible, ...

LES INTERACTIONS : PRESENTATION GENERALE

5 https://developer.android.com/guide/components/activities.html

6 https://developer.android.com/guide/components/services.html

7 https://developer.android.com/guide/topics/providers/content-providers.html

- Les intentions (Intent)[8]
  - Permet d'échanger des informations entre composants
  - Démarrage d'un composant en lui envoyant des données
  - Récupération de résultats depuis un composant
  - Recherche d'un composant en fonction d'un type d'action à réaliser
- Les filtres d'intentions (<intent-filter>)
  - Permet à un composant d'indiquer ce qu'il sait faire
  - Permet au système de sélectionner les composants susceptibles de répondre à une demande de savoir-faire d'une application

LE FICHIER ANDROIDMANIFEST.XML

Il s'agit d'une description globale de l'application contenant[9] :

- La liste des composants de l'application.
- Le niveau minimum de l'API requise.
- La liste des caractéristiques physiques nécessaires
  - (gestion de la visibilité sur Google Play)
- La liste des permissions dont l'application a besoin (plus dans la version actuelle)
- La liste des autres API nécessaires (Google Map…)

Ce fichier est généré automatiquement par Android Studio (l'IDE utilisé pour développer sous Android)

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android=http://schemas.android.com/apk/res/android package="fr.stephane.com" >

<application

android:allowBackup="true"

android:icon="@mipmap/ic_launcher"

android:label="@string/app_name"

android:theme="@style/AppTheme" >

<activity

android:name=".MainActivity"

android:label="@string/app_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
```

8 https://developer.android.com/guide/components/intents-filters.html

9 https://developer.android.com/guide/topics/manifest/manifest-intro.html

```
</intent-filter>
</activity>
</application>
</manifest>
```

LES RESSOURCES

Ce sont toutes les données (autres que le code) utilisées par l'application[10].

Rangées dans le dossier **res**, puis incluses dans l'apk lors de la compilation.

- res/drawable et res/mipmap (images en différentes résolutions)
- Layout (description en XML des interfaces)
- Menus (description en XML des menus)
- Values (définitions en XML des constantes utilisées par l'application : chaînes, tableaux, valeurs numériques,

STRING.XML ET INTERNATIONALISATION

Fichier ressources, contenant toutes les chaînes constantes[11]

```
<resources>
<string name="app_name">MyApplication</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
</resources>
```

L'internationalisation est assez simple

Il s'agit de disposer de plusieurs versions des textes, libellés, utilisés par l'application

Le choix sera alors automatique en fonction de la configuration du périphérique

En pratique, il faut :

- Dupliquer le fichier strings.xml : 1 version par langue supportée
- Stocker chaque version dans un dossier spécifique values-xx (ex. values-en, values-fr …)

Ceci est grandement facilité par l'IDE Android Studio (cf TP1)

10 https://developer.android.com/guide/topics/resources/index.html

11 https://developer.android.com/guide/topics/resources/localization.html

LA CLASSE R : UTILISATION DES RESSOURCES EN JAVA

Cette classe est générée automatiquement lors de la compilation de l'application, et permet l'accès aux ressources depuis du code java[12].

Elle contient des classes internes dont les noms correspondent aux différents types de ressources (drawable, layout, …) ainsi que des propriétés permettant de représenter l'ensemble des ressources de l'application

- Utilisation en Java : R. type.identificateur

```
<resources>
<string name="app_name">MyApplication</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
</resources>
```

UTILISATION DES RESSOURCES EN XML

La forme générale d'utilisation des ressources dans les fichiers xml est : **@type/identificateur**

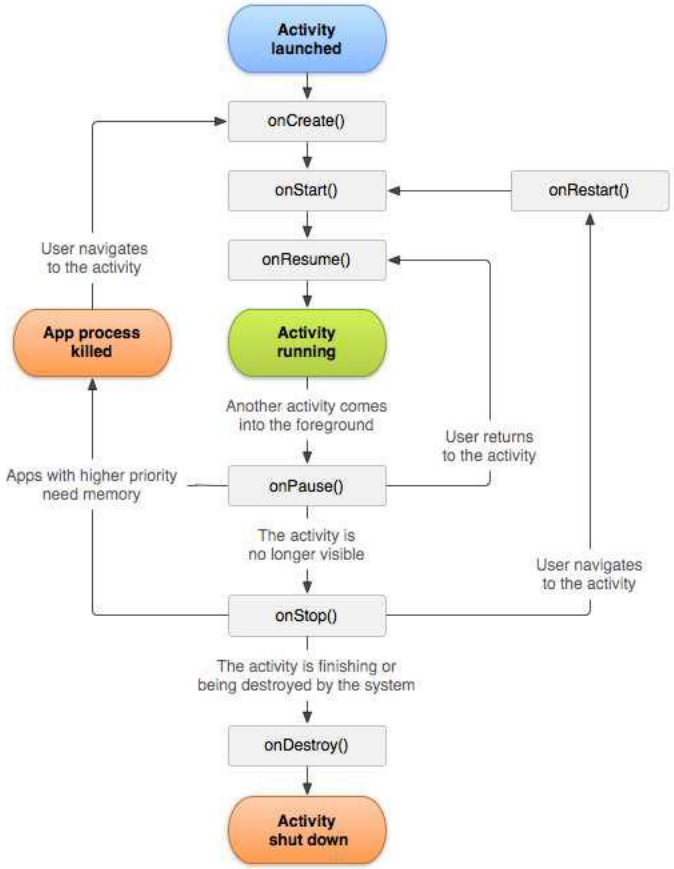
```
<resources>
<string name="app_name">MyApplication</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
</resources>
```

LES ACTIVITES

Une activité est un composant d'une application, doté d'une interface graphique (IHM) et d'un contexte

12 https://developer.android.com/guide/topics/resources/accessing-resources.html

CYCLE DE VIE D'UNE ACTIVITE



Une activité peut se trouver dans différents états en fonction des actions du système et/ou de l'utilisateur :

• Active (resumed) : après un appel à onResume()

• Suspendue (paused) : après un appel à onPause()

• Arrêtée (stopped) : après un appel à onStop()

• Terminée ; après un appel à onDestroy()

DEVELOPPEMENT ANDROID

Les grands principes :

- Une classe java par activité qui hérite de la classe Activity[13] ;
- Les ressources associées dans le répertoire /res(layout, menu, etc.) ;
- Génération d'un code minimum par défaut sous Android Studio.

```
public class Test extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

 super.onCreate(savedInstanceState);
```

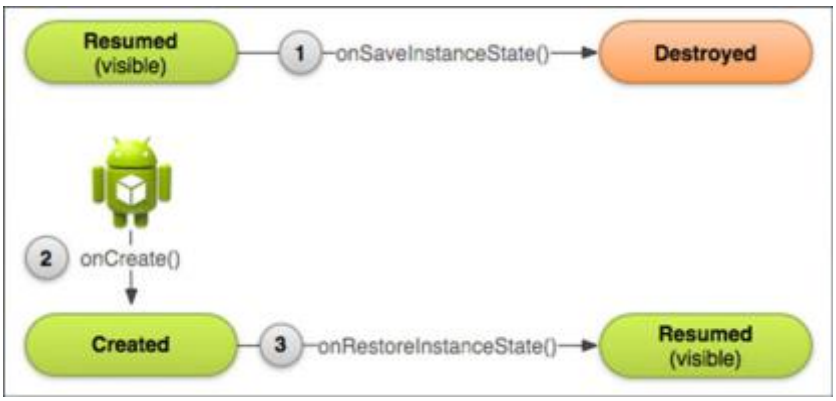
13 https://developer.android.com/reference/android/app/Activity.html

```
  setContentView(R.layout.Bidon);
 }
}
```

D'autres méthodes peuvent être surchargées, en précisant ce qui doit être fait quand :

- protected void onDestroy() : l'activité se termine
- protected void onStart() : l'activité démarre (ou redémarre)
- protected void onPause() : l'activité n'est plus au premier plan
- protected void onResume() : l'activité revient au premier plan
- protected void onStop() : l'activité n'est plus visible
- protected void onRestart() : l'activité redevient visible

L'UTILISATION DU BUNDLE : SAUVEGARDE ET RESTAURATION



- void onSaveInstanceState(Bundle outState)
- void onCreate(Bundle savedInstanceState)
- void onRestoreInstanceState(Bundle savedInstanceState)

DEFINIR UNE INTERFACE GRAPHIQUE

On peut utiliser du XML (façon déclarative) ou Java (façon programmatique) (sauf traitement de l'interaction : Java seul) [14]

Il faut privilégier le XML pour la souplesse de mise à jour et la prise en compte simplifiée de différents types d'écran ; ainsi que pour la facilité de création des interfaces à l'aide d'un outil spécifique.

14 https://developer.android.com/guide/topics/ui/index.html

LES LAYOUTS

Un layout est une zone invisible assurant l'organisation automatique des composants graphiques[15]

Les layouts :

- Peuvent être déclarées en XML ou Java, mais il faut privilégier XML pour :
    - La séparation du code et de la mise en page
    - La souplesse d'adaptation à différents périphériques
- Possèdent des propriétés « intuitives » permettant l'organisation des composants

Il y a de nombreux layouts différents (qui peuvent être imbriqué), de plus :

- Un layout **doit** être spécifié depuis onCreate()
    - setContentView(R.layout.nom_du_layout)
- Pour la gestion multi-écrans,
    - Différentes tailles small, normal, large,xlarge sont possibles.
    - Différentes densités de pixels low (ldpi), medium (mdpi), high (hdpi), extra high (xhdpi) également.
    - Il faut souvent prévoir un layout par taille (et orientation) de l'écran.
- Effets de positionnements relatifs pouvant être gênants
    - Prévoir des images en différentes résolutions
- Fonctionnement similaire à l'internationalisation
    - Un sous-dossier spécifique à chaque layout et/ou à chaque image

RELATIVELAYOUT

C'est le Layout par défaut pour un nouveau projet[16].

Positionnement par rapport au parent ou les uns par rapport aux autres

- match_parent : S'adapte à la taille du conteneur parent (ici l'écran)
- wrap_content : s'adapte à la taille de ce qu'il contient (ici deux zones de texte)
- ou dimension fixe

15 https://developer.android.com/guide/topics/ui/declaring-layout.html

16 https://developer.android.com/guide/topics/ui/layout/relative.html

MainActivity.java × activity_main.xml × strings.xml ×

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView
        android:text="@string/hello_world_01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:text="@string/hello_world_02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</RelativeLayout>
```

Le comportement par défaut est que tous les nœuds sont positionnés à partir du coin supérieur gauche

•Attention à la superposition. !!

LINEARLAYOUT

Aligne les noeuds dans une seule direction horizontale (par défaut) ou verticale[17].

17 https://developer.android.com/guide/topics/ui/layout/linear.html

MainActivity.java × activity_main.xml × strings.xml ×

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity"
    >
```

On peut modifier le «poids » de chaque nœud, et ainsi, changer la taille de la zone occupée par chaque noeud dans l'écran. Il faut ajouter un attribut android: layout_weight à chaque nœud.

```
<TextView
    android:text="@string/hello_world_01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />

<TextView
    android:text="@string/hello_world_02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="2"
    />
<TextView
    android:text="@string/hello_world_03"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    />
<TextView
    android:text="@string/hello_world_04"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="2"
    />
<TextView
    android:text="@string/hello_world_05"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
```

Pour l'alignement de chaque noeud dans sa zone, on peut ajouter un attribut android:layout_gravity. De nombreuses valeurs sont possibles (center, center_vertical, center_horizontal, left, right, top, bottom …)

LES WIDGETS

Composants graphiques visibles par l'utilisateur qui héritent de la classe View[18].

- Widgets simples : zones de texte, boutons, listes, etc.
- Widgets plus complexes : horloges, barres de progression, etc.

Utilisation :

- Définition en XML (type, taille, centrage, position, etc.)
- Comportement en Java
- Peuvent également être créés dynamiquement en Java

LES TEXTVIEW

18 https://developer.android.com/reference/android/view/View.html

Widget permettant l'affichage d'un texte normalement non éditable

Exemple :

```
<TextView
android:id="@+id/text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/letexte"
android:hint="texte initial"
android:layout_gravity="center"
android:gravity="center"
/>
```

Les edittext, les buttons ….. à découvrir lors des TP

IMPLEMENTATION D'UN COMPORTEMENT

Les fichiers XML ne permettent que de :

- positionner les composants ;
- définir leurs caractéristiques.

Il faut donc maintenant définir leur comportement et type d'interaction (clic court, clic long, etc.).

Pour lier composant et code Java, on dispose du :

- XML : attribut android:onClick
- Java : instanciation d'un event listener

ATTRIBUT ANDROID:ONCLICK

```
public void nomDeLaMethode(View maVue)
<Button
android:id="@+id/monBouton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/monTexte"
android:onClick="onBoutonClique"
/>
public void onBoutonClique(View maVue) {
```

```
System.out.println("le bouton avec l'id maVue.getid() a été cliqué") ;
}
```

En général, maVue.getId() permet de récupérer des informations sur le composant graphique qui a généré l'événement

LES EVENT LISTENER

- Ce sont des interfaces de la classe View, ne disposant que d'une seule méthode à implémenter Celle-ci est appelée quand le composant associé est déclenché par l'utilisateur

Exemple : l'interface View.onClickListener

```
Button button = (Button) findViewById(R.id.button_name);
button.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
// Do something in response to button click
}
});
```

ou alors implémenter directement l'interface pour la classe activité.

LES INTENTIONS

Un « intent » est une classe représentant un message échangé entre une activité et

- Une autre activité
- Un service
- Un diffuseur d'événements.

Deux types de messages

- Explicite : on nomme le composant à démarrer
- Implicite : on demande au système de trouver un composant adéquat, en fonction d'une action à effectuer. L'utilisateur devra choisir.

LES INTENTIONS EXPLICITES

Il s'agit d'un message adressé à un composant connu. On doit ainsi spécifier le nom de la classe correspondante.

On utilise généralement ce type d'appel aux composants appartenant à la même application …

On utilise la méthode **startActivity(intent)**

LES INTENTION IMPLICITES

Il s'agit d'un message à destination d'un composant inconnu

- Le système se charge de trouver le composant adéquat et le lance
- En cas de possibilités multiples, il affiche les choix à l'utilisateur

```
Intent = new Intent() ;

Intent.setClassName(this,  SecondActivity.class) ;

startActivity(intent) ;
```

```
Intent intention = new Intent (…) ;

startService(intention) ;
```

Récupération

Coté récepteur (dans la méthode onCreate() de l'activité appelée)

```
public void onCreate (Bundle savedInstanceState) {

Intent intention = getIntent () ;

}
```

Intention avec résultats

```
void maFonction(...) {

Intent intention = new Intent (…) ;

startActivityForResult(Intent intention, int requestCode) ; }

protected void onActivityResult (int requestCode, int resultCode, Intent data){

…}
```

requestcode : Code créé par le développeur pour identifier de manière unique son intention (>0)

```
public static final int CODE = 4 ;
```

Code de retour d'exécution de l'intent

```
Activity.RESULT_OK

Actvity.RESULT_CANCELED
```

Données transmises en retour à l'activité appelante

Intentions avec résultats (récepteur)

```
void maFonction(...) {

Intent intention = new Intent (…) ;

startActivityForResult(Intent intention, int requestCode) ;
```

```
}

protected void onActivityResult (int requestCode, int resultCode, Intent data){

}
```

```
public void onCreate (Bundle savedInstanceState)

{

// récupérer l'intent

Intent intention = getIntent () ;

// extraire les données et les traiter

// créer l'intent résultat

Intent resultat = new Intent();

// ajout des résultats

setResult(RESULT_OK, resultat);

finish();

}
```

Il y a de nombreuses constantes prédéfinies dans la classe Intent :

- ACTION_MAIN : démarre l'activité comme point d'entrée principal
- ACTION_VIEW : affiche les données fournies à l'utilisateur
- ACTION_EDIT : permet l'édition des données fournies par l'utilisateur
- ACTION_SEND : permet l'envoi des données (mail, réseau social, ...)

LES DONNEES

Formatées à l'aide des URI (Uniform Ressource Identifier)

- Chaîne de caractères représentant un endroit ou une ressource
- Syntaxe normalisée :

<Schéma> : <information> [? <requêtes>][# <fragment>]

Nature de l'information (schéma) : http, https, content, geo, file tel , voicemail, sms,smsto, mms, mmsto

L'information dépend du schéma

tel:06000007 geo:123.456,12.3456 http://www.google.fr

```
Intent intention = new Intent() ;

Uri sms = Uri.parse("sms:06000007") ;
```

```
Intention = intention.setData(sms) ;
```

LES EXTRAS

Les extras permettent d'insérer des données dans l'intention

La structure des extra est : **Intent.putExtra(String cle, type valeur) ;**

Pour récupérer l'extra : type get{type}Extra(String cle, type vdefaut) ;

```
public class mainActivity extends Activity {

public final static String MARQUE = " MARQUE" ;

public final static String PUISS =  "PUISS" ;

…I

intent intention = new Intent(this, otherActivity.class) ;

String marque = "Citroen" ;

int puissance = 6 ;

intention.putExtra(mainActivity.MARQUE, marque) ;

intention.putExtra(mainActivity.PUISS, puissance) ;

startActivity(intention) ;

…}
```

```
public class otherActivity extends Activity{

…I

Intent intention = getIntent() ;

int p = intention.getIntExtra(mainActivity.PUISSANCE, 0) ;

String m = intention.getStringExtra(mainActivity.MARQUE) ;

…}
```

LES INTENT FILTERS

Le système doit connaître les types d'intents que peut recevoir chaque composant. Il est donc nécessaire de préciser pour chaque composant les intents possibles.

Les intent filters sont précisés dans le fichier manifest.xml pour chaque composant d'un application.

Leur syntaxe est :

```
<composant ...>
```

```
<intent-filter>
<action android:name="..." />
<category android:name="..." />
<data android:mimeType="..." />
</intent-filter>
</composant>
```

Composant principal d'une application doit apparaître dans la liste des applications pouvant être lancées :

```
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

LES PERMISSIONS

Il s'agit de restreindre ce qu'une application a le droit de faire, de protéger les données présentes sur le périphérique[19].

Par défaut, impossibilité d'impacter une autre application, le système ou les données utilisateur car chaque application est lancée dans son propre espace (sandbox)

Il est donc nécessaire de préciser les permissions demandant une validation par l'utilisateur.

On a deux groupes de permissions

- Normales, sans risque pour l'utilisateur, qui sont attribuées directement par l'application
- Dangereuses (lire les contacts…) qui nécessitent une validation de l'utilisateur.

Les permission se définissent dans le fichier manifest.xml en utilisant la balise <use-permission>

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="fr.SF " >
<uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

À noter que depuis la version 6.0, il est possible de gérer les permissions de façon programmatique à l'exécution de façon individuelle, plutôt qu'à l'installation de façon complète (l'utilisateur peut refuser certaines permissions et faire tourner l'application sans que toutes les permissions ne soient acceptées)[20].

19 https://developer.android.com/guide/topics/security/permissions.html

20 https://developer.android.com/training/permissions/requesting.html

LES MENUS

MENU D'UNE APPLICATION[21]

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/file" android:title="@string/file" >
      <!-- "file" submenu -->
      <menu>
         <item android:id="@+id/create_new" android:title="@string/create_new" />
         <item android:id="@+id/open"  android:title="@string/open" />
      </menu>
   </item>
</menu>
```

et dans votre activité :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.game_menu, menu);
   return true;
}
```

MENU CONTEXTUEL

1. Enregistrez le View auquel le menu contextuel devrait être associé en appelant registerForContextMenu() et transmettre le View. Dans la méthode oncreate
2. Implémenter la méthode onContextItemSelected ;

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
   switch (item.getItemId()) {
      case R.id.edit:
         editNote(info.id);
         return true;
      case R.id.delete:
         deleteNote(info.id);
         return true;
      default:
         return super.onContextItemSelected(item);
   }
}
```

Cf https://developer.android.com/guide/topics/ui/menus.html

21 https://developer.android.com/guide/topics/ui/menus.html