

DÉVELOPPER UNE APPLICATION ANDROID NATIVE

---

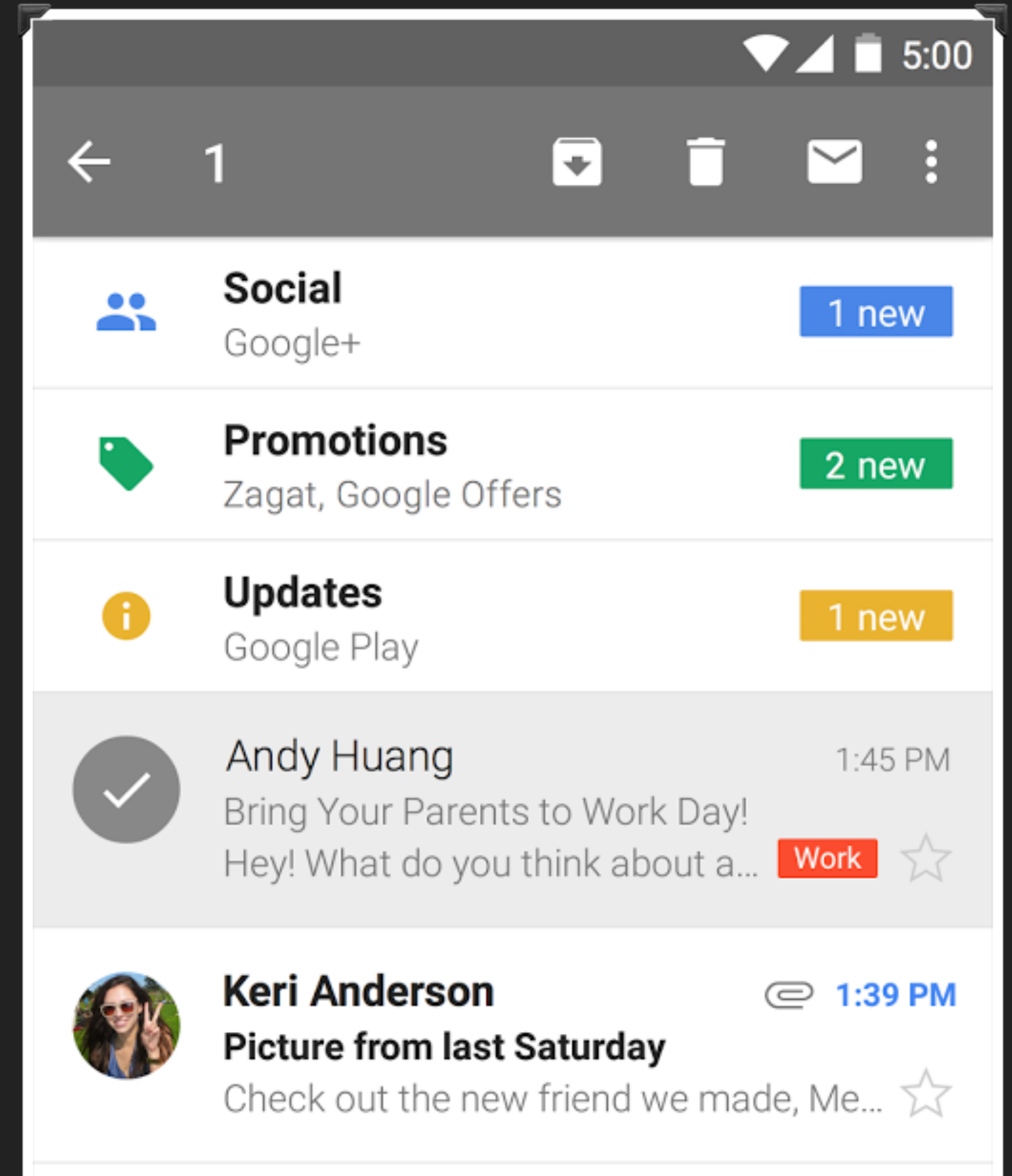
**LES RECYCLER VIEW**

## INTRODUCTION

- ▶ Jusqu'à maintenant, nous avons conçu des applications avec des ListView comme objet pour afficher une liste sur Android.
- ▶ Cependant, les ListView ne suffiront plus si vous souhaitez développer une application avec des listes plus complexes (ex.: Réseau social).
- ▶ Nous allons voir dans ce cours qu'est ce qu'une RecyclerView et quel sont ses avantages.

## QU'EST CE QU'UNE RECYCLERVIEW

- ▶ La RecyclerView est un objet qui permet d'afficher une liste (ex.: tweets, posts, mails...). Elle remplace la ListView.
- ▶ Elle contient des avantages non négligeables et devient indispensable pour tout projet d'envergure.



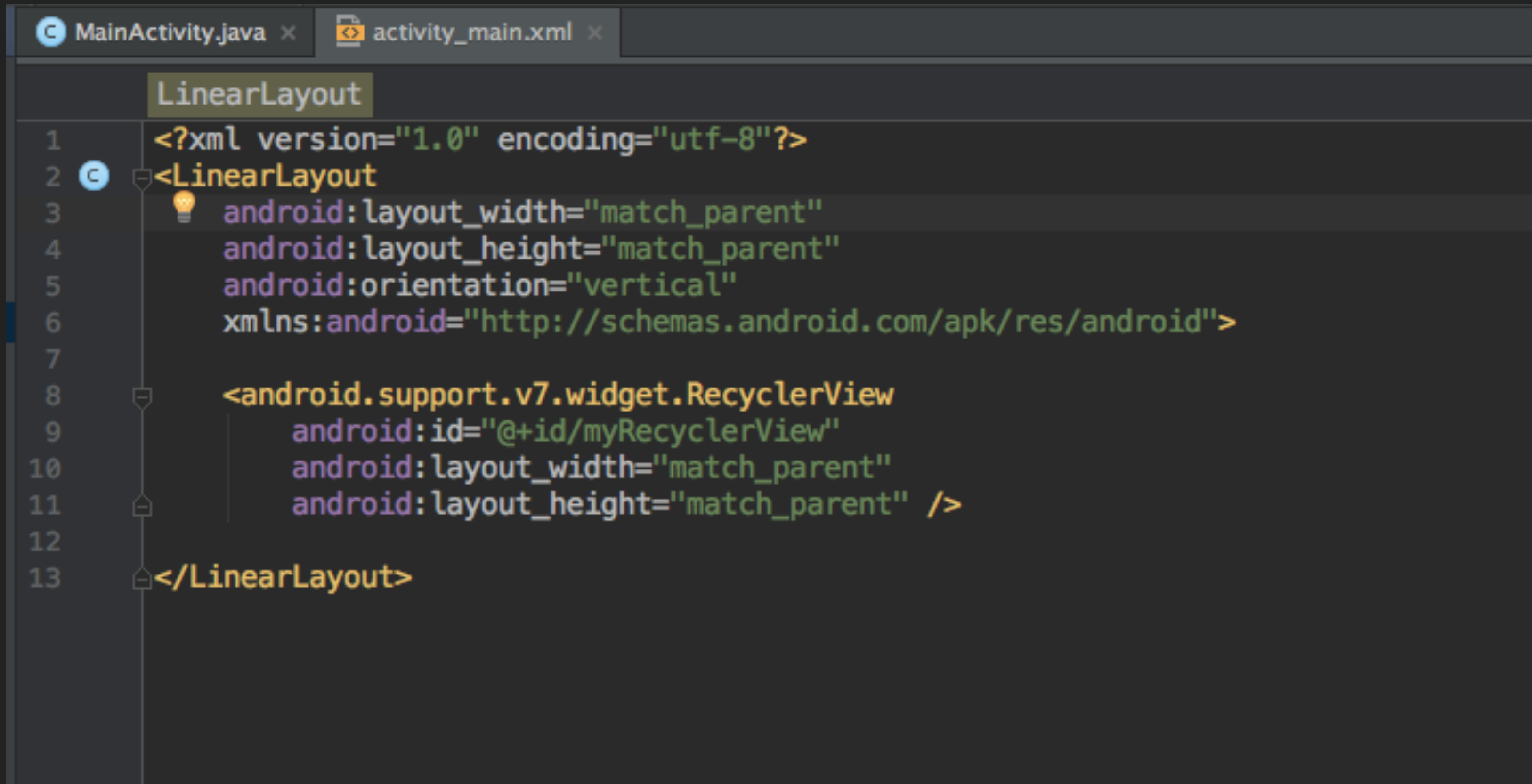
## LES DIFFÉRENCES AVEC LA LISTVIEW

- ▶ Dans une ListView classique, le système se charge de créer toutes les vues qui correspondent aux éléments de votre liste avant même qu'elles ne soient visible à l'écran. Ce système peut saturer la mémoire de votre machine virtuelle et lever une erreur **OutOfMemoryError**.
- ▶ Dans une RecyclerView, le système ne va charger que les vues qui seront visible à l'écran ! Lors du scroll, elle réutilisera les vues qui disparaissent pour charger les éléments suivants.



# MISE EN PLACE

- ▶ Commencez par créer un projet « Empty Activity » sur Android Studio
- ▶ Ajoutez ce code dans notre activity\_main.xml (il contient un LinearLayout comme parent et notre fameuse RecyclerView).



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical"
6      xmlns:android="http://schemas.android.com/apk/res/android">
7
8      <android.support.v7.widget.RecyclerView
9          android:id="@+id/myRecyclerView"
10         android:layout_width="match_parent"
11         android:layout_height="match_parent" />
12
13 </LinearLayout>
```



- Nous allons dans un premier temps, comme pour une ListView, créer un objet java qui va représenter l'entité qui sera affiché dans la liste, dans notre cas, un jeu vidéo, nous développons donc une liste de jeux vidéos :-)

```
1  package fr.tutoriel.umbertoemonds.recyclerview;
2
3  /**
4   * Created by umbertoemonds on 16/09/2017.
5   */
6
7  class JeuVideo {
8
9      private String name;
10     private float price;
11
12     JeuVideo(String name, float price) {
13         this.name = name;
14         this.price = price;
15     }
16
17     public String getName() { return name; }
20
21     public float getPrice() { return price; }
24
25 }
```

- ▶ Voici la vue qui représente nos cellules que j'ai nommé jeu\_video\_item.xml. Elle contient deux TextView, un pour la désignation du produit et l'autre pour le prix.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:textSize="18sp"
        tools:text="Forza Motorsport 5" />
    <TextView
        android:id="@+id/price"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="bottom|end"
        android:padding="10dp"
        tools:text="29,90€" />
</LinearLayout>
```



- ▶ Nous allons maintenant créer l'adapter, classe qui est responsable de chaque cellules dans notre RecyclerView. C'est ici que sera implémenté le mécanisme de recyclage des vues.
- ▶ Je vais créer une classe que je vais nommer MyVideosGamesAdapter qui héritera de RecyclerView.Adapter<ViewHolder> afin d'indiquer que notre classe est bien un adapter pour pouvoir surcharger ses méthodes. Si vous codez en même temps que moi, il est normal que vous ayez une erreur au niveau de l'héritage, nous allons devoir créer ce qu'on appelle un ViewHolder afin de pouvoir l'indiquer dans notre héritage.

```
class MyVideosGamesAdapter extends RecyclerView.Adapter<MyVideosGamesAdapter.MyViewHolder> {
```

- ▶ Veuillez à ajouter les bon imports

```
import android.support.v7.widget.RecyclerView;
```

- ▶ Le ViewHolder est un objet utilisé pour accélérer le rendu de votre liste. Dans les ListView, nous récupérions nos vues directement dans la méthode `getView()` avec nos `findViewById()`, ce qui signifiait qu'à chaque fois qu'une cellule se construisait, toutes les vues étaient de nouveau récupérées, liées.
- ▶ Le ViewHolder est un objet qui contient le modèle de nos cellules et évite ce problème car nous lions nos vues qu'une seule fois via nos `findViewById()` et les mettons simplement à jour à chaque recyclage de cellules.
- ▶ Le ViewHolder se déclare comme une classe au sein de notre adapter.

- ▶ J'ai nommé ma classe MyViewHolder et je la fait hériter de RecyclerView.ViewHolder pour indiquer que c'est un ViewHolder !
- ▶ Rappelez-vous que le ViewHolder est le modèle pour toutes nos cellules.
- ▶ Je génère le constructeur par défaut obligatoire. Notez qu'il contient itemView comme paramètre ce qui représente la vue de chaque cellules.
- ▶ Avant de finaliser notre ViewHolder, nous allons créer la vue xml de notre cellule.

```
class MyViewHolder extends RecyclerView.ViewHolder{

    private TextView mNameTV;
    private TextView mPriceTV;

    MyViewHolder(View itemView) {
        super(itemView);

        mNameTV = (TextView)itemView.findViewById(R.id.name);
        mPriceTV = (TextView)itemView.findViewById(R.id.price);
    }

    void display(JeuVideo jeuVideo){

        mNameTV.setText(jeuVideo.getName());
        mPriceTV.setText(jeuVideo.getPrice() + "€");
    }
}
```



- ▶ Maintenant que ma vue est créée, je vais déclarer deux variables de type TextView et les lier à mon ViewHolder dans le constructeur via la méthode findViewById().
- ▶ Nous allons maintenant créer une méthode au sein de notre ViewHolder qui servira uniquement à mettre à jour nos cellules avec les informations de notre liste. Cette méthode sera appelée à chaque recyclage d'une cellule.
- ▶ Je l'ai appelée display(JeuVideo jeuVideo) et je lui passe en paramètre un jeu vidéo.

```
class MyViewHolder extends RecyclerView.ViewHolder{

    private TextView mNameTV;
    private TextView mPriceTV;

    MyViewHolder(View itemView) {
        super(itemView);

        mNameTV = (TextView)itemView.findViewById(R.id.name);
        mPriceTV = (TextView)itemView.findViewById(R.id.price);
    }

    void display(JeuVideo jeuVideo){

        mNameTV.setText(jeuVideo.getName());
        mPriceTV.setText(jeuVideo.getPrice() + "€");
    }

}
```

- Maintenant que notre ViewHolder est fin prêt, finalisons notre adapter. Explications.

```
List<JeuVideo> mesJeux;

MyVideosGamesAdapter(List<JeuVideo> mesJeux){
    this.mesJeux = mesJeux;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    LayoutInflater inflater = LayoutInflater.from(parent.getContext());
    View view = inflater.inflate(R.layout.jeu_video_item, parent, false);
    return new MyViewHolder(view);
}

@Override
public void onBindViewHolder(MyViewHolder holder, int position) {
    holder.display(mesJeux.get(position));
}

@Override
public int getItemCount() {
    return mesJeux.size();
}
```



- ▶ J'ai ajouté comme variable globale notre liste de jeux vidéos et j'ai généré le constructeur afin de passer la liste que nous voulons afficher lors de la création de l'adapter.
- ▶ J'ai surchargé la méthode `onCreateViewHolder()` qui, comme son nom l'indique, est exécutée lors de la création de notre `ViewHolder`, cette méthode n'est exécutée qu'une seule fois, c'est pourquoi nous allons charger la vue `.xml` de notre cellule ici.
- ▶ J'ai surchargé la méthode `onBindViewHolder()` qui est exécutée lorsque que le `ViewHolder` est de nouveau lié à l'adapter, c'est à dire lors du recyclage d'une cellule, nous recevons d'ailleurs comme paramètre sa position et l'instance du `ViewHolder`. Je vais me contenter dans cette méthode de mettre à jour ma cellule courante grâce à ma méthode `display()` qui rafraichit mes `TextView`.
- ▶ Je finit par surcharger la méthode `getItemCount()` qui permet de renvoyer le nombre d'items de notre liste, comme nous ne savons pas forcément combien d'éléments il y aura dans notre liste, j'utilise la méthode `.size()` sur la liste de jeux vidéos.

## FINALISATION

- Rendons nous maintenant dans notre activité MainActivity, activité qui contient notre liste

```
public class MainActivity extends AppCompatActivity {  
  
    private RecyclerView mRecyclerView;  
    private List<JeuVideo> mesJeux;  
    private MyVideosGamesAdapter monAdapter;
```

- J'ai déclaré 3 variables, celle de ma RecyclerView, la liste de jeux vidéos qui sera affiché dans la RecyclerView et celle de mon fameux adapter créé précédemment.

- ▶ Voici le code que je vais ajouter dans ma méthode onCreate(). Explications.

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

    mRecyclerView = (RecyclerView)findViewById(R.id.myRecyclerView);

    mesJeux = new ArrayList<>();

    mesJeux.add(new JeuVideo("Forza Horizon 3", 49.90F));
    mesJeux.add(new JeuVideo("Forza Motorsport 5", 20.00F));
    mesJeux.add(new JeuVideo("Asseto Corsa", 22.00F));
    mesJeux.add(new JeuVideo("Gran Theft Auto 5", 49.90F));

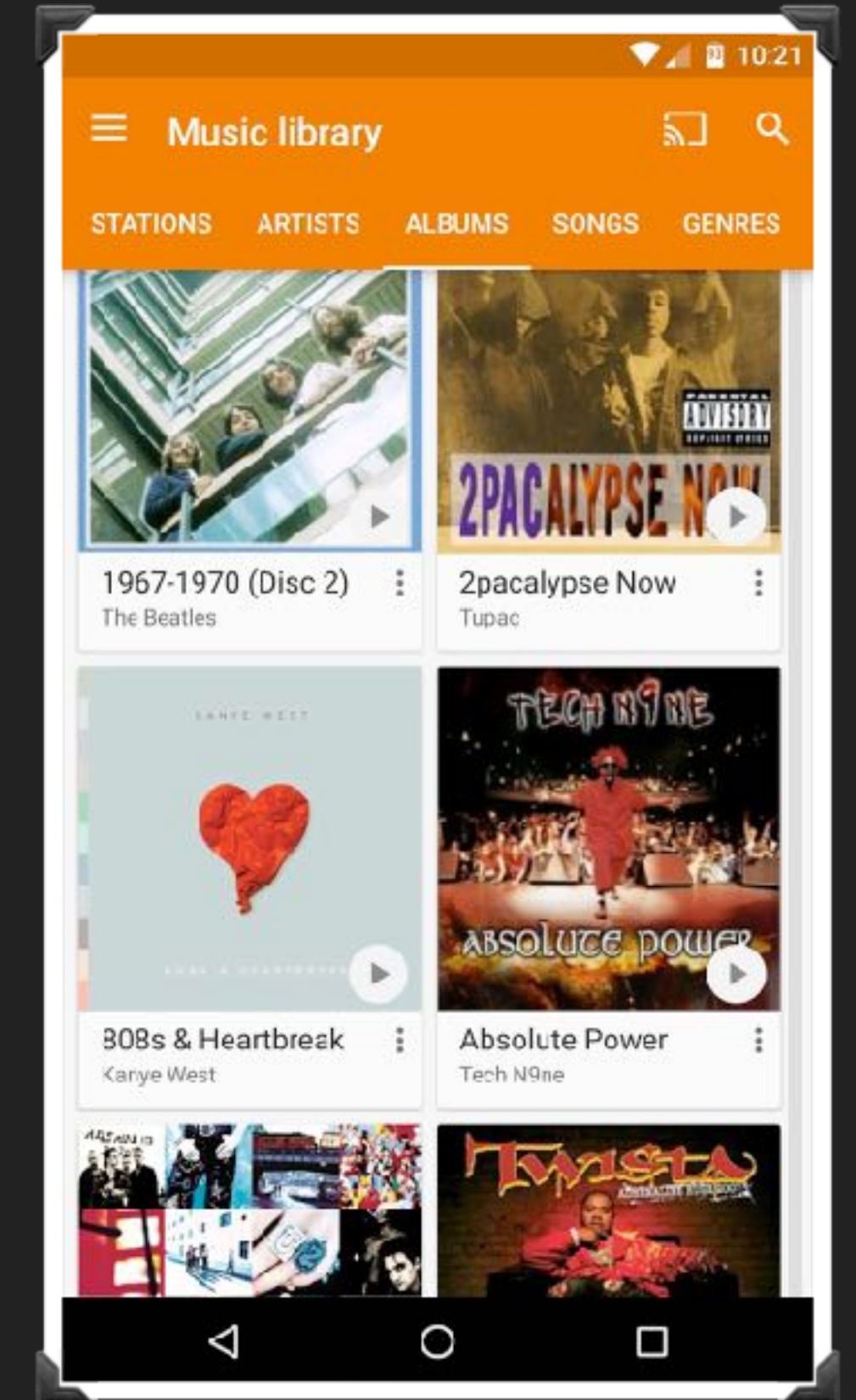
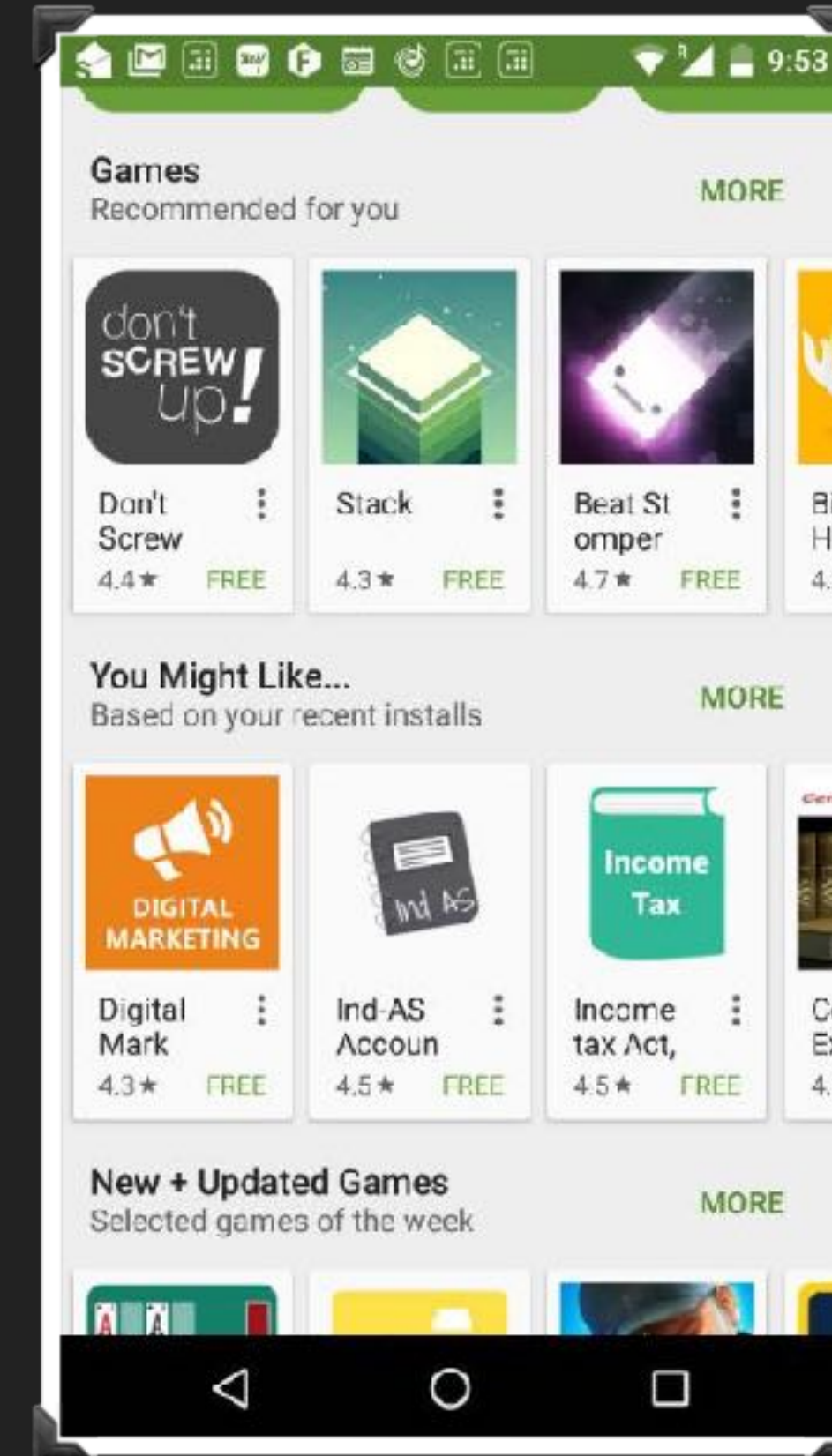
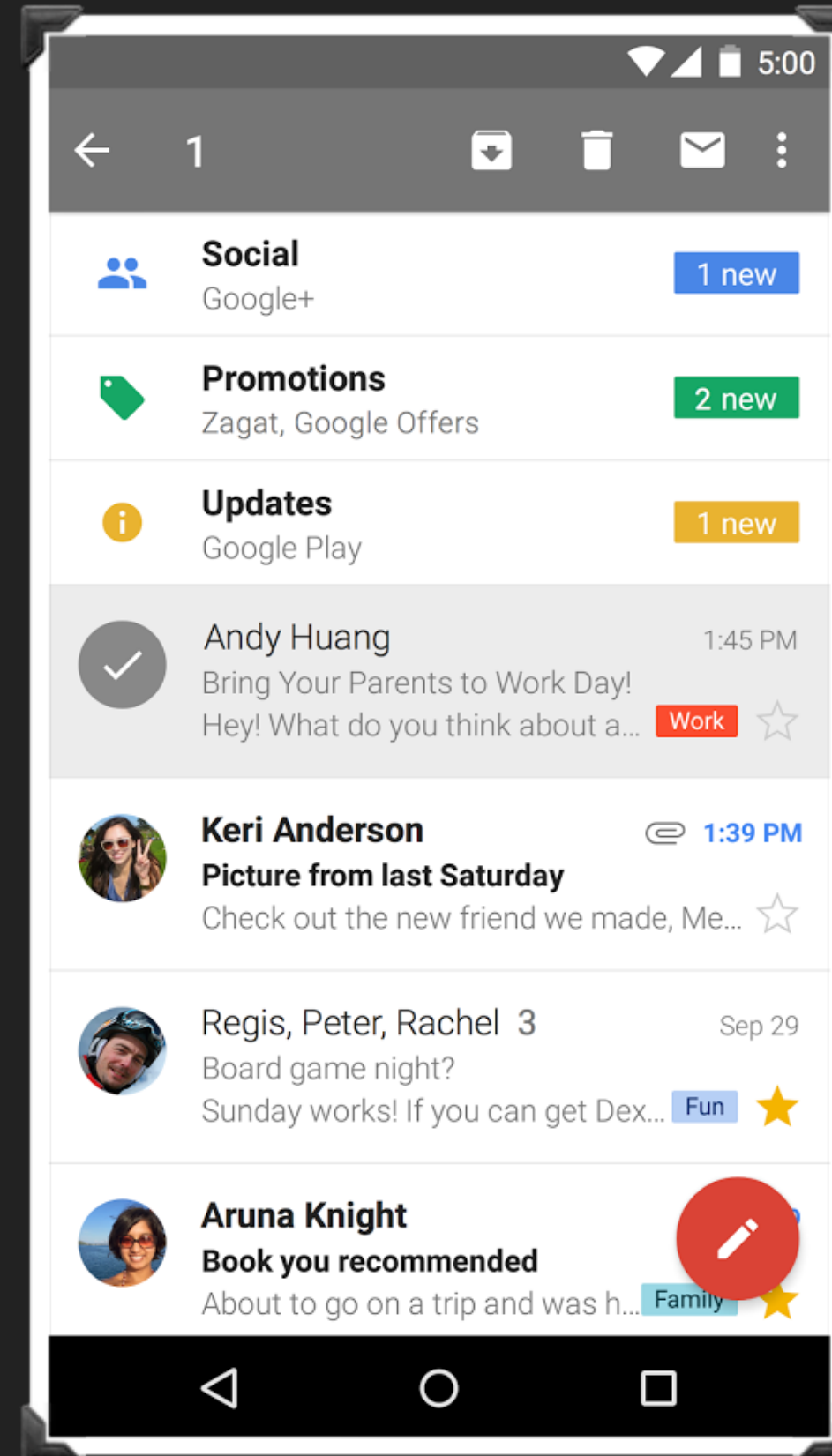
    monAdapter = new MyVideosGamesAdapter(mesJeux);

    mRecyclerView.setLayoutManager(new LinearLayoutManager(getApplicationContext()));
    mRecyclerView.setAdapter(monAdapter);
}
```

- ▶ Premièrement, je récupère la RecyclerView déclarée précédemment dans notre activity\_mail.xml via la méthode findViewById() et je l'affecte à ma variable mRecyclerView.
- ▶ J'instancie ma liste de jeux comme nouvel ArrayList<JeuVideo>.
- ▶ J'ajoute des nouveaux jeux vidéos dans ma liste en utilisant la méthode add() sur la liste en lui passant comme paramètre le nouveau jeu vidéo que je souhaite créer.
- ▶ J'instancie mon adapter en lui passant comme paramètre dans le constructeur ma liste mesJeux;
- ▶ Nous allons maintenant choisir comment nous voulons disposer notre liste: soit de manière verticale, soit de manière horizontale, soit en grille. Pour cela, nous utiliserons la méthode setLayoutManager() sur notre variable mRecyclerView. Explications.



- ▶ Les `LinearLayout` sont capable d'afficher les éléments soit de haut en bas (verticalement). Voir l'app Gmail.
- ▶ Soit de gauche à droite (horizontalement). Voir l'app Play Store.
- ▶ Les `GridLayout` affichent les éléments sous forme de grille. Voir l'app Play Musique.





- ▶ Le code utilisé pour un affichage vertical: j'utilise la méthode `setLayoutManager()` sur ma `RecyclerView` et je lui passe en paramètre un nouveau `LinearLayoutManager` qui prends en paramètre le `context`, l'orientation et un `boolean` qui indique si je souhaite inverser le sens d'affichage de la liste.
- ▶ Même signature pour un affichage vertical, nous passerons juste notre orientation de `VERTICAL` à `HORIZONTAL`.

```
mRecyclerView.setLayoutManager(new LinearLayoutManager(getApplicationContext(), LinearLayoutManager.VERTICAL, false));
```

```
mRecyclerView.setLayoutManager(new LinearLayoutManager(getApplicationContext(), LinearLayoutManager.HORIZONTAL, false));
```

- ▶ Le code utilisé pour un affichage en grille: j'utilise un nouveau `GridLayoutManager` en lui passant comme paramètre le `context` et le nombre de colonnes.

```
mRecyclerView.setLayoutManager(new GridLayoutManager(getApplicationContext(), 3));
```

- Pour terminer, j'utilise la méthode `setAdapter()` sur ma `RecyclerView` pour lier mon adapter à ma liste.

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

    mRecyclerView = (RecyclerView)findViewById(R.id.myRecyclerView);

    mesJeux = new ArrayList<>();

    mesJeux.add(new JeuVideo("Forza Horizon 3", 49.90F));
    mesJeux.add(new JeuVideo("Forza Motorsport 5", 20.00F));
    mesJeux.add(new JeuVideo("Asseto Corsa", 22.00F));
    mesJeux.add(new JeuVideo("Gran Theft Auto 5", 49.90F));

    monAdapter = new MyVideosGamesAdapter(mesJeux);

    mRecyclerView.setLayoutManager(new LinearLayoutManager(getApplicationContext()));
    mRecyclerView.setAdapter(monAdapter);
}
```

DÉVELOPPER UNE APPLICATION ANDROID NATIVE

---

À VOUS DE JOUER