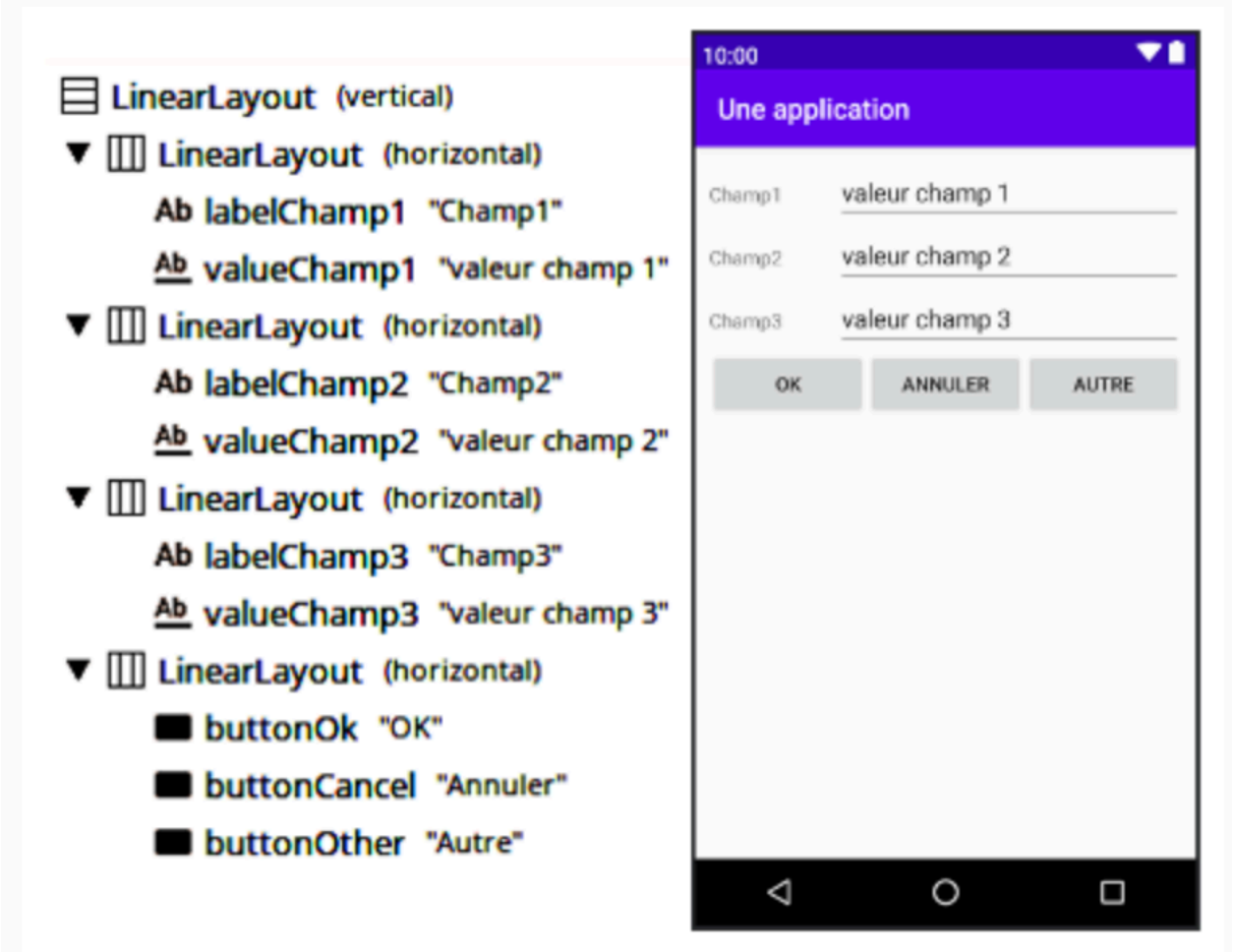
Widgets graphiques

La plupart des applications Android impliquent une interaction avec l'utilisateur. Nous avons introduit au chapitre précédent la notion d'activité qui correspond à un écran. Dans ce chapitre nous allons voir plus en détail comment créer des activités riches et responsives.

View et Layout

Une activité est composée de **views**. Il s'agit des composants graphiques qui peuvent être affichés à l'écran (on parle plus généralement en programmation de *widgets* graphiques). Par exemple, une zone de texte ou un bouton sont des views représentés en Java respectivement par les classes **TextView** et **Button**. Certaines vues permettent de grouper d'autres vues ensembles (**ViewGroup**) afin de créer une hiérarchie structurée des composants graphiques à afficher.

Exemple de hiérarchie des composants graphiques d'une activité



La difficulté d'une bonne conception d'interfaces graphiques avec Android et de concevoir des activités responsives, c'est-à-dire capables de s'adapter à des configurations diverses : la résolution de l'écran qui varie suivant les modèles d'appareil, la ratio hauteur / largeur de l'écran qui dépend de l'orientation de l'appareil (mode paysage ou mode portrait), la langue utilisée, la configuration de l'utilisateur… De plus une application Android peut parfois être installée sur des appareils très différents : smartphones, tablettes, Smart TV…

Pour aider à la conception d'interfaces graphiques, l'API Android fournit des *layouts*.
Un layout agit comme un ViewGroup : il peut contenir plusieurs vues. Un **layout** n'a pas de représentation graphique particulière. Cependant, il va positionner à l'écran les vues qu'il contient

suivant un modèle et en s'adaptant aux conditions particulières d'affichage. Par exemple, le layout **LinearLayoutCompat** permet d'organiser toutes le vues qu'il contient horizontalement ou verticalement.

Layout XML

Il est possible de créer un fichier **XML** pour décrire le layout d'une activité (le terme layout est ici à prendre au sens général d'un arrangement de composants graphiques sur l'écran). Au démarrage de l'activité, on associe ce layout grâce à la méthode **setContentView** :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Avec Android Studio, vous avez la possibilité d'utiliser l'outil de conception graphique ou d'éditer et de modifier directement le fichier XML.

Layout linéaire et positionnement

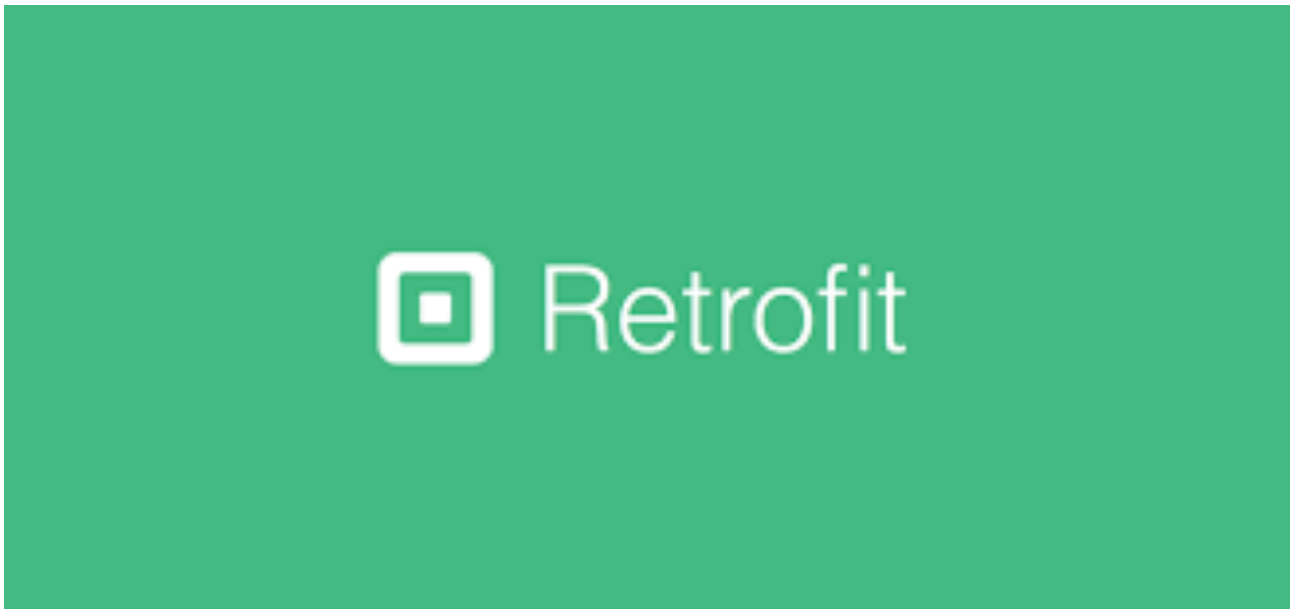
Pour illustrer les bases du layout, nous allons utiliser le fichier suivant :

Layout linéaire vertical

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bonjour" />

</androidx.appcompat.widget.LinearLayoutCompat>
```

Ce fichier commence par utiliser un **LinearLayoutCompat**. Ce dernier permet d'organiser les vues qu'il contient horizontalement ou verticalement. L'attribut android:orientation indique pour cet exemple que le placement sera vertical.

Client HTTP (Retrofit)



Un client REST de type sécurisé pour Android et Java.

Retrofit transforme votre API REST en une interface Java. Il utilise des annotations pour décrire les requêtes HTTP, le remplacement des paramètres URL et la prise en charge des paramètres de requête est intégrée par défaut. En outre, il fournit des fonctionnalités pour le téléchargement de requêtes et de fichiers en plusieurs parties.

Lien vers la documentation officielle

L'objectif de Retrofit est de fournir un framework puissant qui vous permet de mettre en place une abstraction simple et cohérente pour vos appels réseaux, changeant votre API HTTP en interface Java.

Ainsi Retrofit :

- Permet de déclarer votre couche réseau sous forme d'une interface ;
- Fournit un objet Call qui encapsule l'interaction avec une requête et sa réponse ;
- Permet de paramétrer l'objet Response ;
- Offre de multiples et efficaces convertisseurs (XML, JSON) ;
- Offre de multiples mécanismes pluggables d’exécution.

Bref, elle vous simplifie la vie au niveau de la couche réseau de votre application, tout en implémentant pour vous les bonnes pratiques d'abstraction et en vous permettant de personnaliser son comportement.

De plus, elle est bâtie sur OkHttp pour convertir vos objets JSON.

OkHttp est un client HTTP, pour la lecture/écriture du flux de données. Son objectif est de prendre en charge la communication avec le serveur.

Sérialisation / Désérialisation (GSON)

GSON est une bibliothèque Java open source pour pour convertir un objet Java dans sa représentation JSON et vise-versa.

GSON propose un système d'annotations pour permettre de personnaliser les opérations de sérialisation/désérialisation, on retrouve notamment :

- *@Expose* : Permet de préciser si un champ doit être utilisé ou non lors des opérations de sérialisation et de désérialisation.
- @SerializedName : Permet de préciser le nom du champ qui sera utilisé lors des opérations de sérialisation/désérialisation.
- @Since : Permet de définir à partir de quelle version le champ ou la classe doit être prise en compte.
- @Until : Permet de définir jusqu'à quelle version le champ ou la classe doit être prise en compte.

Comment les implémenter dans mon applications mobile ?

Dans le fichier « build.gradle » :

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.google.code.gson:gson:2.2.4'
    implementation 'com.squareup.retrofit2:retrofit:2.1.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
}
```