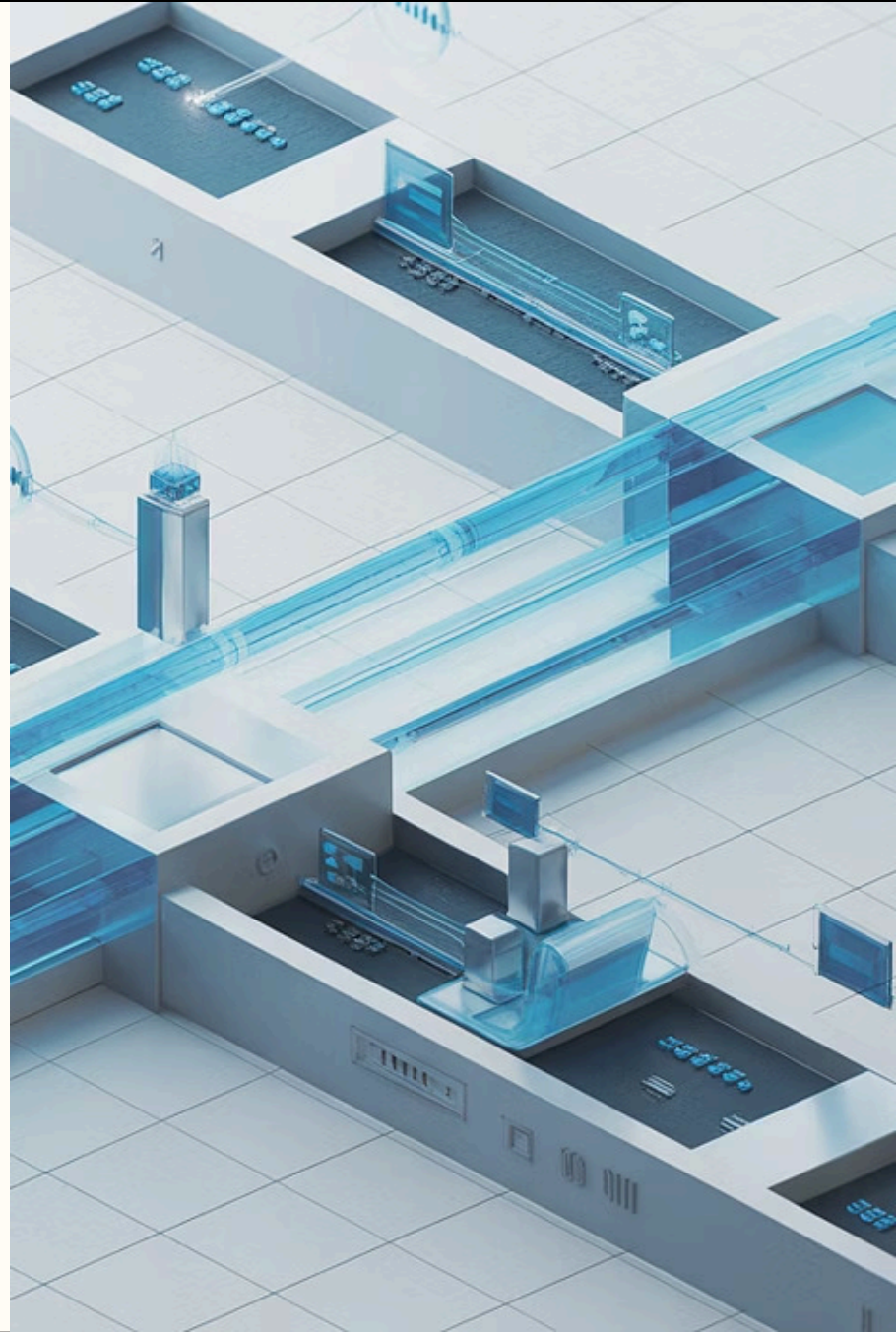
Chapitre 6 — Les Intents et la Communication entre Composants Android



6.1 Introduction aux Intents

Un Intent est un message envoyé à Android pour déclencher une action

Il permet la communication entre :

- les activités d'une même application
- les services
- ou d'autres applications installées
- des composants du système Android (ex. AlarmManager)

C'est un des fondements du système Android

Deux types principaux : Intent explicite et Intent implicite

6.2 Les Intents Explicites

Permettent de lancer une activité précise (classe connue)

```
Intent intent = new Intent(this, DetailActivity.class);
startActivity(intent);
```

- Utilisés pour la navigation interne à l'application
- Chaque activité cible doit être déclarée dans le AndroidManifest.xml
- Il peut contenir d'autres informations (surtout les Extras)

6.3 Les Intents Implicites

Décrivent l'action à accomplir, sans nommer le composant

Android choisit automatiquement une application capable de répondre

Exemple de code:

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://emsi.ma"));
startActivity(intent);
```

Utilisés pour (exemples) :

- ouvrir un lien web
- lancer un appel
- envoyer un message
- partager un fichier ou une image

Composantes d'un Intent

Un Intent est composé de plusieurs éléments :

Action : ce que l'on veut faire

```
intent.setAction(Intent.ACTION_VIEW);
```

Data (URI) : sur quelle ressource

```
intent.setData(Uri.parse("https://emsi.ma"));
```

Type : type MIME du contenu

```
intent.setType("text/plain");
```

Extras : les données supplémentaires

```
intent.putExtra("key", "value");
```

Category : contexte d'utilisation

```
intent.addCategory(Intent.CATEGORY_BROWSABLE);
```

Flags : modifier le comportement de l'Intent

```
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

Exemples d'Actions Prédéfinies

Les Intents implicites utilisent des actions prédéfinies par Android pour interagir avec les fonctionnalités du système ou d'autres applications.

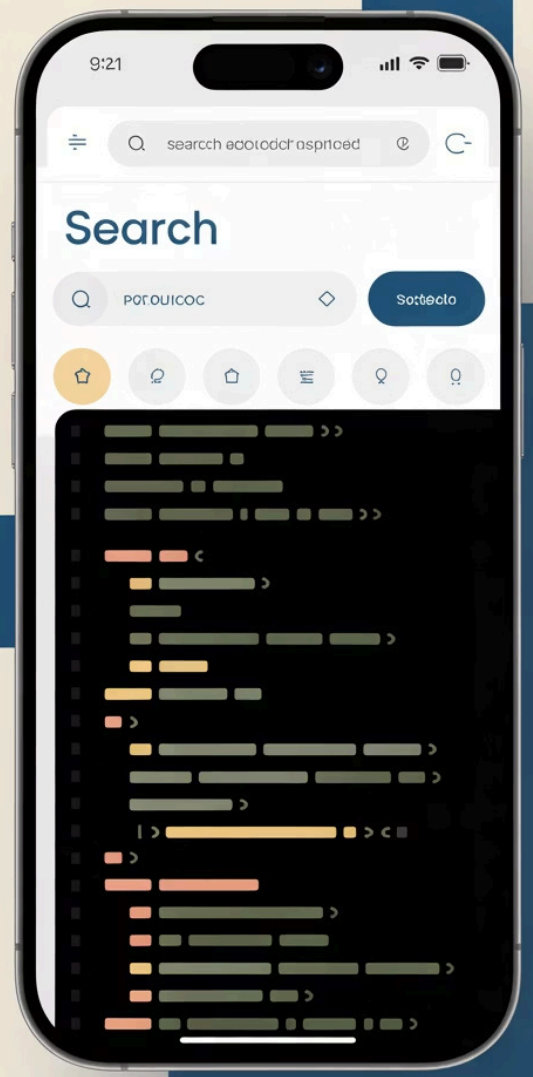
Intent.ACTION_VIEW	Ouvrir un lien, une image ou un contact	ACTION_VIEW, Uri.parse("https://...")
Intent.ACTION_DIAL	Ouvrir le composeur téléphonique	ACTION_DIAL, Uri.parse("tel:...")
Intent.ACTION_SEND	Partager du contenu (email par exemple)	ACTION_SEND avec des extras
Intent.ACTION_PICK	Sélectionner une image ou un contact	ACTION_PICK avec MediaStore.Images.Media.EXTERNAL_CONTENT_URI

Exemple 1 : Ouvrir une Page Web

Vous pouvez utiliser le code suivant pour aller à une page web dans le navigateur par défaut :

```
Uri webpage = Uri.parse("https://www.google.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
startActivity(webIntent);
```

Toujours utiliser "https://" pour les URLs modernes



Exemple 2 : Faire un Appel Téléphonique

Pour ouvrir le composeur téléphonique avec un numéro prérempli :

```
Uri number = Uri.parse("tel:0645127859");
Intent call = new Intent(Intent.ACTION_DIAL, number);
startActivity(call);
```

ACTION_DIAL ouvre le composeur sans lancer l'appel directement (pas besoin de permission)

Exemple 3 : Ouvrir une Carte Géographique

Pour ouvrir une carte indiquant une latitude/longitude donnée :

```
Uri location = Uri.parse("geo:37.425239,-122.0836?z=18");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);
startActivity(mapIntent);
```

Paramètres:

- geo: latitude,longitude
- z= niveau de zoom (1-23)
- Ouvre Google Maps ou toute application de cartographie installée

Exemple 4 : Partager du Texte

Pour partager du texte via d'autres applications (email, SMS, réseaux sociaux) :

```
Intent shareIntent = new Intent(Intent.ACTION_SEND);
shareIntent.setType("text/plain");
shareIntent.putExtra(Intent.EXTRA_SUBJECT, "Sujet du message");
shareIntent.putExtra(Intent.EXTRA_TEXT, "Contenu à partager");
startActivity(Intent.createChooser(shareIntent, "Partager via"));
```

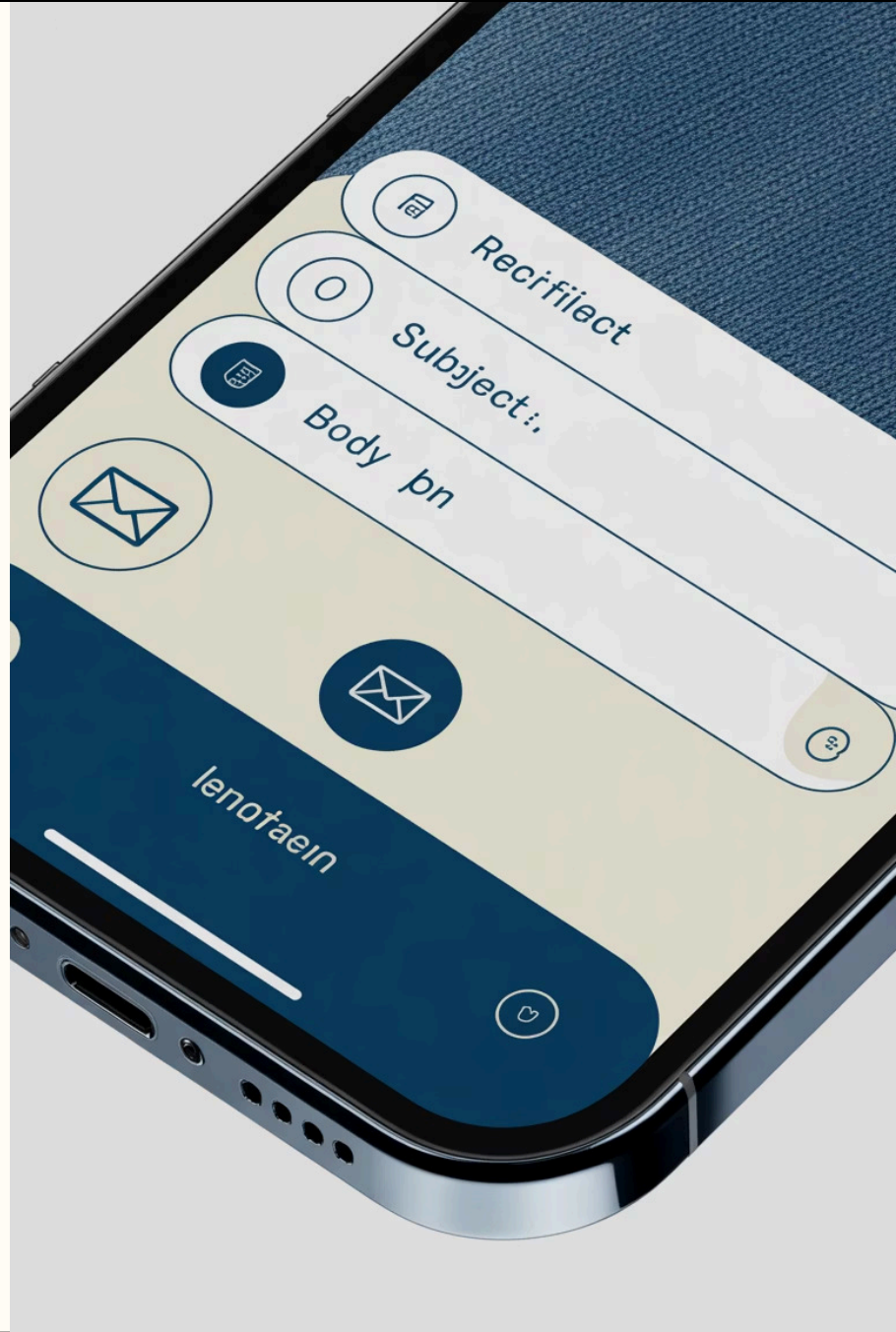
createChooser() affiche toujours le sélecteur d'applications

Exemple 5 : Envoyer un Email

Pour composer un email avec destinataire, sujet et corps préremplis :

```
Intent emailIntent = new Intent(Intent.ACTION_SENDTO);
emailIntent.setData(Uri.parse("mailto:contact@emsi.ma"));
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Demande
d'information");
emailIntent.putExtra(Intent.EXTRA_TEXT, "Bonjour,\n\nJe souhaite...");
startActivity(emailIntent);
```

ACTION_SENDTO avec "mailto:" garantit que seules les applications email répondent



6.4 Passage de Données entre Activités

Les Intents transportent des données via les "Uri" et les "Extras"

Exemple d'envoi d'un Extra de type String:

```
Intent intent = new Intent(this, ProfileActivity.class);
intent.putExtra("username", "Ali");
startActivity(intent);
```

Récupération:

```
String user = getIntent().getStringExtra("username");
```

Pour des objets complexes : implémenter l'interface Parcelable

Transmission d'Objets Complexes

Parcelable est la méthode recommandée pour passer des objets

Exemple d'envoi:

```
intent.putExtra("user", userObject);
```

Récupération dans l'activité cible:

```
User user = getIntent().getParcelableExtra("user");
```

Plus efficace que Serializable sur Android! Il faut implémentet l'interface Parcable (comme Serializable)

6.5 Récupération de Résultats d'une Activity

Historiquement, on utilisait les méthodes `startActivityForResult()` et `onActivityResult()` pour gérer le retour de données d'une activité secondaire. Cette approche est maintenant dépréciée.

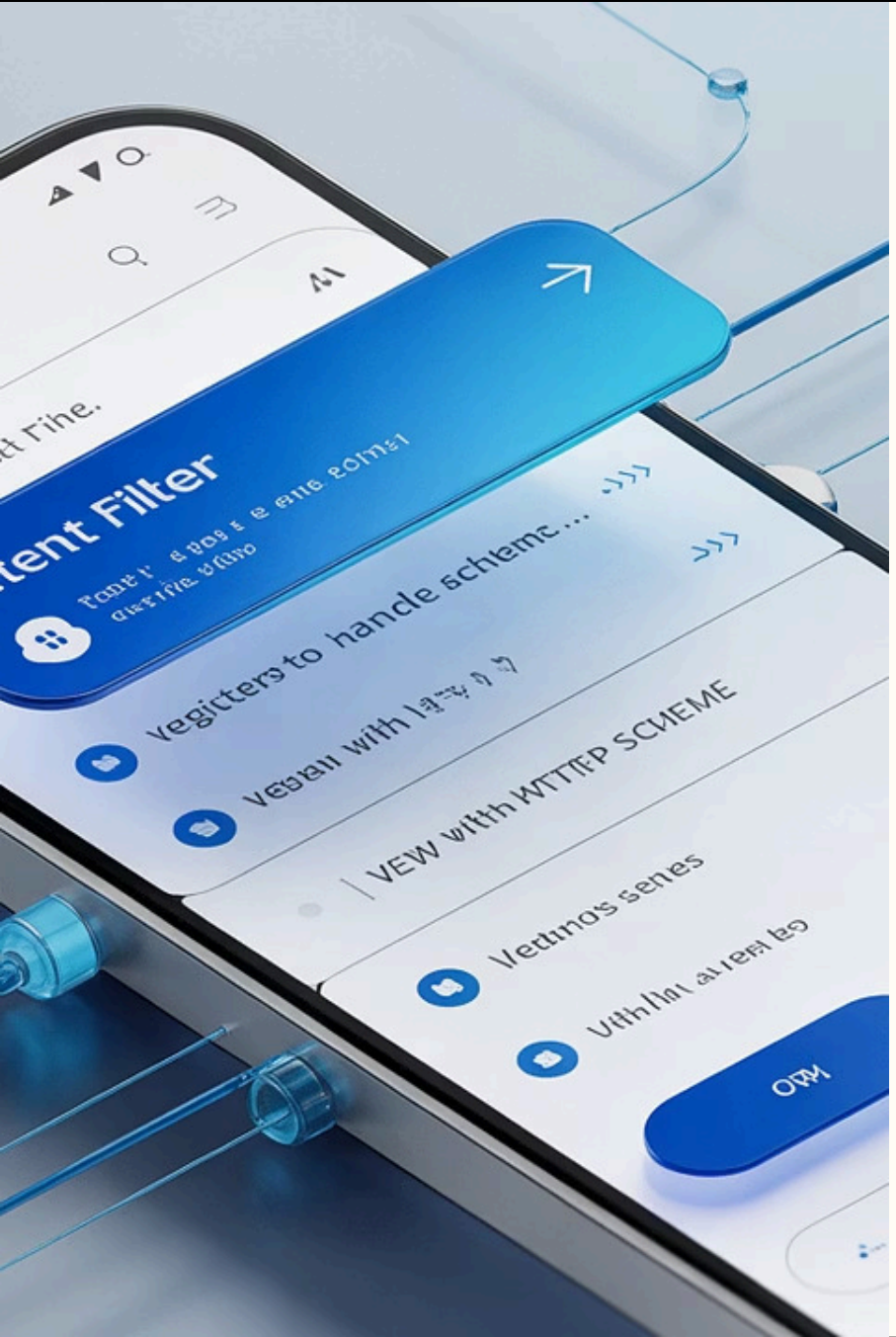
```
startActivityForResult(intent, REQUEST_CODE);

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
        // traitement du résultat
    }
}
```

Nouvelle méthode standard : Activity Result API

La nouvelle méthode standard et recommandée est l'Activity Result API, offrant une approche plus moderne et plus sûre pour la gestion des résultats d'activité.

```
ActivityResultLauncher<Intent> launcher =
    registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> { /* traitement */ });
```



6.6 Filtres d'Intents

Les applications peuvent déclarer des filtres dans le AndroidManifest.xml

Cela permet à Android d'identifier quelles activités peuvent répondre à certaines actions

Exemple:

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="https" />
</intent-filter>
```

6.7 Broadcast et Communication Globale

1 Les Broadcasts

diffusent des messages à plusieurs composants

2 Deux types :

- Système (batterie, réseau, écran, démarrage …)
- Personnalisé (créé par l'application)

Pour envoyer un broadcast personnalisé :

```
Intent intent = new Intent("com.emsi.MON_ACTION_PERSONALISEE");
sendBroadcast(intent);
```

3 Exemple:

```
registerReceiver(new MyReceiver(), new IntentFilter(Intent.ACTION_POWER_CONNECTED));
```

4 Important :

désenregistrer le Receiver (unregisterReceiver())

L’attribut android:exported

Obligatoire depuis Android 12 pour toute activité, service ou receiver possédant un `<intent-filter>`.

Cet attribut indique si le composant (activité, service, broadcast receiver) peut être appelé par d'autres applications installées sur l'appareil.

```
<activity
   android:name=".ShareActivity"
   android:exported="true">
   <intent-filter>
      <action android:name="android.intent.action.SEND" />
      <category android:name="android.intent.category.DEFAULT" />
   </intent-filter>
</activity>
```

Toujours définir `android:exported="false"` par défaut pour les composants internes qui ne sont pas destinés à être accessibles par d'autres applications.

Intent Filters et Actions Personnalisées

Les filtres d'Intents permettent à une application de déclarer spécifiquement les actions personnalisées qu'elle est capable de gérer, ouvrant la porte à des communications inter-applications basées sur ces actions.

Émission d'un Intent personnalisé

```
Intent intent = new
Intent("com.emsi.ACTION_SHOW_MESSAGE");
intent.putExtra("msg", "Bonjour EMSI !");
sendBroadcast(intent);
```

Dans le BroadcastReceiver

```
public void onReceive(Context c, Intent i) {
    Toast.makeText(c, i.getStringExtra("msg"),
Toast.LENGTH_SHORT).show();
}
```

Logique de Réception de l'Intent >

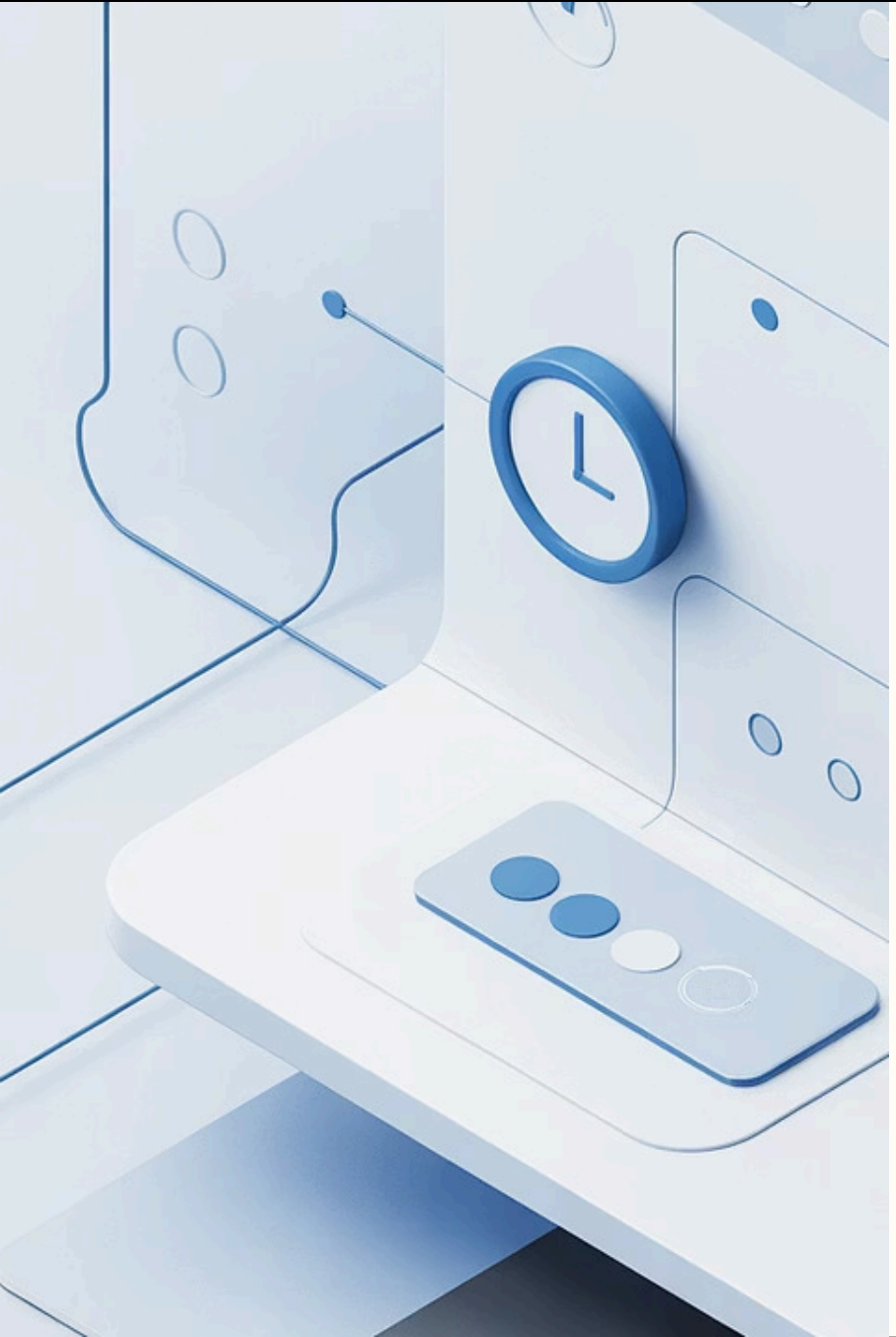
Pour que le message soit reçu, un `BroadcastReceiver` doit être enregistré (soit dynamiquement, soit via le Manifest) avec un `IntentFilter` spécifiant l'action personnalisée "com.emsi.ACTION_SHOW_MESSAGE".

Déclaration d'un Intent Filter Personnalisé

Pour que le `BroadcastReceiver` soit capable de recevoir l'action personnalisée, il doit être déclaré dans le fichier `AndroidManifest.xml` avec un `<intent-filter>` correspondant.

```
<receiver android:name=".MessageReceiver">
    <intent-filter>
        <action android:name="com.emsi.ACTION_SHOW_MESSAGE" />
    </intent-filter>
</receiver>
```

Il est impératif de préfixer toutes les actions personnalisées avec le nom de votre package (par exemple, `com.emsi.MON_ACTION`) afin de garantir leur unicité et d'éviter les conflits avec d'autres applications.



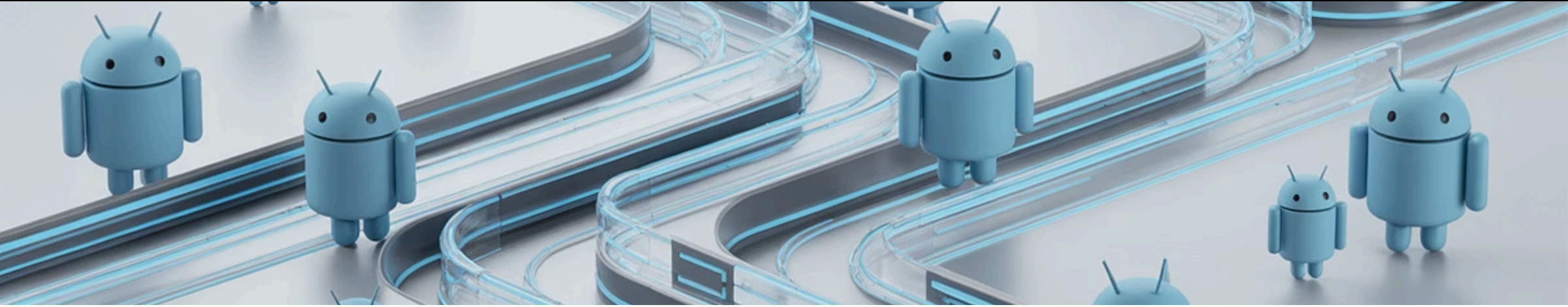
6.8 PendingIntent

Un Intent différé exécuté plus tard, souvent par le système. Utilisé pour :

- Notifications
- Alarmes
- App Widgets

Exemple:

```
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
intent, PendingIntent.FLAG_IMMUTABLE);
```



Communication entre Composants

- la navigation entre activités
- la communication avec les services
- la réception d'événements système

Android garantit la sécurité et l'isolation entre applications grâce au concept des intentions

6.9 Bonnes Pratiques

Toujours vérifier les cibles disponibles :

```
if
(intent.resolveActivity(getPackageManager())
!= null)
    startActivity(intent);
```

Ne pas exposer inutilement des filtres d'intent publics

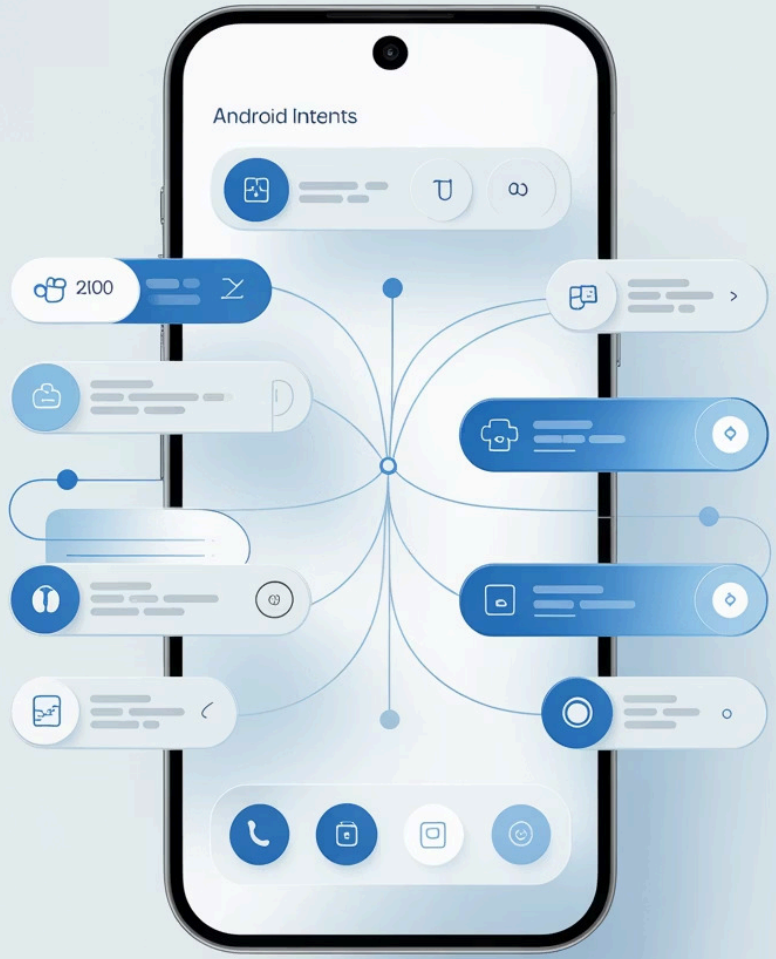
Utiliser des clés constantes pour les extras

Idées pour pratiquer

1. startActivityForResult : Ajouter une Activity formulaire pour modifier une ToDoTask dans votre projet

2. Lors du passage de LoginActivity à ToDoActivity, utilisez un Intent implicite au lieu de l'Intent Explicite

3. PendingIntent : Ajouter des données supplémentaires à vous tasks (Date/Heure ou Localisation GPS) et recevez des alertes selon le déclancheur (LocationManager, AlarmManager …)



Résumé du Chapitre

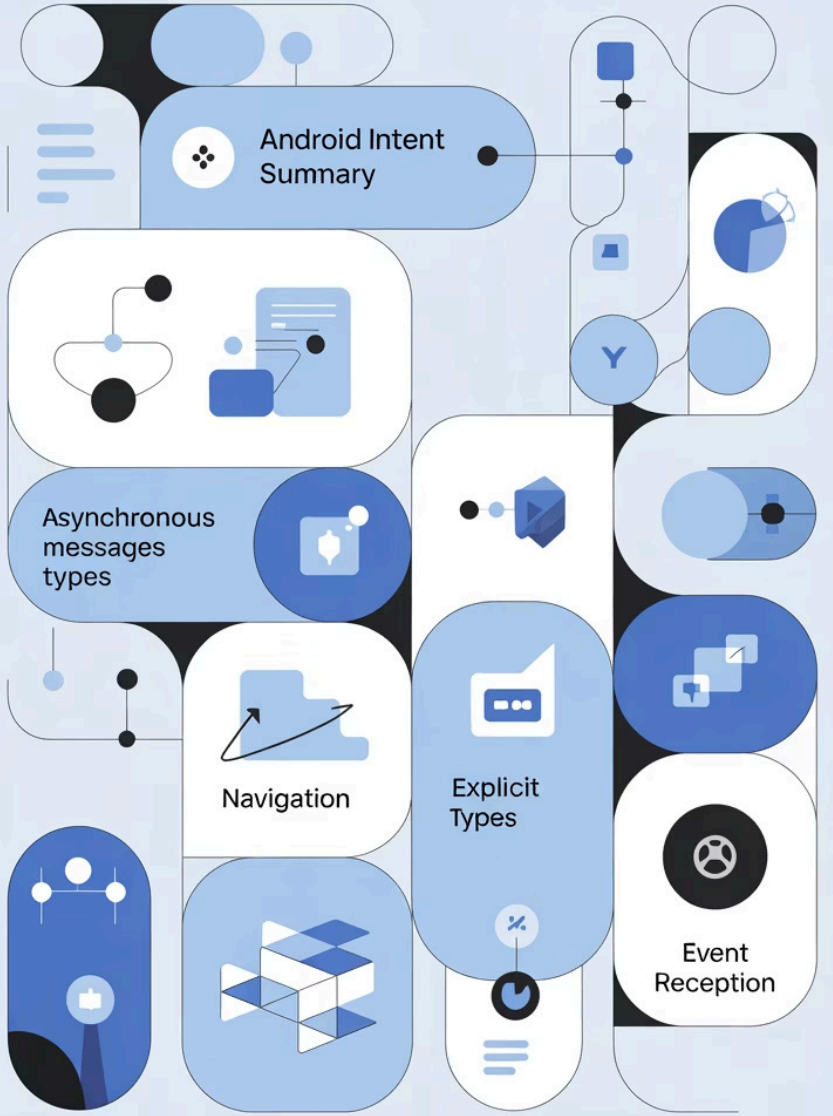
Un Intent est un message asynchrone pour lancer une action

Deux types : Explicite (même app) et Implicite (autres apps)

Permet de :

- naviguer
- partager
- recevoir des événements

Outils modernes : ActivityResultLauncher, FileProvider, PendingIntent



Fin du chapitre 6