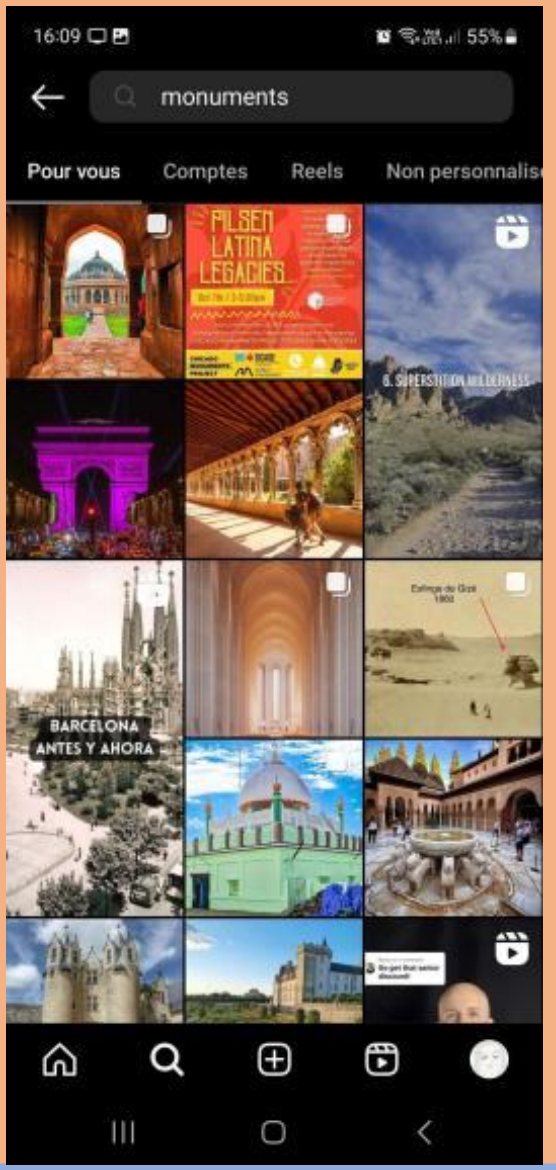
Troisième cours

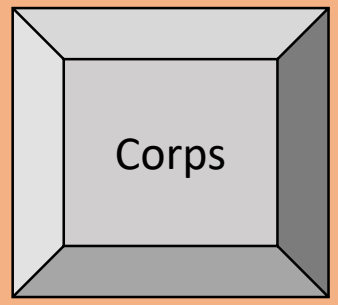
Des mises en pages(Layouts) dynamiques

- Une activité est généralement composée de plusieurs mis en page

- Elles sont dynamiques pour présenter les fonctionnalités:
  - Des fenêtres que l'on peut ouvrir et fermer
  - Des listes dynamiques avec l'ajout de nouveaux items
  - Ajouter et supprimer des éléments
  - Un agencement personnalisé
  - …



La conception des conteneurs(Listes) dynamiques:



Présentation des données:

- Concevoir la forme de présentation
- Identifier la forme
- Identifier la donnée
- Relier donnée/forme

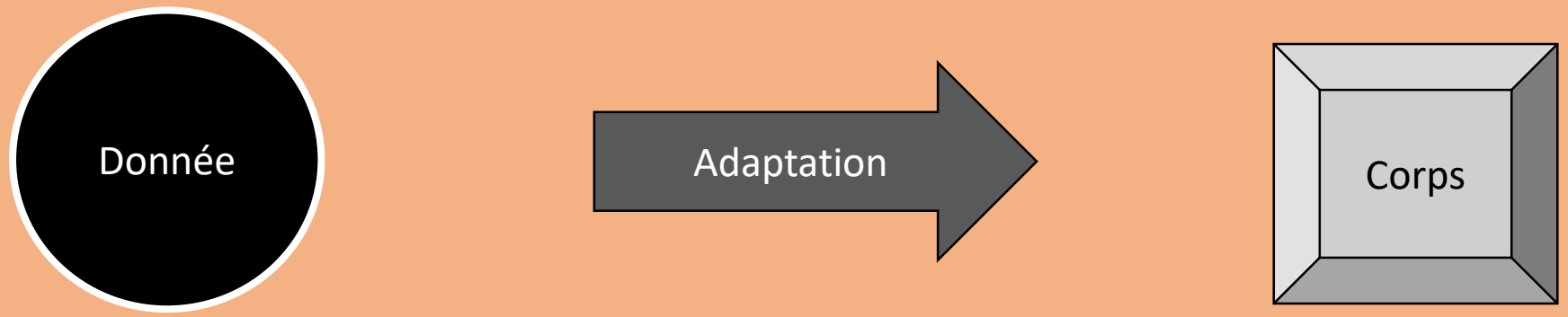
➢ Fichier: .xml
➢ Fichier: .kt

Remplir la liste avec les éléments:

- Identifier la liste
- Définir son adaptation des éléments qui la compose
- Gestion de la liste

➢ Fichier: .xml
➢ Fichier: MaintActivity.kt

La conception des items: adapter



Les données:
- Identifier la donnée
- Pouvoir y accéder

➢ Fichier: MaintActivity.kt
➢ Fichier: .kt

Présentation des données:
- Concevoir la forme de présentation
- Identifier la forme
- Identifier la donnée
- Relier donnée/forme

➢ Fichier: .xml
➢ Fichier: .kt

Développer l'IHM principale: la liste


```xml
<ListView
    android:id="@+id/list"
    android:layout_width="match_parent"
    android:layout_height="500dp"
    android:contentDescription="@string/une_liste"
    android:layout_below="@+id/title"
    android:layout_margin="25dp"
    android:padding="10dp"
    android:background="@drawable/border"
    android:dividerHeight="5dp"
    tools:listitem="@layout/theitem"/>
```

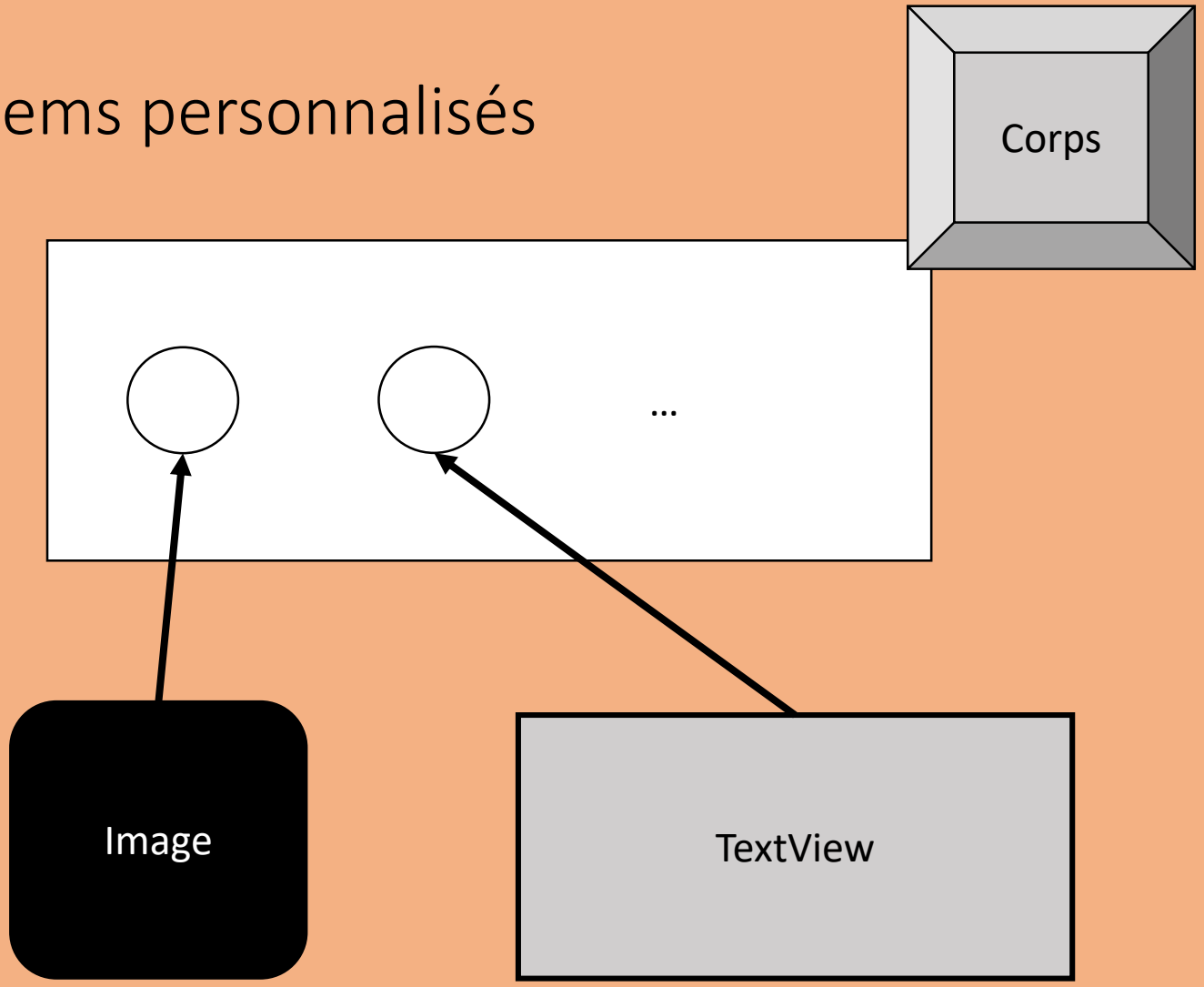
Fichier:
activity_main.xml

```kotlin
val list = findViewById<ListView>(R.id.list)

list.adapter = ???
list.isClickable = true
list.setOnItemClickListener { parent, view, position, id -> }
```

MaintActivity.kt

Concevoir les items personnalisés



Définir la composition des groupes de données

Donnée

```
data class ListData(var image: Int, var text: String)
```

Fichier: .kt

Adapter les données avec leurs mise en forme: Adapter


```
class ListAdapter(context: Context, dataArrayList: ArrayList<ListData?>?) :
    ArrayAdapter<ListData?>(context, R.layout.theitem, dataArrayList!!) {

    override fun getView(position: Int, view: View?, parent: ViewGroup): View {
        var view = view
        val listData = getItem(position)

        if (view == null) {
            view = LayoutInflater.from(context).inflate(R.layout.theitem, parent, false)
        }
```

Adapter les items à la liste:
Adapter

Adaptation

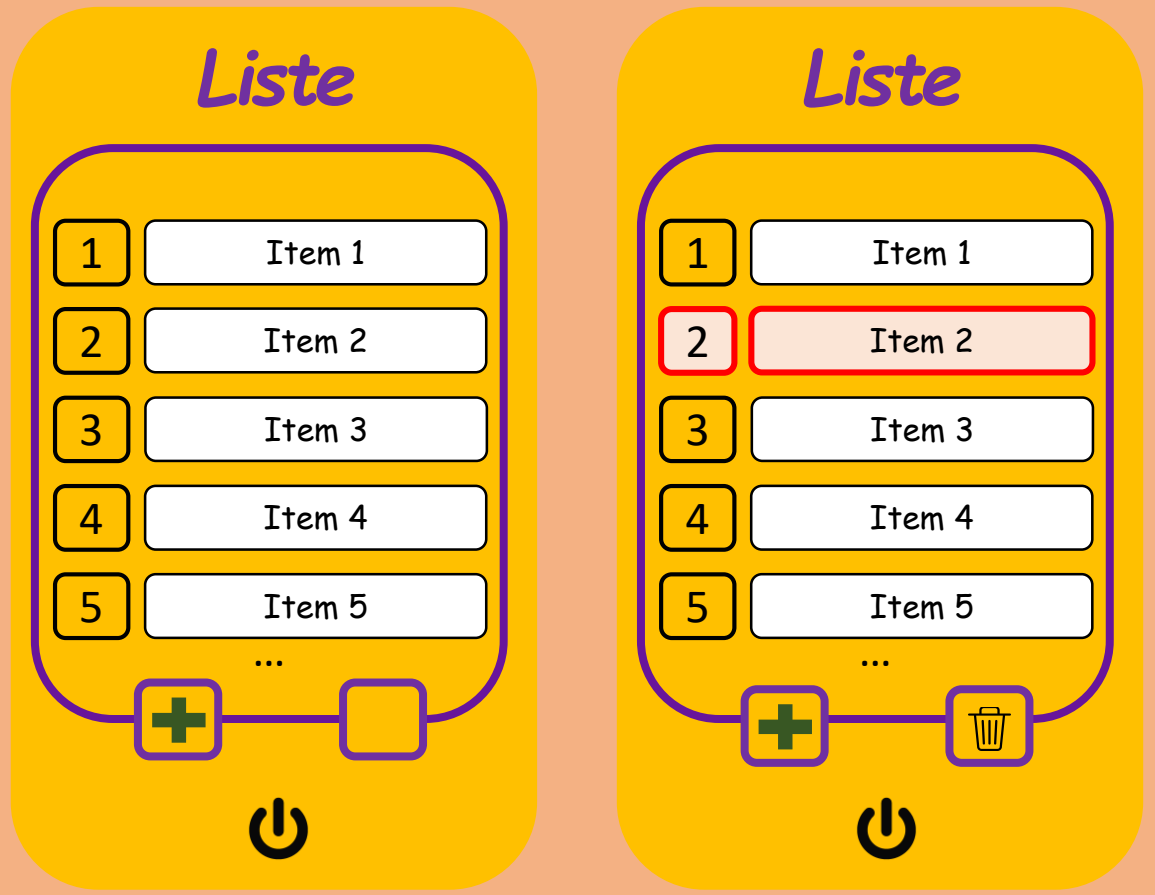
```
val list = findViewById<ListView>(R.id.list)
```

```
val listAdapter = ListAdapter(this, this.dataArrayList)
var itemFlag = View(this)
var itemColor = getColor(R.color.holo_purple)

list.adapter = listAdapter
```

Troisième exemple

Exercice 03: la maquette



Titre: « Une liste dynamique »

Objectif: L'application présente une liste dans laquelle nous pouvons ajouter et supprimer des éléments. Les éléments ont un style (une forme, des couleurs) personnalisé et un contenu (un numéro, un texte) définit.

L'interface est dîtes dynamique pour deux raisons:

1. la première est que l'on peut ajouter un éléments dans la liste
2. La deuxième est que le bouton supprimer apparaît lorsque l'on sélectionne un élément

Troisième exercice

Exercice supplémentaire: RecyclerView

Il s'agit d'un exercice similaire à celui de la liste.
Le RecyclerView permet d'agencer les éléments d'une liste sous différentes formes(lignes ou grilles).
Les deux exercices suivants sont conçu par les développeurs de chez Android pour maîtriser les bases des conteneurs dynamiques.

- Première partie : https://developer.android.com/codelabs/basic-android-kotlin-training-recyclerview-scrollable-list?hl=fr#0
- Deuxième partie: https://developer.android.com/codelabs/basic-android-kotlin-training-display-list-cards?hl=fr#0