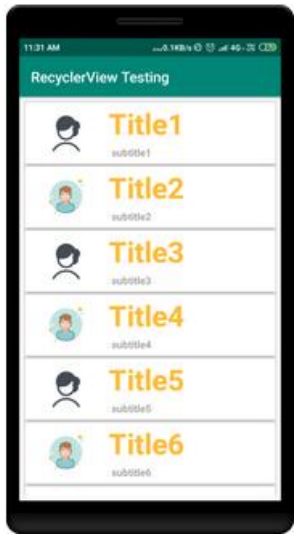
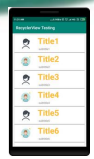
Kotlin



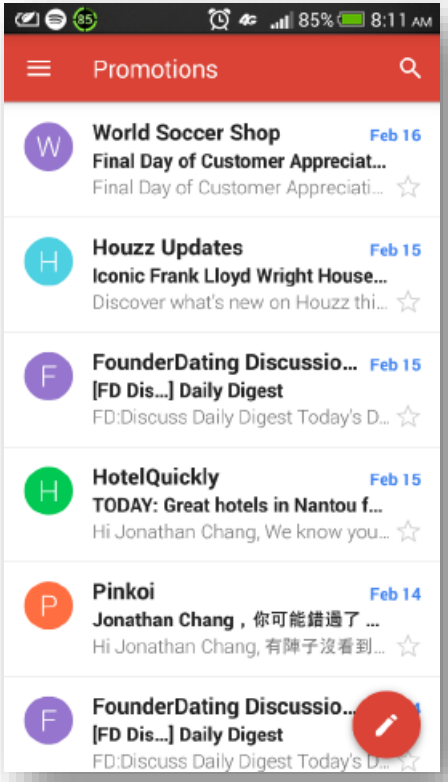
Programmation Android

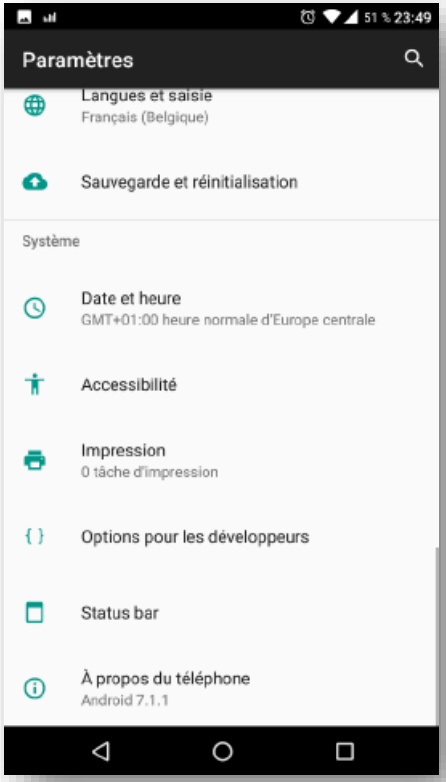
RecyclerView

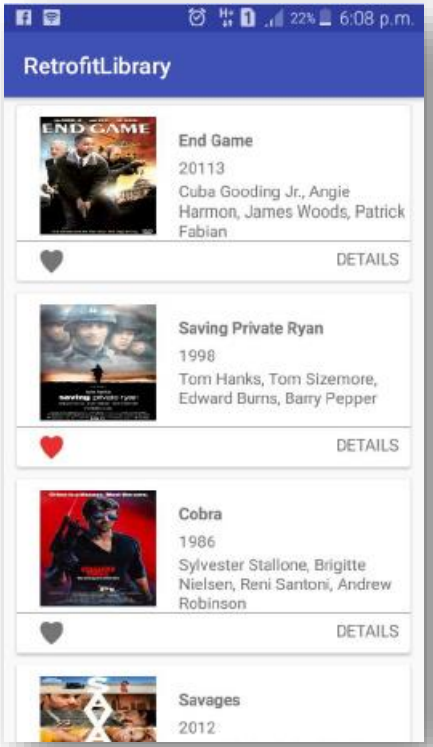


Présentation:

Le RecyclerView est un composant qui permet d'afficher une liste de vues construites sur des structures identiques.

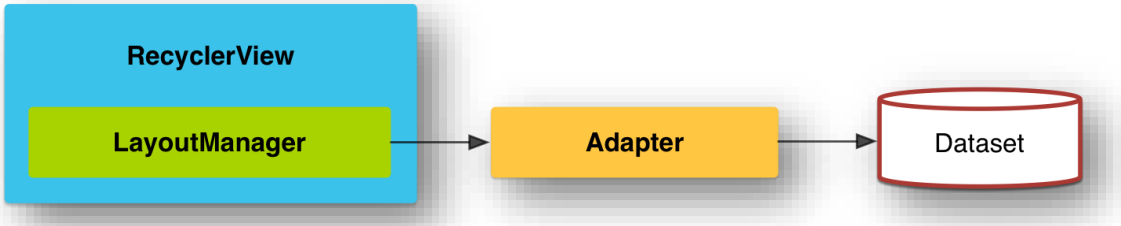






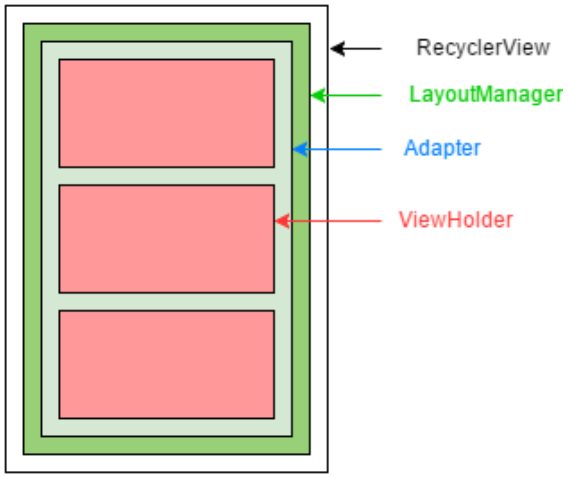
Fonctionnement:

Les données à afficher (Dataset) sont contenues en général dans une Collection (ArrayList).



*Le **RecyclerView** est lié à un **adapter***

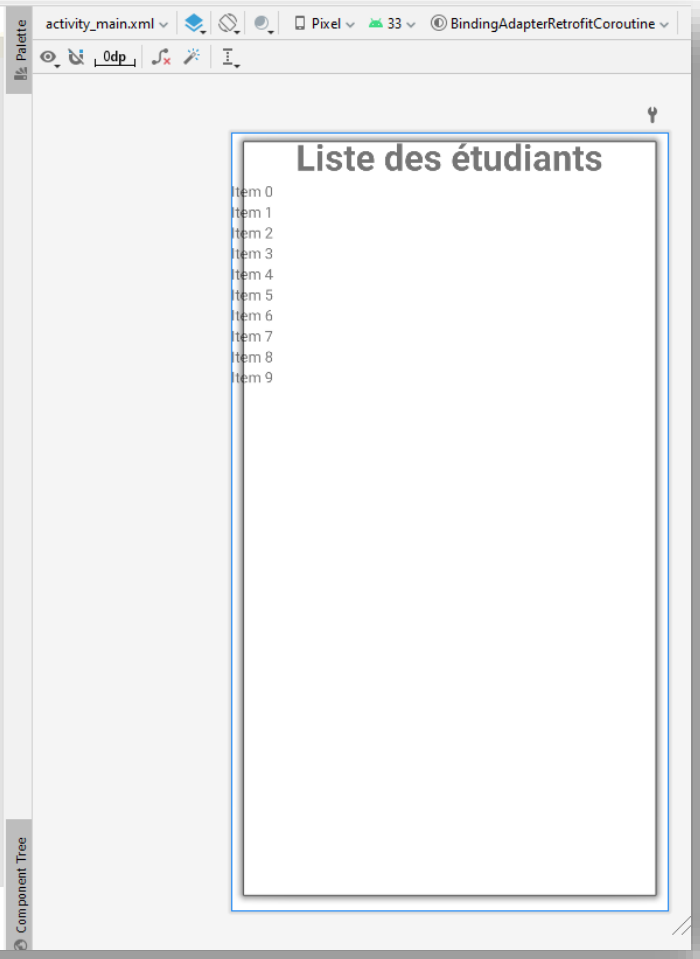
- *qui crée le nombre de vues nécessaires dans le **RecyclerView**,*
- *gère les **données à afficher**,*
- *et **recycle les objets** vues lorsqu'elles ne sont plus visibles après un scrolling,*



Implémentation:

A. *Créer le Layout Principal contenant un RecyclerView.*

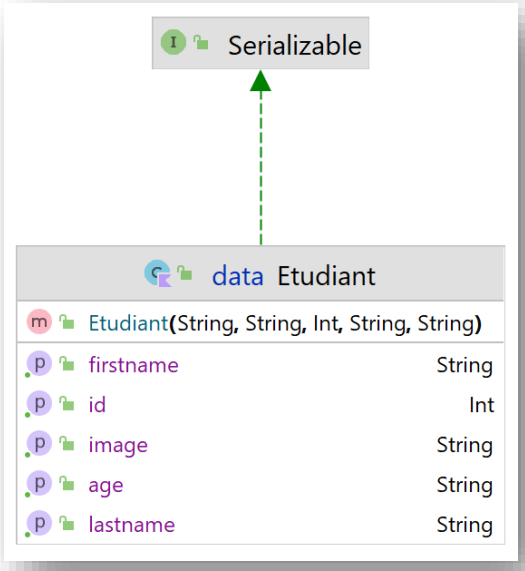
```xml
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity"
        android:background="@android:drawable/alert_light_frame">


    <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

        <TextView
                android:text="@string/titre"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/textView"
                android:textAppearance="@style/TextAppearance.AppCompat.Display1"
                android:textStyle="bold"
                android:textAllCaps="false"
                android:textAlignment="center"/>

        <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/rvEtudiants"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:clickable="true"
                android:focusable="true"/>
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```



Implémentation:

B. *Créer le POJO modèle (TousLesEtudiants et Etudiant) :*

- *Rendre la classe data Etudiant sérialisable pour des futurs bundles entre intents*



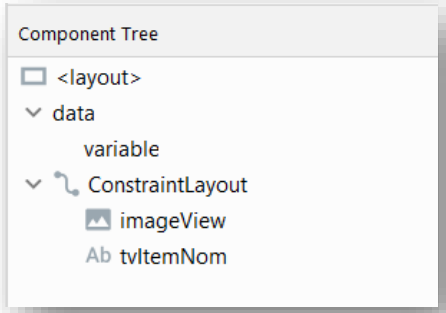

```
package com.example.bindingadapterretrofitcoroutine.model

import com.google.gson.annotations.SerializedName
import java.io.Serializable

data class Etudiant(
    @SerializedName("age")
    val age: String,
    @SerializedName("firstname")
    val firstname: String,
    @SerializedName("id")
    val id: Int,
    @SerializedName("image")
    val image: String,
    @SerializedName("lastname")
    val lastname: String,
    ): Serializable {

    override fun toString(): String {
        return "($id) $firstname $lastname $age\n"
    }

}
```

- *TousLesEtudiants est une classe avec comme paramètre une ArrayListe d' Etudiant*

```
class TousLesEtudiants : ArrayList<Etudiant>()
```

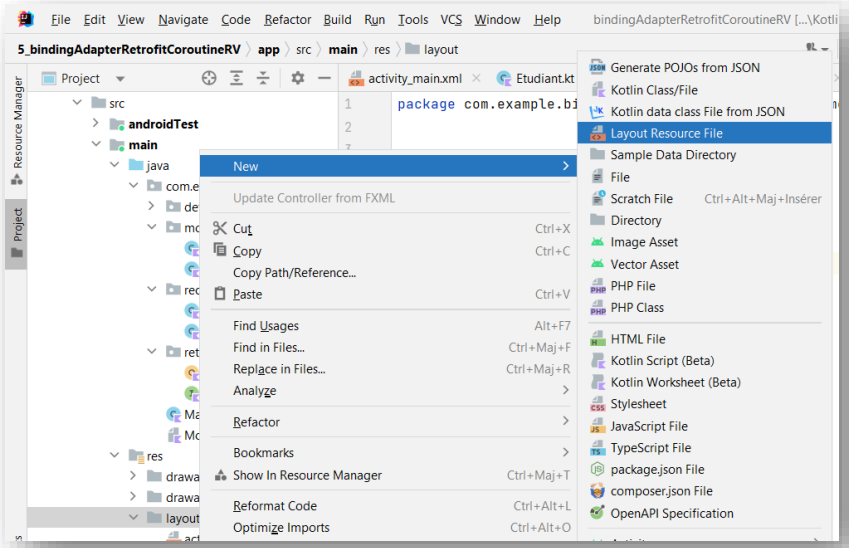
Implémentation:

C. *Créer le Layout correspondant à la vue à répéter (1/7):*



Remarque: item_Etudiants.xml

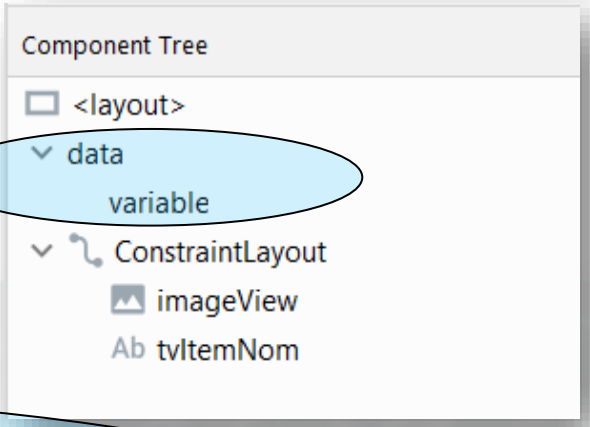
New ->Layout Ressource File



Implémentation:

C. *Créer le Layout correspondant à la vue à répéter (2/7):*

Remarque: utilisation du Jetpack Data Binding




```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto">

    <data>
        <variable
                name="etudiant"
                type="com.example.bindingadapterretrofitcoroutine.model.Etudiant" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        ...
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Implémentation:

C. *Créer le Layout correspondant à la vue à répéter (3/7):*

Remarque: utilisation du Jetpack Data Binding: dans la balise TexteView il faudra lier le paramètre android:texte avec les éléments firstname et lastname du modèle étudiant.

```
<TextView
        android:id="@+id/tvItemNom"
        android:text="@{etudiant.firstname.concat(' '+etudiant.lastname)}"
        …/>
```

Remarque: On peut concaténer des strings pour former une nouvelle chaîne:
`etudiant.firstname.concat(' '+etudiant.lastname)}`

Implémentation:

C. *Créer le Layout correspondant à la vue à répéter (4/7):*

Remarque: utilisation du Jetpack Data Binding: dans la balise ImageView il faudra charger et représenter l'image et pointée par l'URL !

Pour cela nous utiliserons la librairie glide

a. *buildGradle:*

```
plugins {
    …
    id 'kotlin-kapt' //processeur "d'annotation", également appelé KAPT obligatoire pour Glide
}
…
dependencies {
    //Glide: bibliothèque de chargement d'images
    implémentation 'com.github.bumptech.glide:glide:4.14.2'
    …
}
```

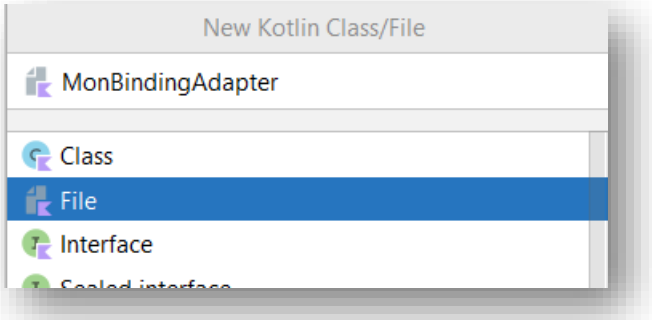
Implémentation:

C. Créer le Layout correspondant à la vue à répéter (5/7):

b. Créer un bindingAdapter (Adaptateurs de liaison):

Les adaptateurs de liaison sont des méthodes annotées dans n'importe quelle classe.
En règle générale, vous organisez vos adaptateurs en classes en fonction du type de vue cible.
Cela signifie évidemment qu'au moment de la compilation, toutes les méthodes de n'importe quelle classe avec l'annotation @BindingAdapter généreront le BindingAdapter.

Ici utilisation du: File.kt



Implémentation:

C. *Créer le Layout correspondant à la vue à répéter (6/7):*

c. *Créer un bindingAdapter (Adaptateurs de liaison):*

MonBindingAdapter.kt

```
import android.widget.ImageView
import androidx.databinding.BindingAdapter
import com.bumptech.glide.Glide

@BindingAdapter("lireImageUrl")
fun ImageView.loadImageUrl(url:String?){
    try {
        Glide.with(context).load(url).into(this)
    }catch (e:Exception){}
}
```

Implémentation:

C. *Créer le Layout correspondant à la vue à répéter (7/7):*

d. layout

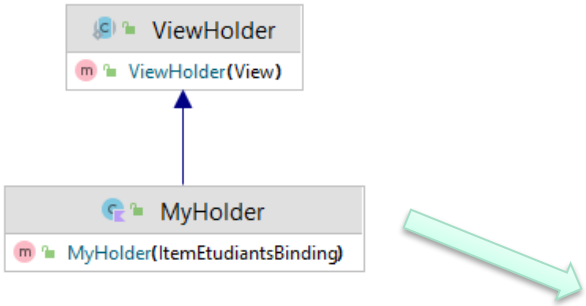
```
<ImageView
        android:id="@+id/imageView"
        lireImageUrl="@{etudiant.image}"
        …/>
```

```
@BindingAdapter(« lireImageUrl")
fun ImageView.loadImageUrl(url:String?){
    …
}
```

Implémentation:

D. *Créer la classe MyHolder:*

- *Cette classe permet de **créer des objets liés à chaque vue de la liste**, qui seront réutilisés en cas de Scrolling.*
- *La classe doit hériter de la classe mère ViewHolder.*




```
import androidx.recyclerview.widget.RecyclerView.ViewHolder
import com.example.bindingadapterretrofitcoroutine.databinding.ItemEtudiantsBinding
import com.example.bindingadapterretrofitcoroutine.model.Etudiant

class MyHolder(private val binding: ItemEtudiantsBinding) : ViewHolder(binding.root) {

    fun bind(etudiant: Etudiant) {
        binding.etudiant = etudiant
    }
}
```

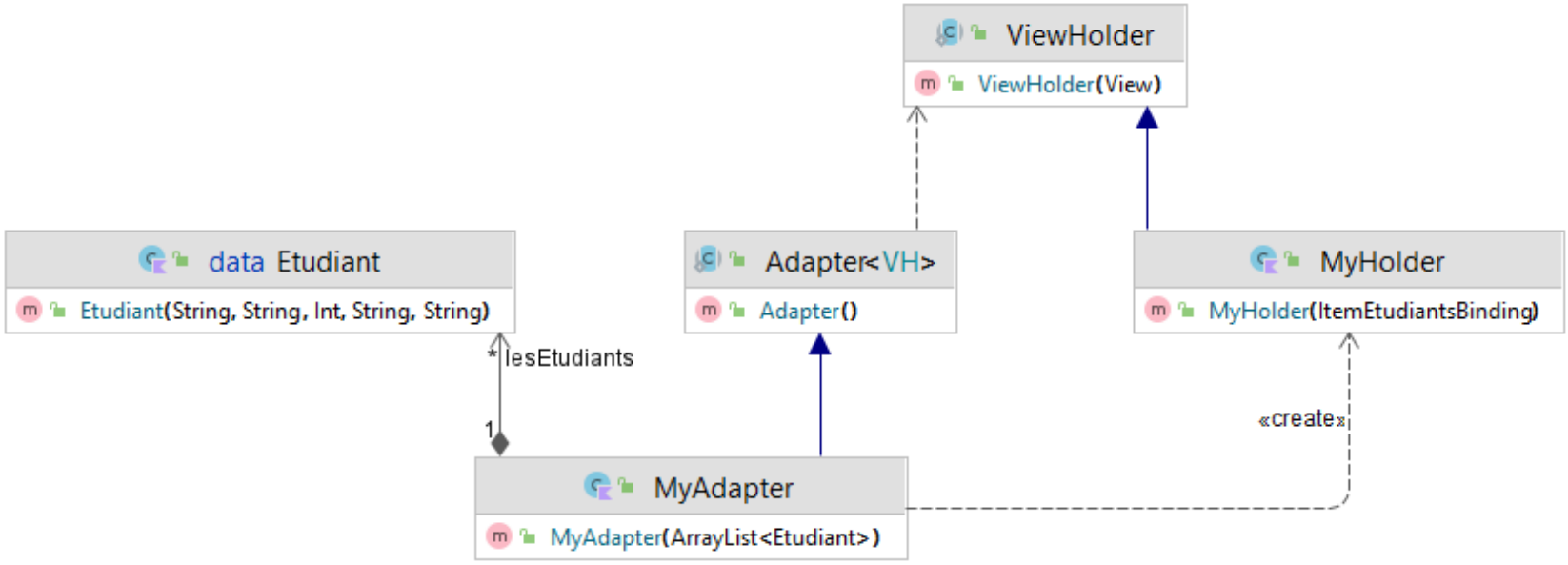
Implémentation:

E. *Créer la classe MyAdapter 1/5 :*

Cette classe

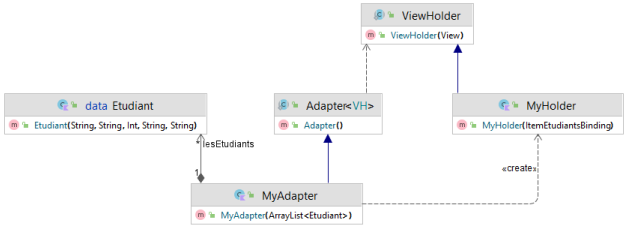
- ***gère les vues,***
- *et fait le **lien avec les données** contenues dans un ArrayList.*

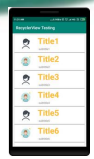
La classe MyAdapter doit hériter de l'Interface générique Adapter<VH>.



Implémentation:

E. *Créer la classe MyAdapter 2/5:*

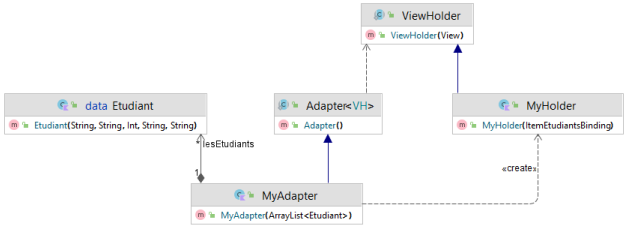


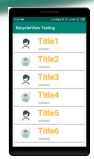


```
class MyAdapter(private val lesEtudiants: TousLesEtudiants) : RecyclerView.Adapter<MyHolder>(){

    private lateinit var binding: ItemEtudiantsBinding

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyHolder {
        binding = ItemEtudiantsBinding.inflate(LayoutInflater.from(parent.context), parent, false)
        return MyHolder(binding)
    }

    override fun onBindViewHolder(holder: MyHolder, position: Int) {
        holder.bind(lesEtudiants.get(position))
    }

    override fun getItemCount(): Int = lesEtudiants.size

}
```

Implémentation:

E. *Créer la classe MyAdapter 3/5 :*



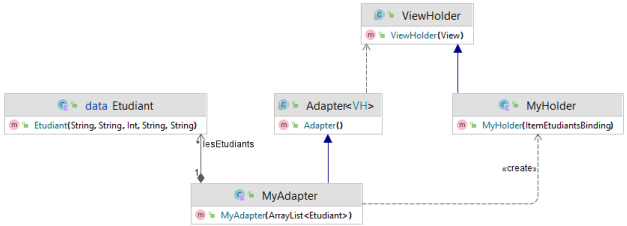


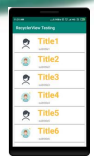
```
class MyAdapter(private val lesEtudiants: TousLesEtudiants) : RecyclerView.Adapter<MyHolder>(){

    private lateinit var binding: ItemEtudiantsBinding

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyHolder {
        binding = ItemEtudiantsBinding.inflate(LayoutInflater.from(parent.context), parent, false)
        return MyHolder(binding)
    }

    override fun onBindViewHolder(holder: MyHolder, position: Int) {
        holder.bind(lesEtudiants.get(position))
    }

    override fun getItemCount(): Int = lesEtudiants.size

}
```

onCreateViewHolder créer une vue à partir de l'Item_etudiant par un inflate (l'opération qui à partir d'une description xml, détermine l'arbre d'objets Java).
Puis crée un holder avec son dataBinding associé à cette vue

Implémentation:

E. Créer la classe MyAdapter 4/5 :



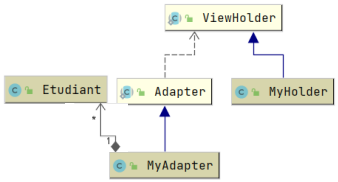


```
class MyAdapter(private val lesEtudiants: TousLesEtudiants) : RecyclerView.Adapter<MyHolder>(){

    private lateinit var binding: ItemEtudiantsBinding

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyHolder {
        binding = ItemEtudiantsBinding.inflate(LayoutInflater.from(parent.context), parent, false)
        return MyHolder(binding)
    }

    override fun onBindViewHolder(holder: MyHolder, position: Int) {
        holder.bind(lesEtudiants.get(position))
    }

    override fun getItemCount(): Int = lesEtudiants.size

}
```

onBindViewHolder créer le binding des données de l'ArrayList avec le holder créé par la méthode précédante

Implémentation:

E. *Créer la classe MyAdapter 5/5 :*




```
class MyAdapter(private val lesEtudiants: TousLesEtudiants) : RecyclerView.Adapter<MyHolder>(){

    private lateinit var binding: ItemEtudiantsBinding

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyHolder {
        binding = ItemEtudiantsBinding.inflate(LayoutInflater.from(parent.context), parent, false)
        return MyHolder(binding)
    }

    override fun onBindViewHolder(holder: MyHolder, position: Int) {
        holder.bind(lesEtudiants.get(position))
    }

    override fun getItemCount(): Int = lesEtudiants.size

}
```

1. Nous avons crée le holder lié à item_etudiants.xml (méthode onCreateViewHolder)
2. Puis nous avons lié les données extraites d'une ArrayList d'etudiant(méthode onCreateViewHolder)
3. Maintenant il faut par la méthode getItemCount lui donné le nombre d'itération pour qu'il fabrique par reproduction du modèle vu et données

Implémentation:

F. *Compléter classe MainActivity 1/2:*

Jetpack: viewBinding

```
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding= ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        executeCoroutineTousLesEtudiants()
    }

    private fun executeCoroutineTousLesEtudiants() {…}
}
```

Implémentation:

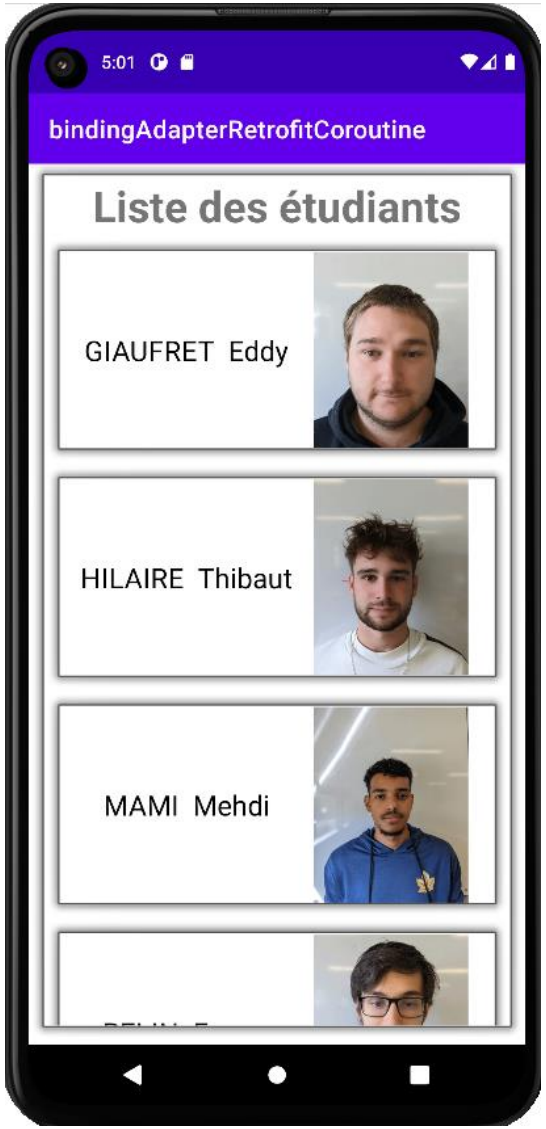
F. Compléter classe MainActivity 2/2:

Ici Dispatchers.Main->coroutine dans le thread principal. Utilisé lorsque nous devons effectuer les opérations d'interface utilisateur dans la coroutine, car l'interface utilisateur ne peut être modifiée qu'à partir du thread principal.

```
private fun executeCoroutineTousLesEtudiants() {

    GlobalScope.launch(Dispatchers.Main) {
        try {
            val rep = ApiAdapteur.apiClient.getTousLesEtudiantsEtudiants()
            if (rep.isSuccessful() && rep.body() != null) {
                val recyclerViewNews = binding.rvEtudiants
                //création de l'adapteur
                val newsAdapter = MyAdapter(rep.body()!!)
                recyclerViewNews.adapter = newsAdapter
                //création du layoutManager
                val layoutManager = LinearLayoutManager(parent, LinearLayoutManager.VERTICAL, false)
                recyclerViewNews.layoutManager = layoutManager
            } else {
                …
            }
        } catch (e: Exception) {
            …
        }
    }
}
```

context Orientation reverseLayout

Implémentation:

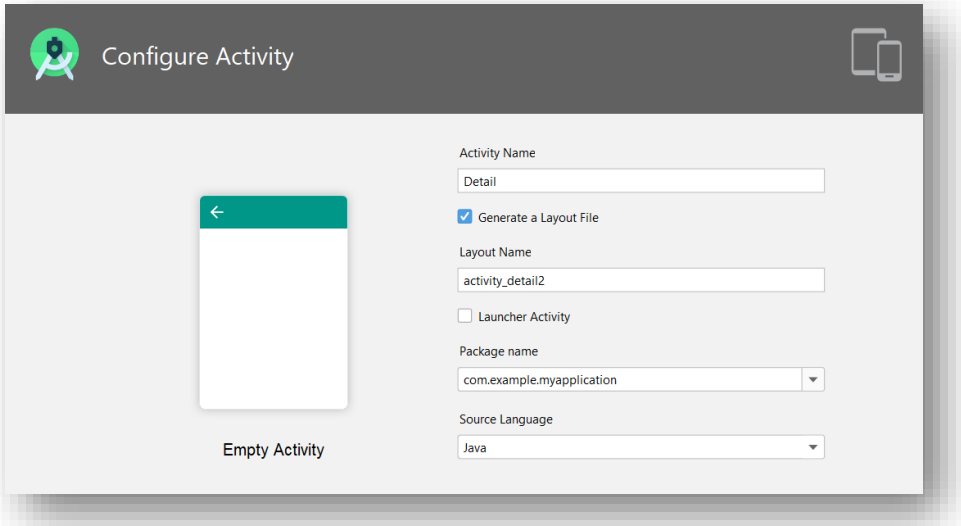


Rendre clikables les items :

A. *Créer un deuxième activité « Details » 1/2:*

Dans le package com…:

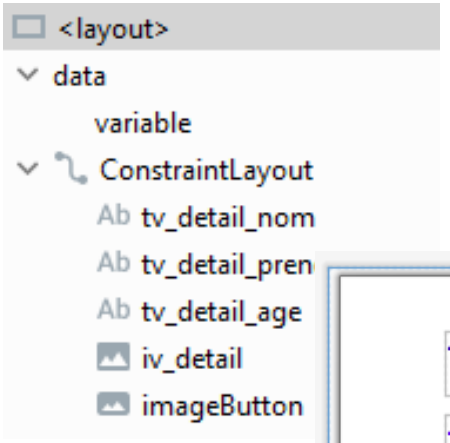
- ***New ->***
  - ***Activity ->***
    - ***Empty Activity ->***

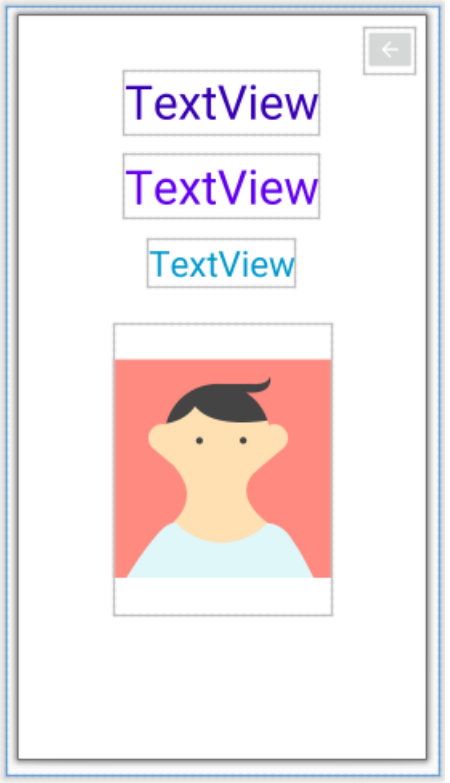


Rendre clikables les items :

A. *Créer un deuxième activité « Details » 2/2:*

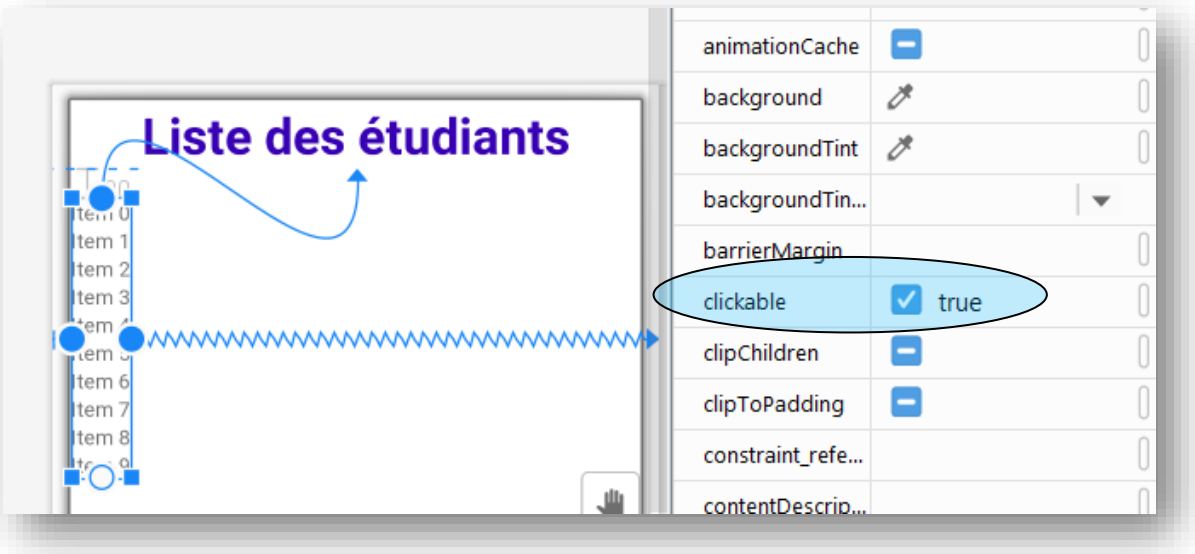
```
class Detail : AppCompatActivity() {

    private lateinit var binding: ActivityDetailBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding= DataBindingUtil.setContentView(this,
             R.layout.activity_detail)
        binding.infoEtudiant=
             intent.getSerializableExtra("objetEtudiant") as Etudiant?
        binding.imageButton.setOnClickListener {
            finish()
        }
    }
}
```





Rendre clikables les items :

B. *RecyclerView (activité main): attribut clickable -> true*



Rendre clikables les items :

C. *Mettre en place un Listener:*

Pour cela modifier la classe MyHolder:

```
import android.content.Intent
import androidx.recyclerview.widget.RecyclerView.ViewHolder
import com.example.bindingadapterretrofitcoroutine.databinding.ItemEtudiantsBinding
import com.example.bindingadapterretrofitcoroutine.detail.Detail
import com.example.bindingadapterretrofitcoroutine.model.Etudiant


class MyHolder(private val binding: ItemEtudiantsBinding) : ViewHolder(binding.root) {

    fun bind(etudiant: Etudiant) {
        binding.etudiant = etudiant
        binding.imageView.setOnClickListener{
            val intent = Intent(it.context, Detail::class.java)
            intent.putExtra("objetEtudiant",etudiant)
            println(etudiant.toString())
            it.context.startActivity(intent)
        }
    }
}
```

Rendre clikables les items :