

Par NIDHAL JELASSI
jelassi.nidhal@gmail.com

DÉVELOPPEMENT MOBILE

PROG. ANDROID

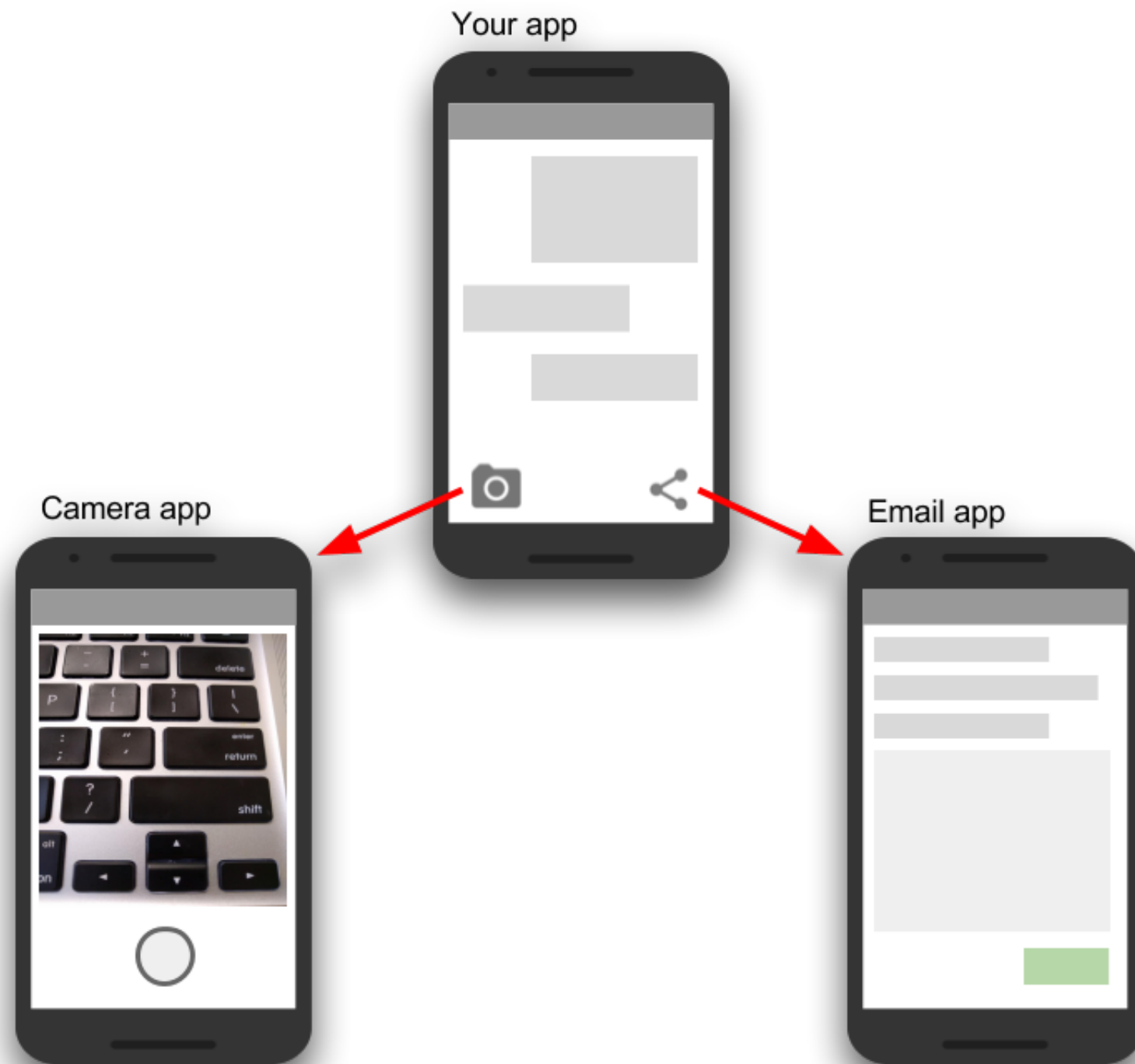


Chapitre 3 : Intents

INTERACTION ENTRE ACTIVITÉS

- ▶ Une application Android peut contenir plusieurs activités :
 - ▶ Chaque **activité** utilise la méthode `setContentView` pour s'associer avec une **interface graphique**.
 - ▶ Les activités sont **indépendantes** les unes des autres, cependant, elles peuvent collaborer pour échanger des données et des actions.
- ▶ Typiquement, l'une des activités est désignée comme étant la première à être présentée à l'utilisateur quand l'application est lancée : on l'appelle l'activité de démarrage
 - ▶ Les activités interagissent en mode **asynchrone**.
 - ▶ Le passage d'une activité à une autre est réalisé en demandant à l'activité en cours d'exécuter un **Intent**.

EXAMPLE



QU'EST CE QU'UN INTENT ?



WHAT IS AN INTENT

QU'EST CE QU'UN INTENT ?

- ▶ Un **intent** est un message qui peut être utilisé pour demander une action à partir d'un autre composant de l'application.
- ▶ Permet invoquer des Activités, des Broadcast Receivers ou des Services. Les différentes méthodes utilisées pour appeler ces composantes sont :
- ▶ `startActivity(intent)` : lance une activité
- ▶ `sendBroadcast(intent)` : envoie un intent à tous les composants Broadcast Receivers intéressés
- ▶ `startService(intent)` ou `bindService(intent, ...)` : communiquent avec un service en arrière plan.

PROPRIÉTÉS

- ▶ Un intent comporte des informations qu'Android utilise
 - ▶ **Nom du composant à démarrer**
 - ▶ **Action à réaliser** : ACTION-VIEW, ACTION_SEND...
 - ▶ **Data** : URI référençant la donnée sur laquelle l'action va agir
 - ▶ **Catégorie** : Information sur le type de composants qui va gérer l'intent (CATEGORY-BROWSABLE, CATEGORY-LAUNCHER...)
 - ▶ **Extras** : Paires clef-valeur qui comportent des informations additionnelles pour réaliser l'action demandée
 - ▶ **Drapeaux** (Flags) : Définissent la classe qui fonctionne comme métadonnée pour cet intent

TYPES D'ENTENT

Il existe deux types d'intents :

- ▶ **Intents Explicites**

- ▶ Spécifient le composant à démarrer par nom (nom complet de la classe).
- ▶ Permettent de démarrer un composant de votre propre application, car le nom de la classe est connu

- ▶ **Intents Implicites**

- ▶ Ne nomment pas un composant spécifique, mais déclarent une action à réaliser.
- ▶ Permet à un composant d'une application d'appeler un composant d'une autre application.

INTENT EXPLICITE

- Démarrer un intent explicite :
 - Le lancement d'une nouvelle Activity de façon explicite s'effectue en deux lignes.

1. `Intent i = new Intent(ActivityA.this, ActivityTarget.class);`

Activité courante



```
graph TD; A[Activité courante] --> B[ActivityA.this]; C[Activité appelée] --> D[ActivityTarget.class];
```

Activité appelée

2. `StartActivity(i)`

INTENT EXPLICITE

Les principaux arguments d'un Intent explicite sont :

- ▶ Le **contexte** déclenchant l'intent, soit :
 - ▶ This, si on le lance à partir de l'activité de départ,
 - ▶ `<Activity_class_name>.this`, sinon
 - ▶ La classe destination : `<Activity_class_name>.class`
- ▶ Il est donc appelé comme suit:
 - ▶ `Intent myActivityIntent = new Intent (StartClass.this, EndClass.class) ;`
 - ▶ `startActivity (myActivityIntent) ;`


INTENT EXPLICITE

- ▶ L'activité démarrée reste à l'écran jusqu'à ce que l'utilisateur appuie sur le bouton Précédent (**Back**) de l'appareil.
- ▶ Cette activité est alors fermée (détruite) et reprise par le système. L'activité d'origine revient au premier plan.
- ▶ Vous pouvez également fermer **manuellement** l'activité démarrée en réponse à une action de l'utilisateur (comme un clic sur un bouton, par exemple) avec la méthode `finish()`.


LES EXTRAS

- ▶ Un **extra** est un couple **clé/valeur**, basé sur le système **Bundle** :
- ▶ La clé est l'identifiant, on peut y mettre ce qu'on veut sous la forme d'une chaîne de caractères.
- ▶ La **valeur** est celle de la donnée. Elle sera associée à la **clé**.
- ▶ Ainsi, quand on crée un extra, on lui donne une clé et une valeur. En revanche, à la récupération de notre extra, c'est à travers la clé qu'on obtiendra la valeur associée.

LES EXTRAS : EXEMPLE




```
Intent i = new Intent(this, SecondActivity.class);  
i.putExtra("zoneTexte", chaine);  
i.putExtra("nbreDeMots", mots);  
startActivity(i);
```




```
Intent i = new Intent(this, SecondActivity.class);  
Bundle bundle = new Bundle();  
bundle.putString("zoneTexte « , chaine);  
bundle.putInt (« nbreDeMots", mots);  
i.putExtras(bundle);  
startActivity(i);
```

LES EXTRAS : EXEMPLE.. SUITE



```
Intent i = getIntent();  
if (i.hasExtra("zoneTexte")) {  
    String str = i.getStringExtra("zoneTexte");  
}  
int nbMots = i.getIntExtra("nbreDeMots", 0);
```



```
Intent i = getIntent();  
Bundle bundle = i.getExtras();  
str = bundle.getString("zoneTexte");
```

LES DATAS

- ▶ L'envoi des données se fait à travers la méthode `setData()` de la classe `Intent`.
- ▶ `setData()` prend comme arguments un objet `URI`, qui représente l'emplacement de la donnée qu'on compte passer à l'intent.
- ▶ Un **URI** (Unified Resource Identifier) peut être un `Url` (`http://`), un numéro de téléphone (`tel://`), un emplacement géographique (`geo://`).
etc.

LES DATAS : EXEMPLE



```
Intent i = new Intent(this, SecondActivity.class);

// adresse URL
i.setData(Uri.parse("http://www.google.com"));

// un fichier
i.setData(Uri.fromFile(new File("/sdcard/sample.jpg")));

// un contenu data
i.setData(Uri.parse("content://mysample.provider/data"));

// type personnalisé
i.setData(Uri.parse("custom:" + dataID + buttonId));
```

- La méthode statique `Uri.parse` permet de construire un objet URI à partir d'une chaîne de caractères.

DATA ET EXTRA

En résumé donc :

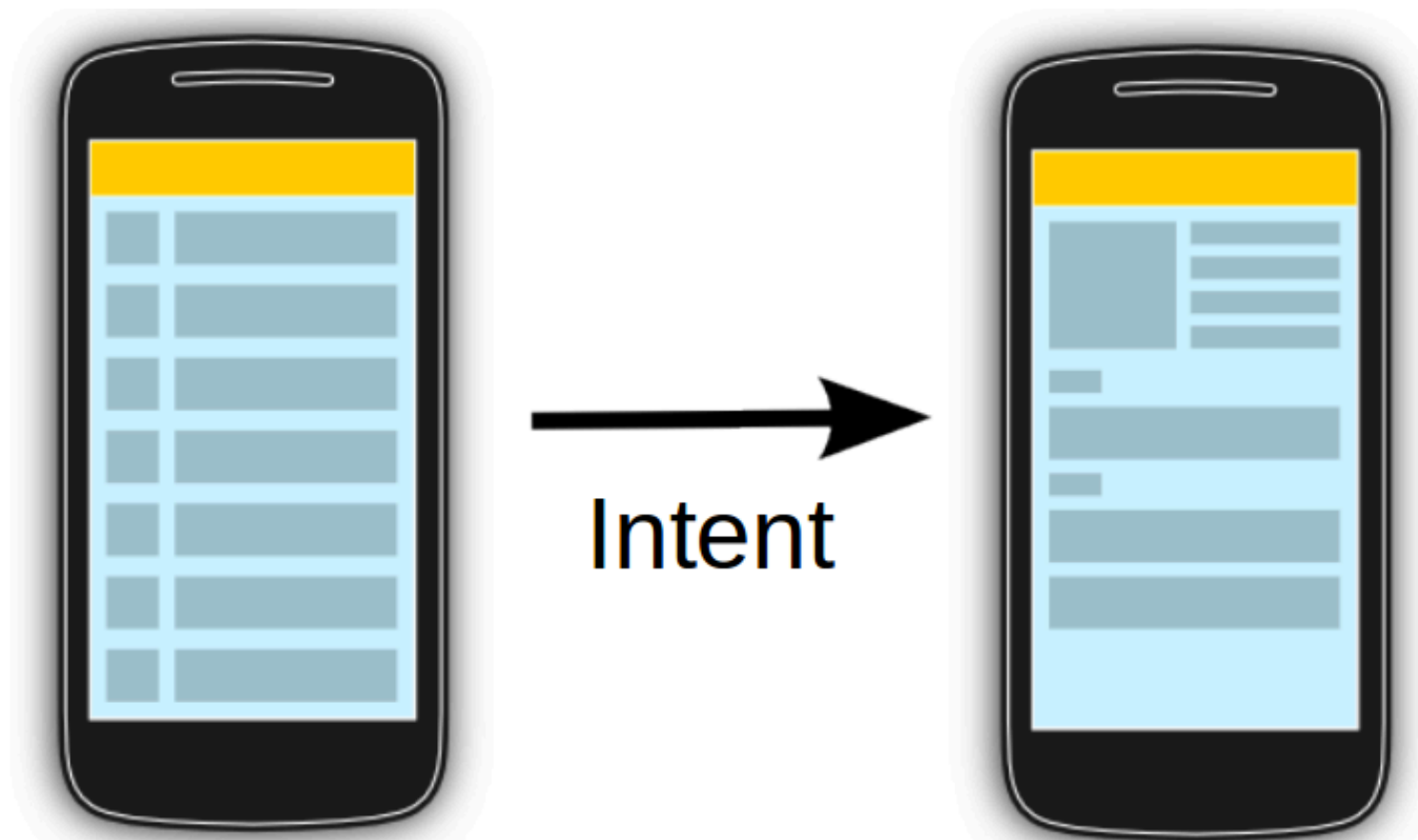
- ▶ dans la *première activité* (d'envoi), il faut :
 - ▶ Créer l'objet d'Intent.
 - ▶ Mettre des données ou des extras dans cet **Intent**.
 - ▶ Commencer la nouvelle activité avec `startActivity()`.
- ▶ Dans la *deuxième activité* (réception), il faut :
 - ▶ Obtenir l'objet d'Intent avec lequel l'activité a été démarrée.
 - ▶ Récupérer les données ou les extras de l'objet Intent.

DATA VS EXTRA

- On opte pour les intent Data quand :
 - ▶ Nous n'allons envoyer qu'une seule information à l'activité cible.
 - ▶ L'information envoyée peut être représenté par un URI
- On opte pour les intent Extras quand :
 - ▶ On désire envoyer plusieurs informations (de différents types en général) à l'activité cible.
 - ▶ Une de ces informations ne peut pas être envoyé à travers un URI.

DATA VS EXTRA

- ▶ Les **datas** des Intents et les **extras** ne sont pas exclusifs.
- ▶ Il est possible ainsi d'utiliser des données pour un URI et des extras pour toute information supplémentaire dont l'activité démarrée a besoin pour traiter les données dans cet URI.




STARTACTIVITYFORRESULT


- ▶ Pour récupérer un résultat à partir d'une autre activité, il est possible d'établir un intent « **bidirectionnel** » entre deux activités.
 - ▶ Pour ça, il faut invoquer `startActivityResult` au lieu de `StartActivity`
- ▶ Evidemment, l'activité cible doit renvoyer un résultat une fois l'opération réalisée.
- ▶ Le résultat est envoyé sous forme d'Intent.
- ▶ L'activité principale le recevra dans la méthode `onActivityResult()`.

INTENT IMPLICITE

- ▶ Démarrer un intent implicite :
- ▶ Exemple :
 - ▶ `String requete = "http://www.google.fr/search?q=Mobile"`
 - ▶ `Intent I = new Intent(Intent.ACTION_VIEW, Uri.parse(requete));`
 - ▶ `startActivity(i);`



Ne définit pas une application en particulier,
Android va tenter de chercher une
application s'étant définie comme capable de
répondre à l'action ACTION_VIEW



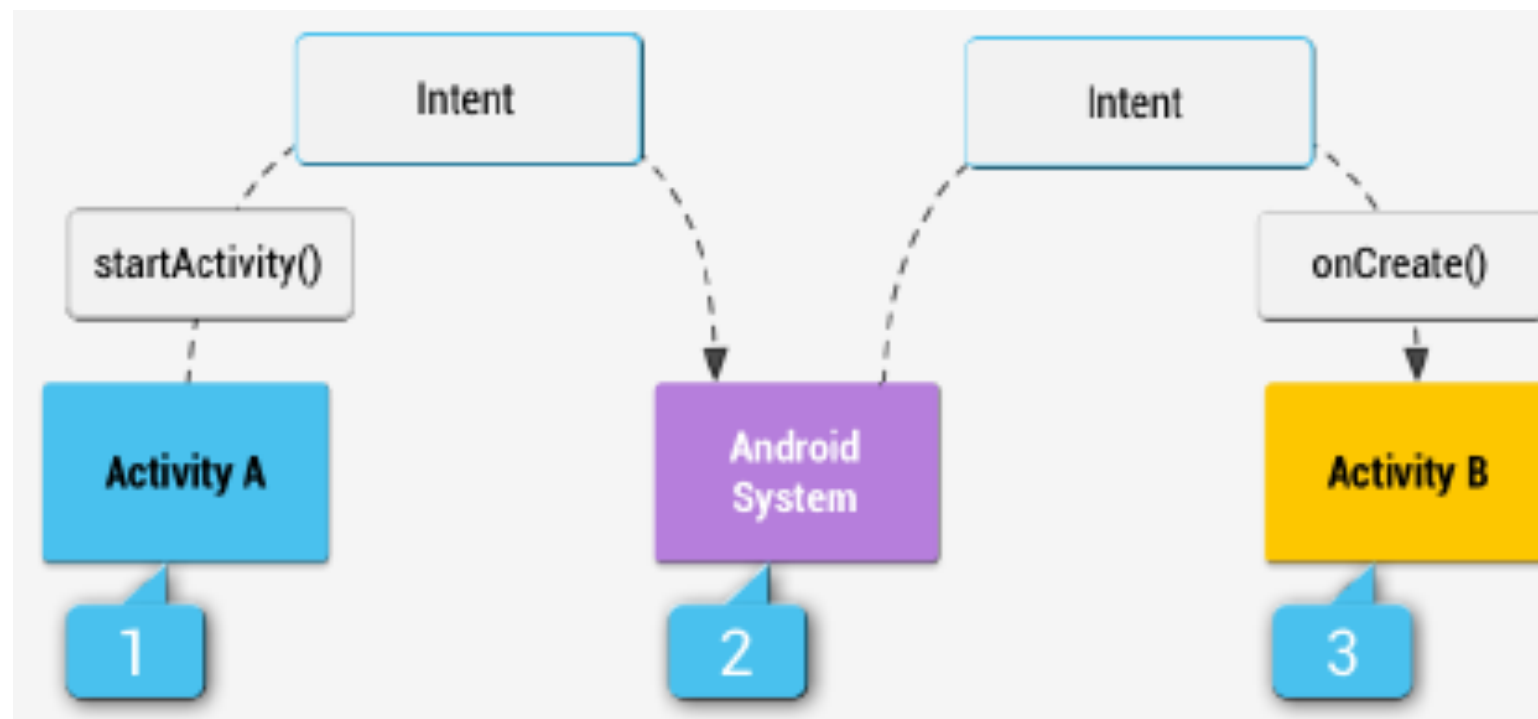
ACTION_VIEW : action définie par le
framework qui consiste à démarrer un
navigateur web sur l'Uri donnée

INTENT IMPLICITE

- ▶ Les principaux arguments d'un Intent implicite sont :
 - ▶ **Action** : l'action à réaliser, peut être prédéfinie (ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.) ou créée par l'utilisateur.
 - ▶ **Données** : Les données principales sur lesquelles on va agir, tel qu'un url ou un numéro de téléphone à appeler.
- ▶ Il est donc typiquement appelé comme suit:
 - `Intent myActivityIntent = new Intent (<action>, <données>);`
 - `startActivity (myActivityIntent);`

INTENT IMPLICITE

- ▶ Un intent implicite se comporte comme suit:
 - ▶ **Activité A** crée un Intent avec une action et le passe en paramètre à `startActivity`
 - ▶ Le système Android cherchent toutes les applications pour trouver un **Intent Filter** qui correspond à cet Intent
 - ▶ Quand une correspondance est trouvée, le système démarrent l'activité (**Activity B**) en invoquant `onCreate` et en lui passant l'intent.



ACTIONS PRÉDÉFINIES

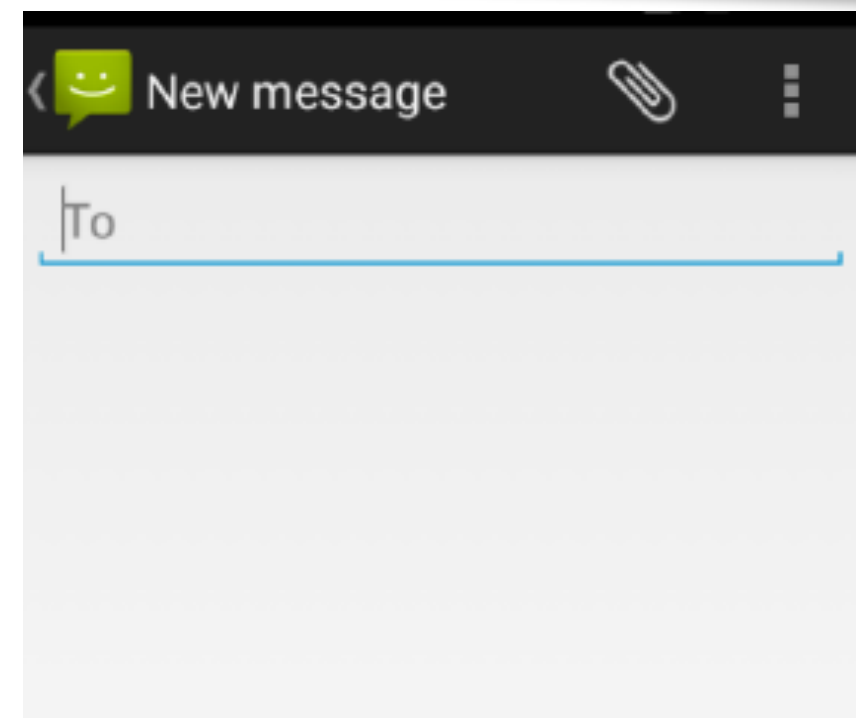
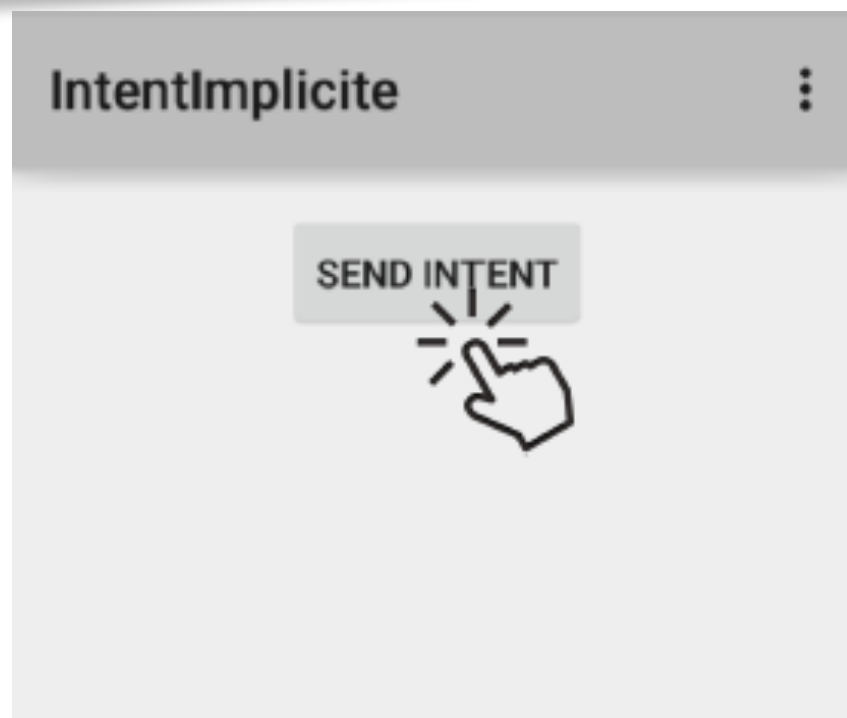
- Exemples d'actions prédéfinies communément utilisées

Action	Donnée	Description
ACTION_DIAL	tel:123	Affiche le numéroteur téléphonique avec le numéro (123) rempli
ACTION_VIEW	http://www.google.com	Affiche la page Google dans un navigateur.
ACTION_EDIT	content://contacts/people/2	Edite les informations sur la personne dont l'identifiant est 2 (de votre carnet d'adresse)
ACTION_VIEW	content://contacts/people/2	Utilisé pour démarrer une activité qui affiche les données du contact numéro 2
ACTION_VIEW	content://contacts/people	Affiche la liste des contacts, que l'utilisateur peut parcourir. La sélection d'un contact permet de visualiser ses détails dans un nouvel Intent.

EXEMPLE

```
public void send(View v){  
    // Create the text message with a string  
    Intent sendIntent = new Intent();  
    sendIntent.setAction(Intent.ACTION_SEND);  
    sendIntent.putExtra(Intent.EXTRA_TEXT, "Hello");  
    sendIntent.setType("text/plain");  
  
    // Verify that the intent will resolve to an activity  
    if (sendIntent.resolveActivity(getPackageManager()) != null) {  
        startActivity(sendIntent);  
    }else{  
        Toast.makeText(this, "The send action could not be performed!", Toast.LENGTH_SHORT).show();  
    }  
}
```

Eviter que l'application crash si l'activité appelée n'existe pas



INTENT FILTER

- ▶ Un **Intent Filter** est une expression dans le fichier **AndroidManifest** d'une application qui spécifie le type d'intents que le composant veut recevoir.
- ▶ Si vous ne déclarez pas d'Intent Filters à votre activité, elle ne pourra pas être déclenchée par un Intent Implicite.
- ▶ Un composant d'une application doit avoir autant de filtres que de capacités de traitement. S'il peut gérer N types d'intent, il doit avoir N filtres.

INTENT FILTER

- ▶ La correspondance entre un **intent** et un **filtre** se fait selon trois critères :
 - L'action
 - La catégorie
 - Les données

```
<activity
    android:name=".MainActivity"
    android:label="TP3"
    android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <action android:name="android.intent.action.SENDTO" />
        <category android:name="android.intent.category.LAUNCHER" />
        <data android:mimeType="text/plain" android:scheme="http" />
        <data android:mimeType="text/plain" android:scheme="https" />
    </intent-filter>
</activity>
```

INTENT FILTER

- ▶ L'activité principale a toujours, et par défaut, l'action et la catégorie ci-dessous.

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

- ▶ L'action spécifie qu'il s'agit de l'activité principale de notre application.
- ▶ La catégorie spécifie que l'activité est répertoriée dans le System's application launcher.



JELASSI.NIDHAL@GMAIL.COM

NIDHAL JELASSI