Sujet de TP N°3 - Première application Android

VUE D'ENSEMBLE

Dans ce TP, vous allez développer une application Android simple qui vous permettra de manipuler des `Intents` explicites et implicites. Ce projet vous aidera à comprendre comment les activités interagissent entre elles ainsi qu'avec d'autres applications.

OBJECTIFS

- Utiliser les `Intent` explicites pour naviguer entre différentes activités.
- Manipuler les `Intent` implicites pour interagir avec des applications externes (navigateur, email, caméra).
- Apprendre à récupérer des données d'une autre activité avec les `Intent`.

Fonctionnalités à mettre en œuvre :

- Une activité principale avec 4 boutons qui déclenchent chacun un `Intent`.
    - Ouvrir une nouvelle activité.
    - Ouvrir un site web dans le navigateur.
    - Ouvrir une application d'email avec un message pré-rempli.
    - Ouvrir l'application caméra pour prendre une photo.

Étapes du TP

1. Prise en main

- Créez un nouveau projet Android dans Android Studio, en utilisant le template **Empty Views Activity** (Compose sera vu dans un cours ultérieur).
- Nommez votre application **TP_03_name_name**.
- Choisissez le langage Kotlin et le minimum SDK **API 29**.
- Explorez l'interface d'Android Studio et localisez le fichier **AndroidManifest.xml**. Vérifiez que l'activité principale contient bien la catégorie `Launcher` qui permet de la lancer depuis le lanceur d'applications.

- Regardez le code de l'activité principale, qui est une sous-classe d'`Activity`. Notez l'appel à `setContentView(...)` dans la méthode `onCreate()`, qui lie le fichier de layout XML à votre activité.

2. Création de la vue

- Dans le fichier de layout XML associé à votre activité principale (généralement `activity_main.xml`), nous allons ajouter des **boutons**. Un bouton est un composant d'interface qui réagit au toucher de l'utilisateur.
- Voici un exemple de code pour ajouter un bouton dans un `ConstraintLayout` :

```
<com.google.android.material.button.MaterialButton
    android:id="@+id/button_new_activity"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Nouvelle Activité"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp" />
```

- Ce code fait plusieurs choses :
    - Il crée un bouton avec l'ID unique `button_explicit` (ce qui permettra de l'identifier dans le code).
    - Le texte visible sur le bouton sera "Nouvelle activité".
    - Le bouton est positionné en haut de l'écran avec une marge de 16dp (décalage par rapport au haut).
    - `wrap_content` signifie que le bouton prendra uniquement la place nécessaire pour afficher son texte.
- **Explication des contraintes** : dans un `ConstraintLayout`, les vues (comme le bouton) sont positionnées en fonction d'autres vues ou des bords de l'écran, appelés **parents**.
    - `app:layout_constraintTop_toTopOf="parent"` signifie que le haut du bouton est lié au haut du parent (l'écran).
    - `app:layout_constraintStart_toStartOf="parent"` et `app:layout_constraintEnd_toEndOf="parent"` signifient que le bouton est centré horizontalement en le reliant aux deux côtés de l'écran.
- Pour les autres boutons :
    - Placez-les sous le bouton précédent en ajoutant une contrainte sur le **haut** du bouton suivant qui sera liée au **bas** du bouton précédent.
    - Ajoutez des marges pour les espacer suffisamment.

- Vous devrez maintenant ajouter trois autres boutons pour les actions "Navigateur", "Email", et "Photo", en suivant le même principe.
- Pensez à vérifier en déployant votre application sur votre smartphone ou dans un émulateur.

3. Interaction avec les boutons

- Maintenant que vous avez ajouté vos boutons dans le layout, nous allons les rendre interactifs en ajoutant des listeners de clics dans le code Kotlin.

  Étape 1 : Récupérer les références des boutons

  Dans le fichier Kotlin de votre activité principale (généralement `MainActivity.kt`), nous devons **récupérer les références** des boutons que vous avez définis dans le fichier XML.

  Pour cela, dans la méthode `onCreate()`, vous pouvez utiliser la fonction `findViewById()` pour chaque bouton, comme ceci :

```
val newActivityButton = findViewById<MaterialButton>(R.id.button_new_activity)
```

- Ici, nous avons récupéré une référence au bouton avec l'ID `button_new_activity` défini dans le fichier XML. Vous devrez faire la même chose pour les trois autres boutons que vous avez créés.

  Étape 2 : Ajouter un listener de clic

- Une fois la référence du bouton récupérée, nous pouvons ajouter un **listener** pour écouter les événements de clic. Par exemple, pour le bouton "Nouvelle activité", nous ajouterons un listener comme ceci :

```
newActivityButton.setOnClickListener {
    // Ici, vous pouvez écrire le code à exécuter lorsqu'on clique sur le bouton
    Log.d("MainActivity", "Le bouton a été cliqué")
}
```

- Ce code permet de déclencher une action lorsqu'on clique sur le bouton. Dans cet exemple, nous ajoutons un message dans le **Logcat** pour vérifier que le bouton fonctionne correctement.

Étape 3 : Vérification avec le Logcat

- Avant d'aller plus loin, vérifiez que vos boutons fonctionnent correctement en exécutant l'application, en cliquant sur les boutons, et en vérifiant le **Logcat** dans Android Studio pour voir si le message est bien affiché.

4. Utilisation des `Intents`

Étape 1 : Gestion du bouton "Navigateur"
Maintenant, nous allons faire en sorte que le bouton "Navigateur" ouvre un site web lorsque l'utilisateur clique dessus. Pour cela, nous allons utiliser un `Intent` implicite.

1. Dans le **listener** de clic que vous avez ajouté pour le bouton "Navigateur", vous allez devoir créer un nouvel `Intent`.
2. Pour ouvrir un site web, vous devez spécifier l'action que vous voulez effectuer. L'action que nous allons utiliser est `Intent.ACTION_VIEW`.
3. Vous aurez également besoin de fournir une **URI** (Uniform Resource Identifier) qui représente l'adresse du site que vous souhaitez ouvrir.
4. Vous pouvez utiliser une adresse web simple, comme `"https://www.google.com"`, pour tester.
5. Une fois que vous avez créé l'`Intent`, vous devez appeler `startActivity(intent)` pour lancer le navigateur avec l'`Intent` que vous venez de créer.

Étape 2 : Gestion du bouton "Email"
Nous allons maintenant configurer le bouton "Email" pour qu'il ouvre une application de messagerie lorsque l'utilisateur clique dessus.

1. Dans le listener de clic pour le bouton "Email", commencez par créer un nouvel `Intent`.
2. Pour ouvrir une application d'email, vous avez besoin d'une action spécifique. Cherchez dans les constantes d'`Intent` ou sur la documentation pour trouver l'action adéquate.
3. Tout comme pour le navigateur, vous allez avoir besoin d'une URI pour votre Intent.
4. Si vous le souhaitez, vous pouvez ajouter des paramètres via la méthode putExtra pour fournir un sujet et un message d'email.

Étape 3 : Gestion du bouton "Photo"
Nous allons maintenant configurer le bouton "Photo" pour qu'il ouvre la caméra de l'appareil afin que l'utilisateur puisse prendre une photo.

1. Dans le listener de clic pour le bouton "Photo", commencez par créer un nouvel `Intent`.
2. Pour ouvrir la caméra, vous aurez besoin d'une action spécifique. Cherchez dans les constantes d'`Intent` ou sur la documentation pour trouver l'action adéquate pour capturer une image.
3. Dans cette partie, on cherche juste à ouvrir l'application caméra et prendre une photo, on ne cherche pas à récupérer le résultat.

Bonus : Afficher la photo prise

- Dans votre fichier layout XML, ajoutez une `ImageView` pour afficher la photo. Assurez-vous de lui donner un `id` unique.
- Pour récupérer le résultat, nous avons besoin d'utiliser `registerForActivityResult(ActivityResultContracts.StartActivityForResult())`
- Vous pouvez trouver un exemple dans la Documentation sur StartActivityForResult

Étape 4 : Gestion du bouton "Nouvelle activité"
Dans cette étape, vous allez créer une nouvelle activité et configurer le bouton "Nouvelle activité" pour qu'il lance cette activité.

1. Créer une nouvelle activité :
    a. Dans Android Studio, faites un clic droit sur le package de votre application dans l'arborescence de projet.
    b. Sélectionnez **New** > **Activity** > **Empty Activity**.
    c. Nommez votre activité (par exemple, `SecondActivity`) et assurez-vous que l'option pour ajouter au fichier `AndroidManifest.xml` est cochée.
2. Gérer le bouton
    a. Dans le listener de clic pour le bouton, créez un nouvel `Intent`.
    b. Utilisez le constructeur de `Intent` qui prend deux paramètres : le contexte (par exemple, `this`) et la classe de la nouvelle activité (par exemple, `SecondActivity::class.java`).
    c. Ensuite, appelez `startActivity(intent)` pour lancer la nouvelle activité.