Créer des écrans riches

Utilisation des RecyclerViews

Adrian Bracigliano

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Déclaration d'un RecyclerView dans un layout

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:recyclerview-v7:24.2.1'
}
```

Définition de la librairie RecyclerView dans build.gradle

RecyclerView

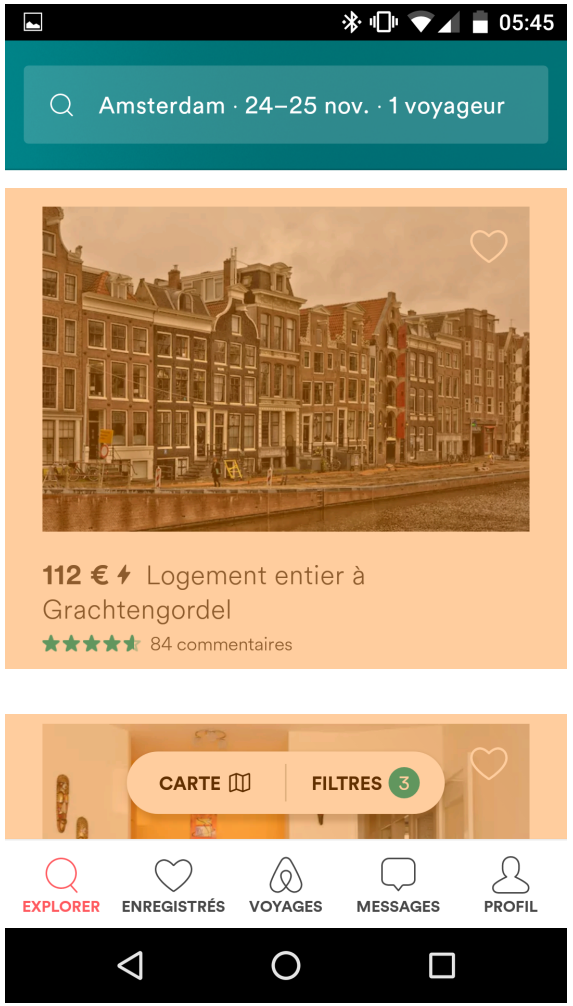
RecyclerView

- Permet d'afficher dans une vue de taille limitée un grand nombre de données du même type en se basant sur le recyclage

Affichage de logements dans Airbnb

Recyclage

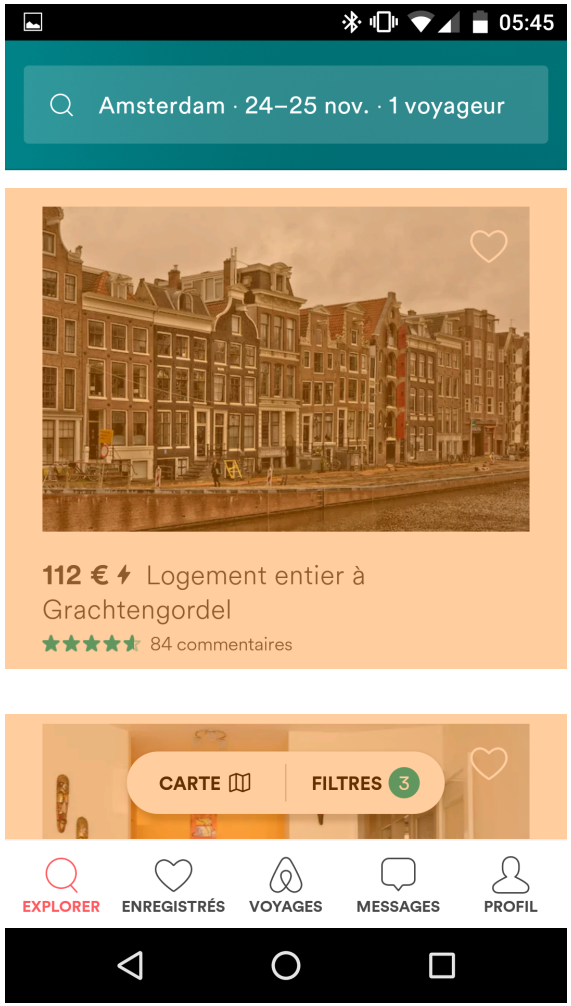
- Liste constituée d'items



Recyclage

- Liste constituée d'items
- Pour chaque item : vue à parser pour récupérer les différents élements
  - Instanciation du fichier xml
  - findViewByIds
  - …

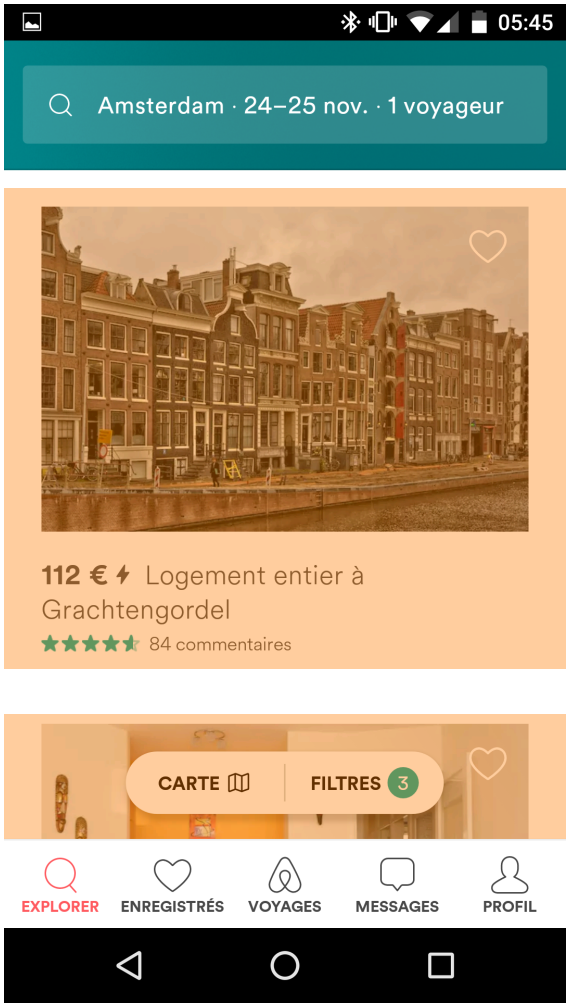
  Traitement long !

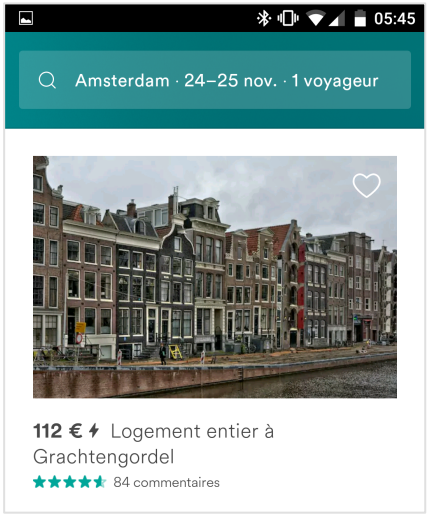


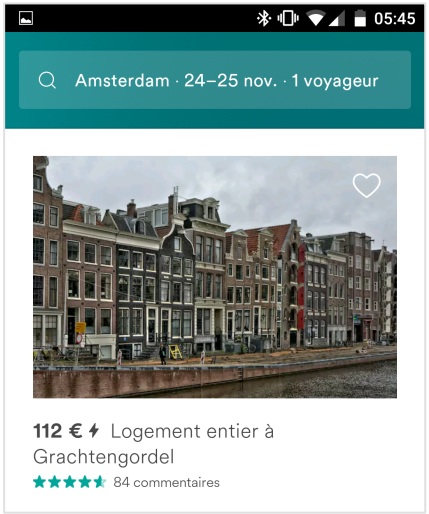
Recyclage

- Liste constituée d'items
- Pour chaque item : vue à parser pour récupérer les différents élements

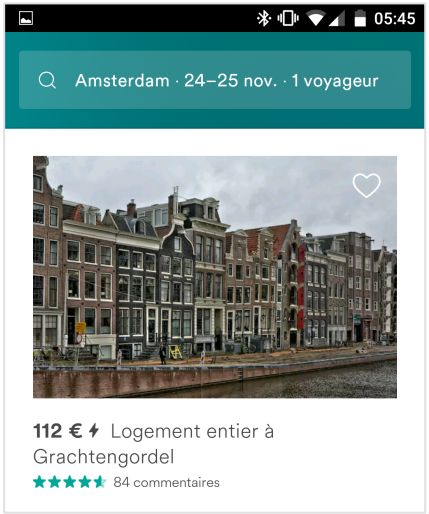
→Recyclage des anciennes vues parsées




05:45
Amsterdam · 24–25 nov. · 1 voyageur
112 € Logement entier à Grachtengordel
84 commentaires

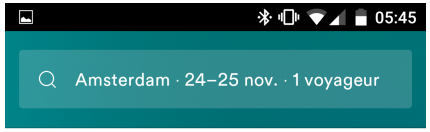

05:45
Amsterdam · 24–25 nov. · 1 voyageur
112 € Logement entier à Grachtengordel
84 commentaires

Création d’un nouveau layout pour prochain item à afficher

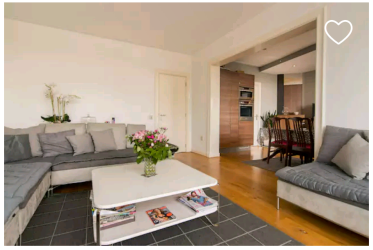


Ancien item invisible →

112 € Logement entier à Grachtengordel
84 commentaires



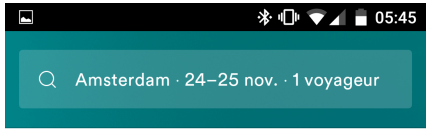
Nouvel item affiché →



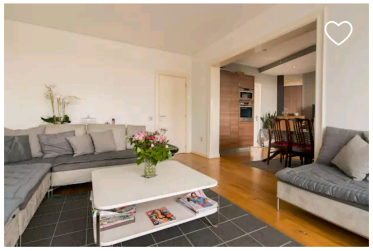
294 € Logement entier à Rivierenbuurt
104 commentaires

Ancien item invisible →

112 € Logement entier à Grachtengordel
84 commentaires



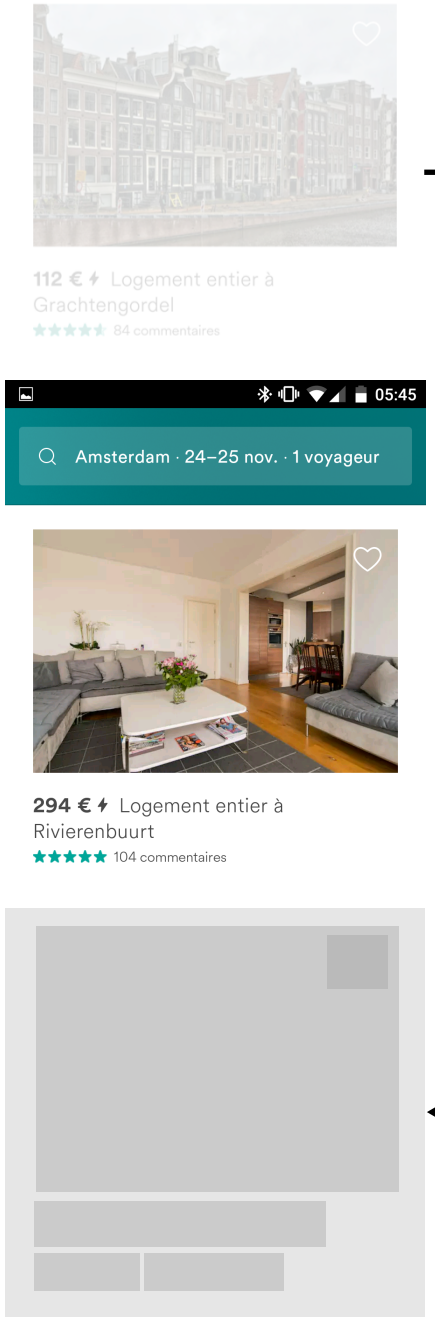
Nouvel item affiché →



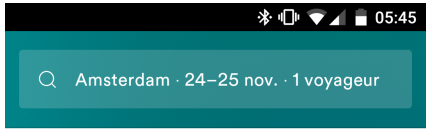
294 € Logement entier à Rivierenbuurt
104 commentaires

Nouvel item affiché

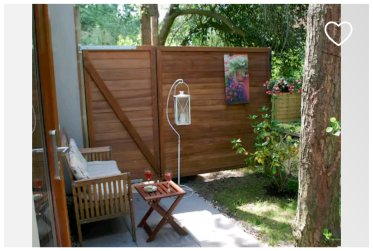
Réutilisation de la vue de l'ancien layout pour affichage du prochain item



294 € Logement entier à Rivierenbuurt
104 commentaires



Nouvel item affiché ⟶



100 € Logement entier à Buitenveldert-West
28 commentaires

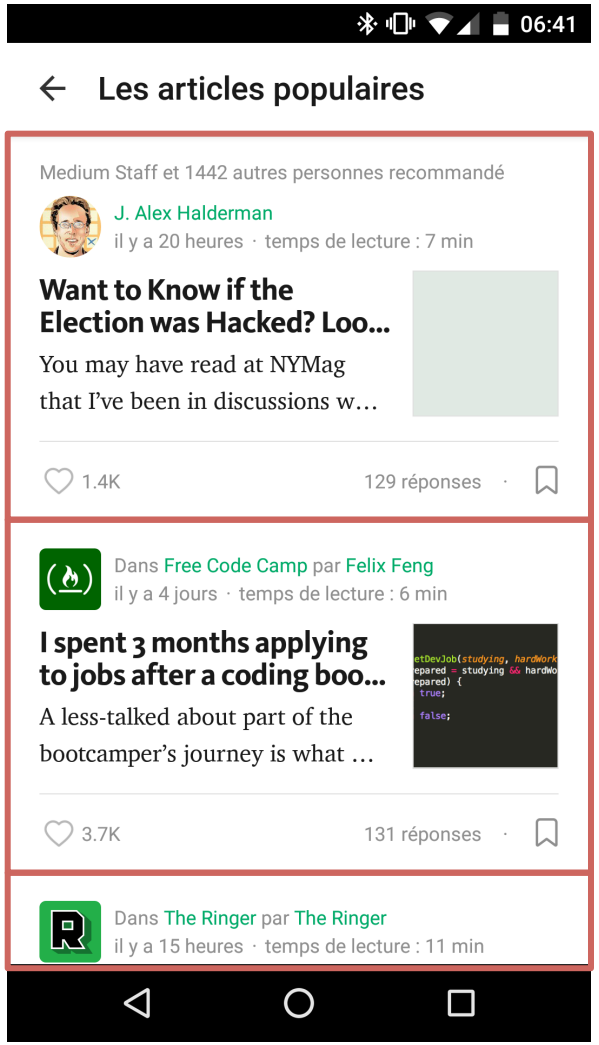
Recyclage

Liste sans recyclage

Liste avec recyclage

RecyclerView

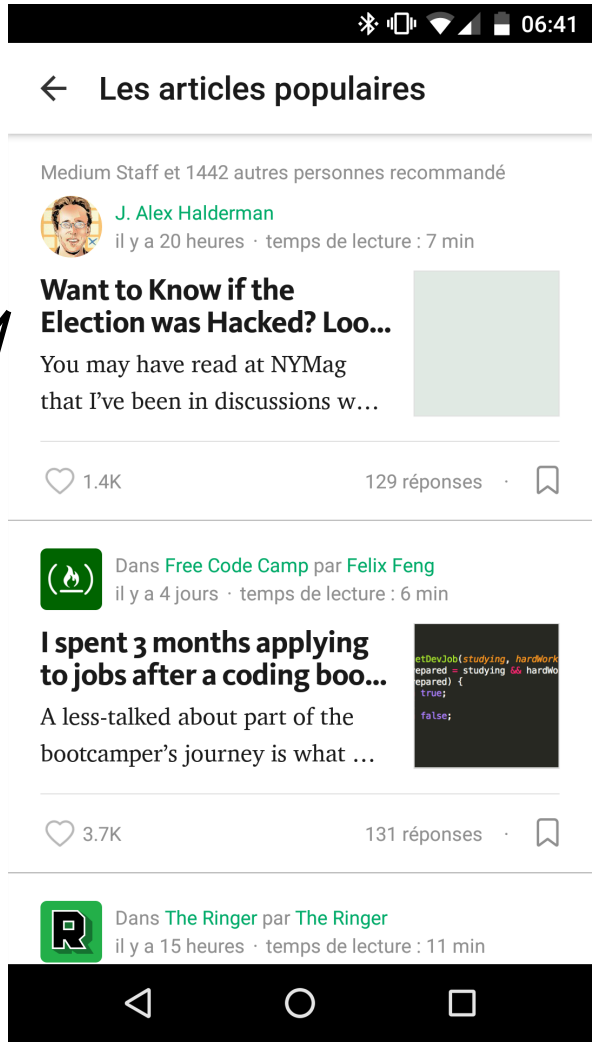
- **LayoutManager**
  - Responsable de mesurer et positionner les items
  - Détermine quand les items sont recyclés



RecyclerView

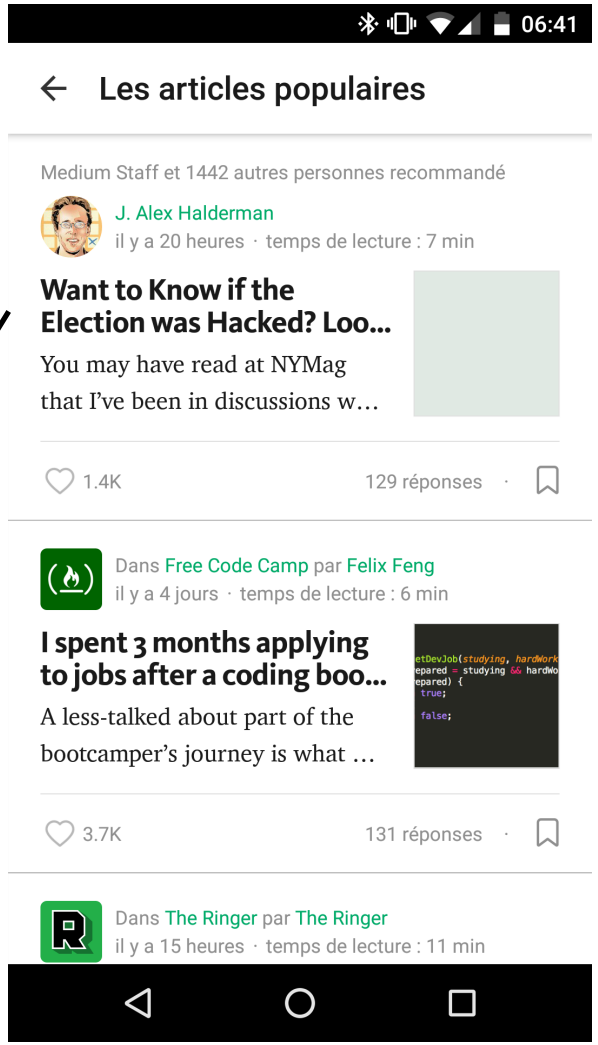
- **LayoutManager**
- **Adapter**
  - fournit les données
  - lie chaque donnée à une vue

```
Article article = new Article();
article.author = "J. Alex Halderman";
article.date = "23/11/2016";
article.time = "7 min";
article.title = "Want to know if ...";
article.content = "You may have read ...";
```

RecyclerView

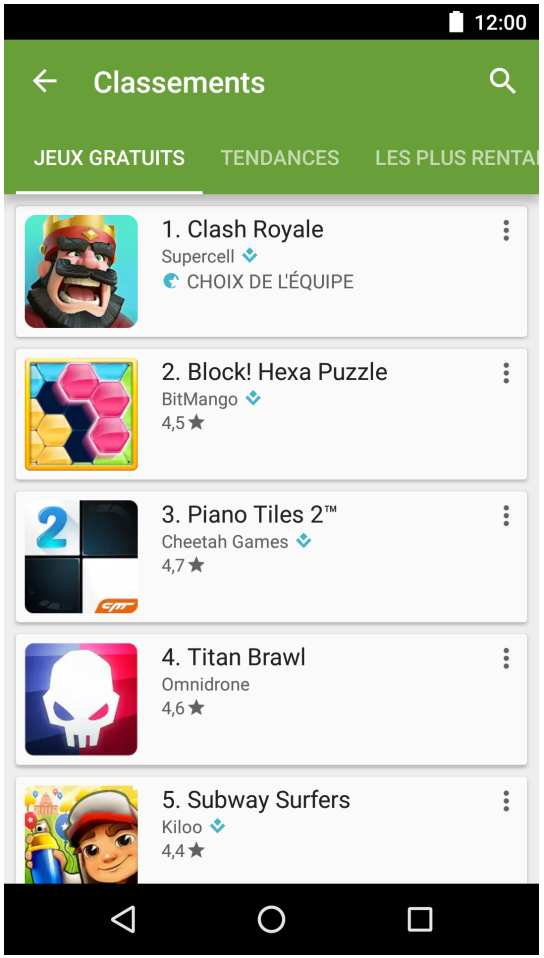
- **LayoutManager**
- **Adapter**
- **ViewHolder**
  - Définition de la vue d'un item
  - Réutilisée pour le recyclage
-



LayoutManager

LinearLayoutManager

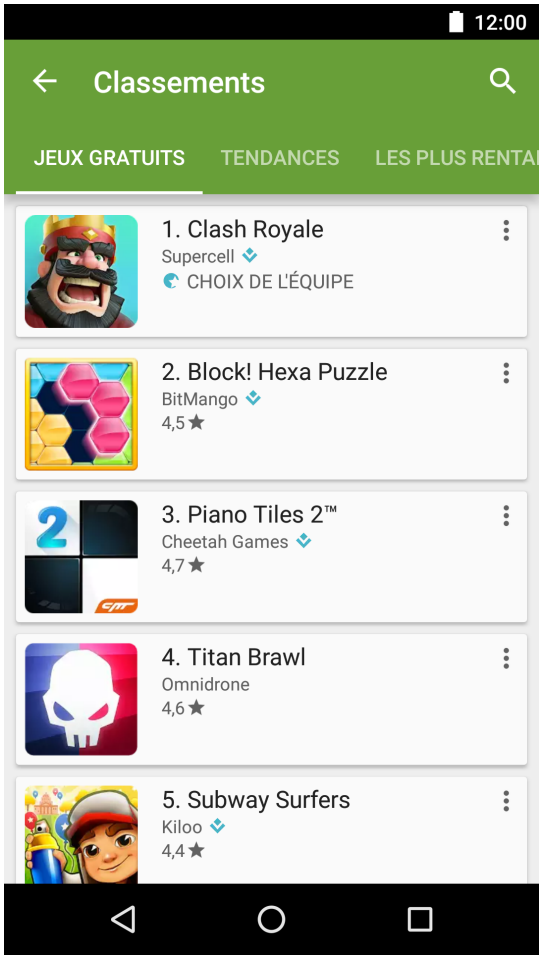
LayoutManager



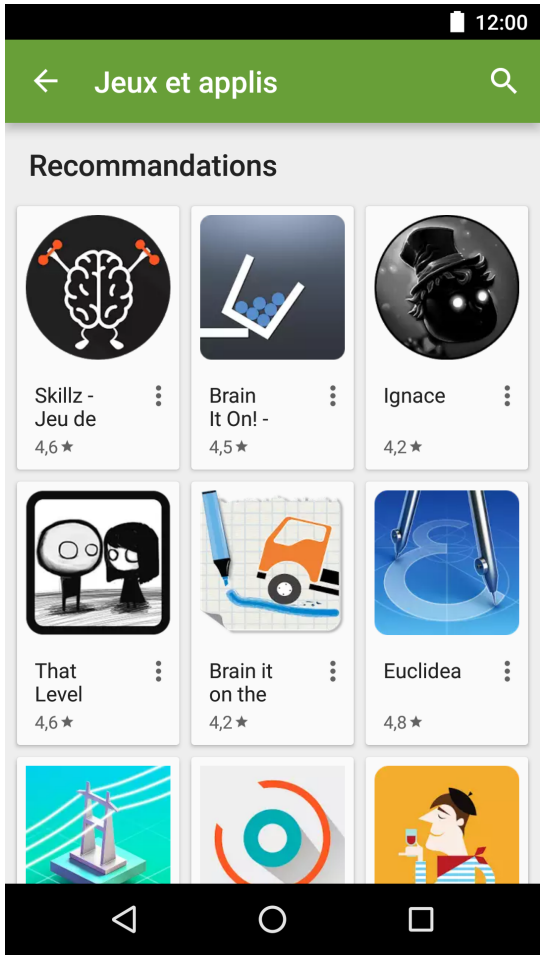
LinearLayoutManager

GridLayoutManager

LayoutManager



LinearLayoutManager



GridLayoutManager

StaggeredGridLayoutManager

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);

LinearLayoutManager layoutManager = new LinearLayoutManager(this);
recyclerView.setLayoutManager(layoutManager);
```

LinearLayoutManager vertical

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);

LinearLayoutManager layoutManager = new LinearLayoutManager(this);
layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
```

LinearLayoutManager horizontal

ViewHolder

```
public class MyViewHolder extends RecyclerView.ViewHolder {

    public TextView name;
    public TextView desc;

    public MyViewHolder(View itemView) {
        super(itemView);
        // Retrieves each field from itemView
        name = (TextView) itemView.findViewById(R.id.name);
        desc = (TextView) itemView.findViewById(R.id.desc);
    }
}
```

Adapter

```
// Called when RecyclerView needs to create a new view for
and item
@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {

    // Instanciates a view
    View view = LayoutInflater
                    .from(parent.getContext())
                    .inflate(R.layout.item, parent, false);

    // ViewHolder takes the view and retrieves each field
    return new MyViewHolder(view);
}
```

Adapter

```
// Called when RecyclerView needs to display a data at a
specific position
@Override
public void onBindViewHolder(MyViewHolder holder, int
position) {

    // Retrieves the data we want to display at position
    Data data = listOfData.get(position);

    // Binds data to the view
    holder.name.setText(data.name);
    holder.desc.setText(data.description);
}
```

Adapter

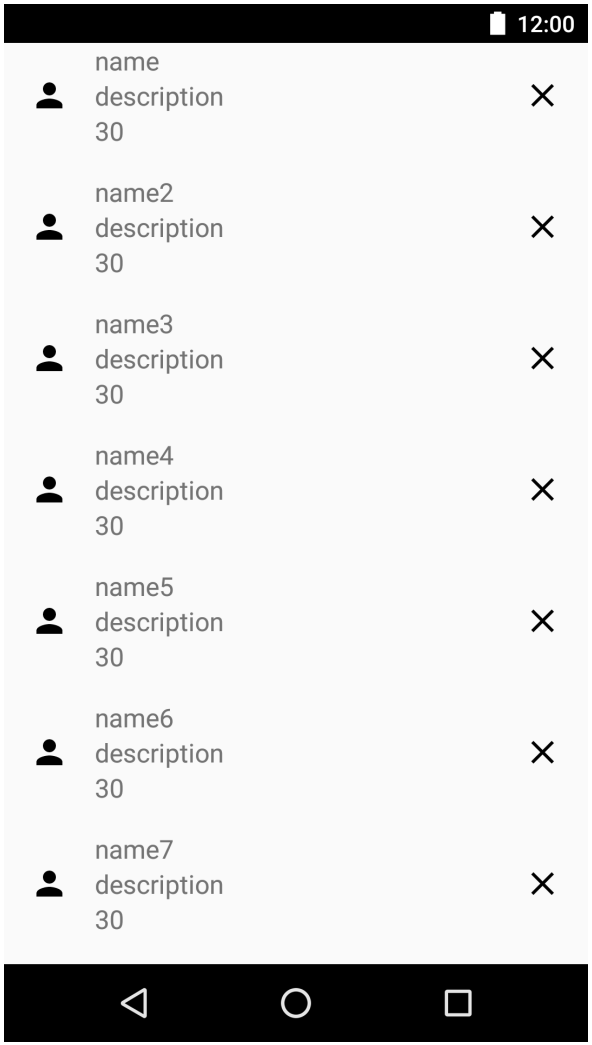
```
// Returns the number of data to display in RecyclerView
@Override
public int getItemCount() {
    return listOfData.size();
}
```

Adapter

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

    public MyAdapter(ArrayList<Data> listOfData){}

    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup p, int v) {}

    @Override
    public void onBindViewHolder(MyViewHolder h, int p) {}

    @Override
    public int getItemCount() {}

    public class MyViewHolder extends RecyclerView.ViewHolder {}
}


MyAdapter recyclerAdapter = new MyAdapter();
recyclerView.setAdapter(recyclerAdapter);
```

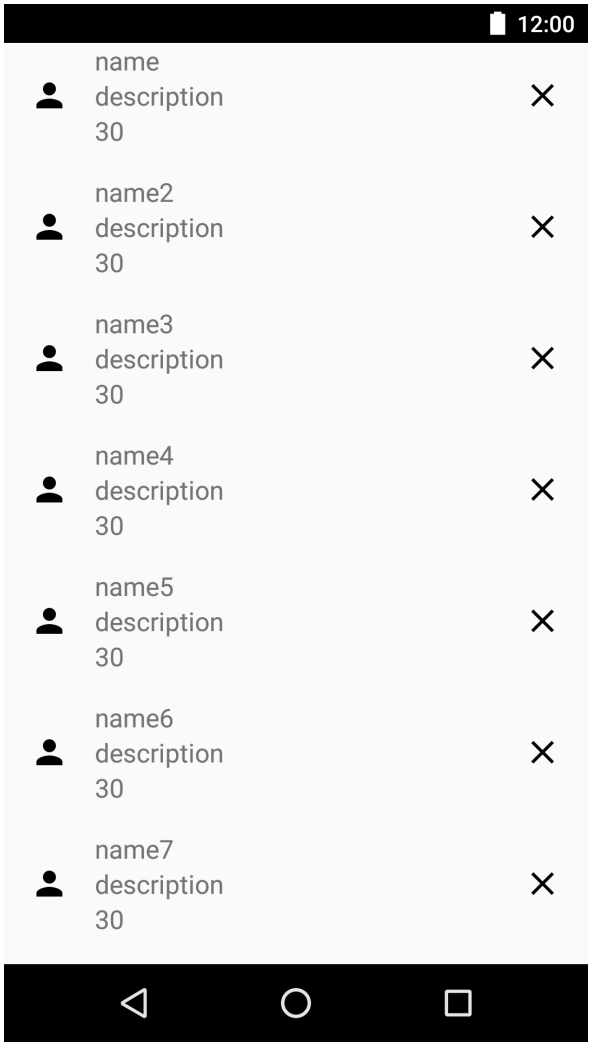
Adapter

- Création de la liste
  - Calcul du nombre d'item possible à afficher à l'écran



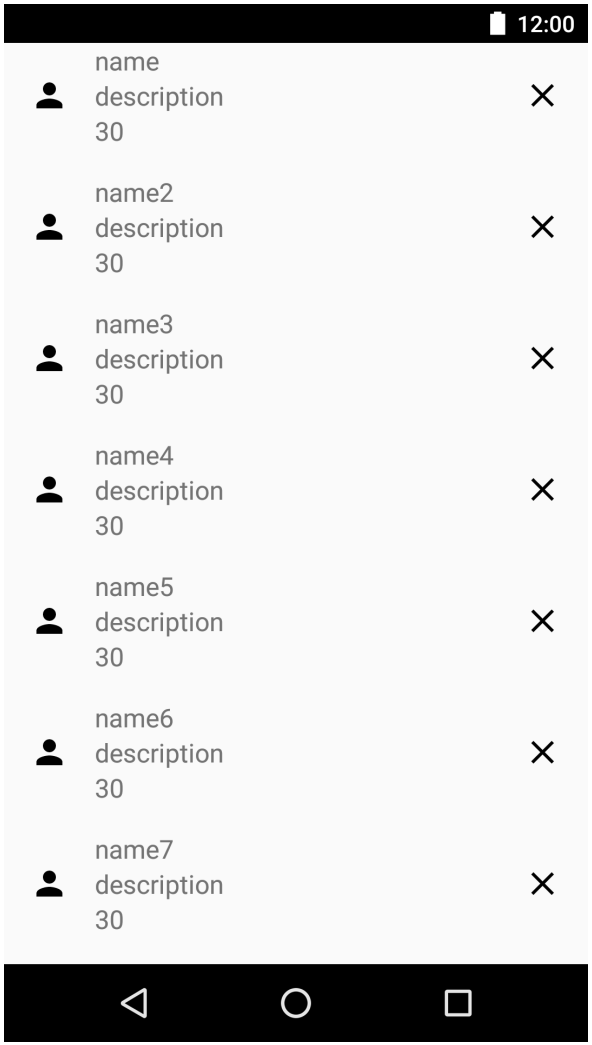
Adapter

- Création de la liste
  - Calcul du nombre d'item possible à afficher à l'écran
  - `onCreateViewHolder()` appelée pour chaque item à afficher



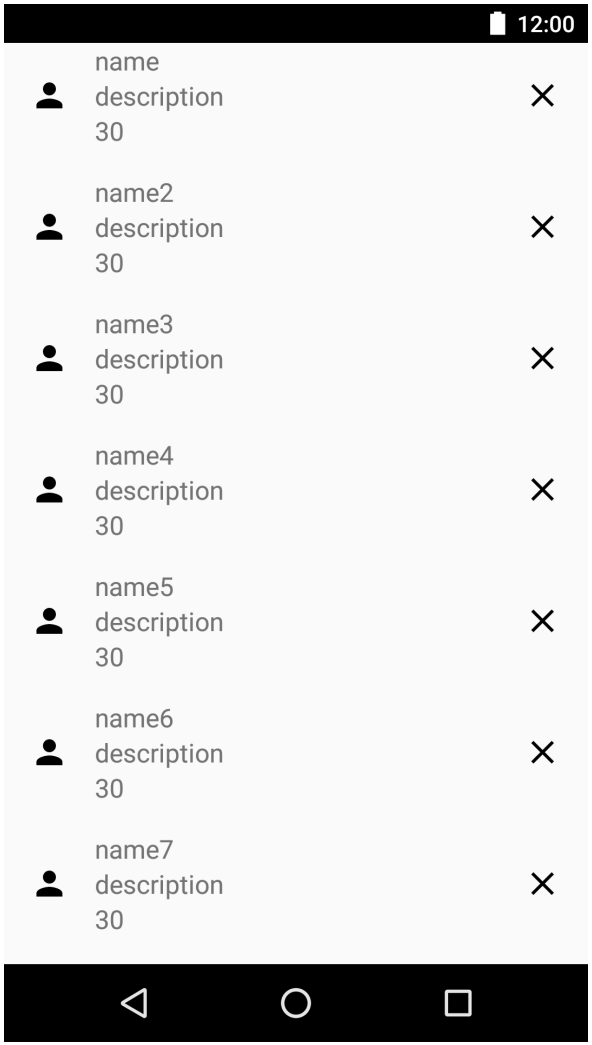
Adapter

- Création de la liste
  - Calcul du nombre d'item possible à afficher à l'écran
  - `onCreateViewHolder()` appelée pour chaque item à afficher
  - création également d'un `ViewHolder` pour chacun d'entre eux



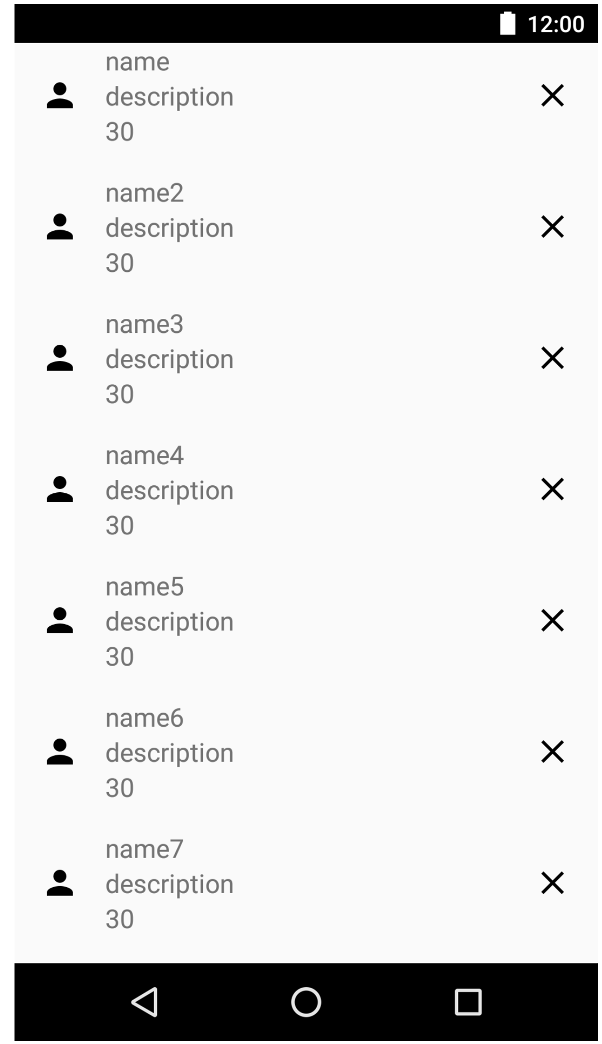
Adapter

- Création de la liste
  - Calcul du nombre d'item possible à afficher à l'écran
  - `onCreateViewHolder()` appelée pour chaque item à afficher
  - création également d'un `ViewHolder` pour chacun d'entre eux
  - appel à `onBindViewHolder` pour lier chaque item à une donnée



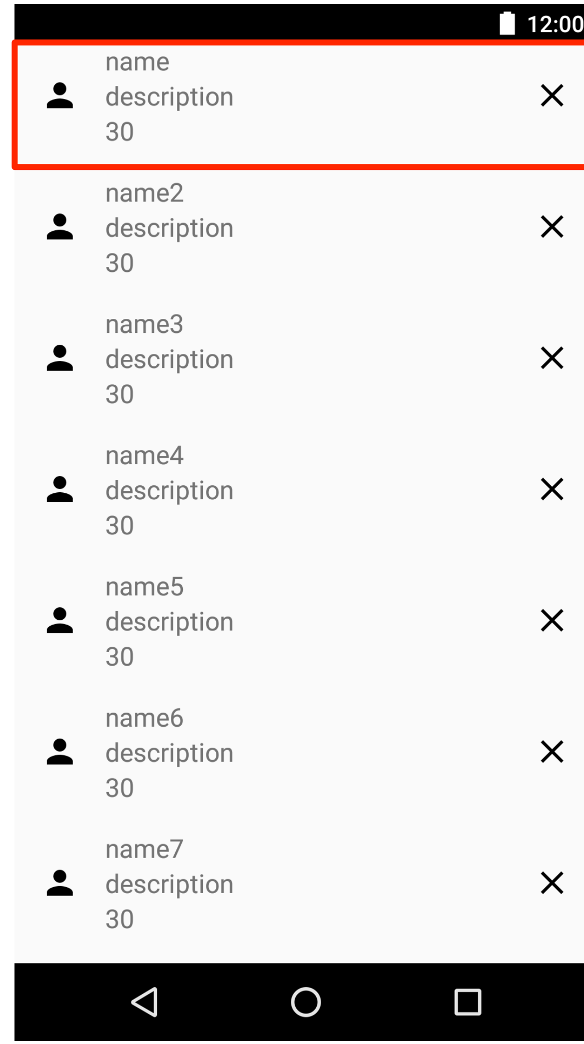
Adapter

- Création de la liste



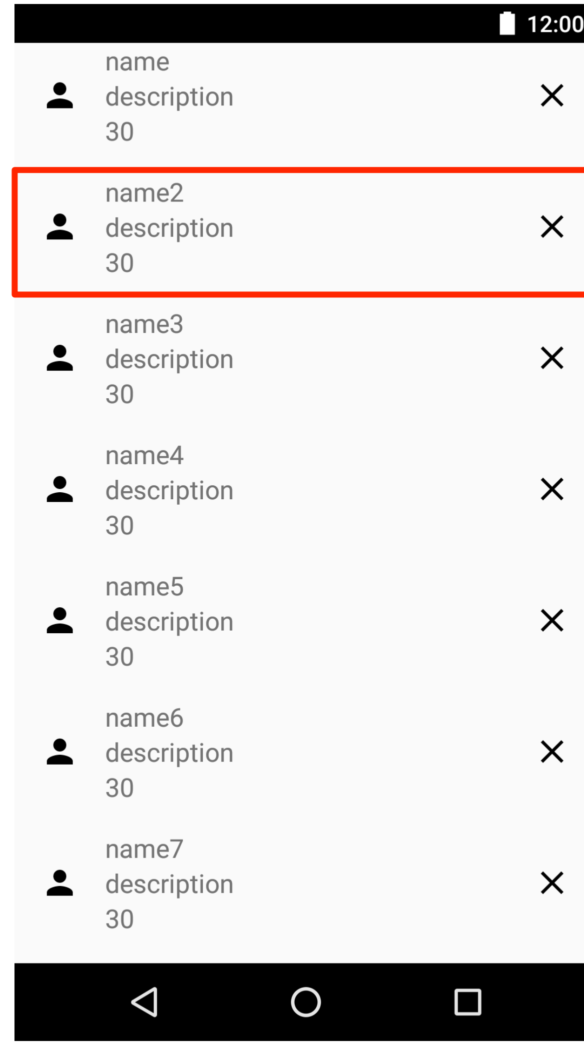
Adapter

- Création de la liste



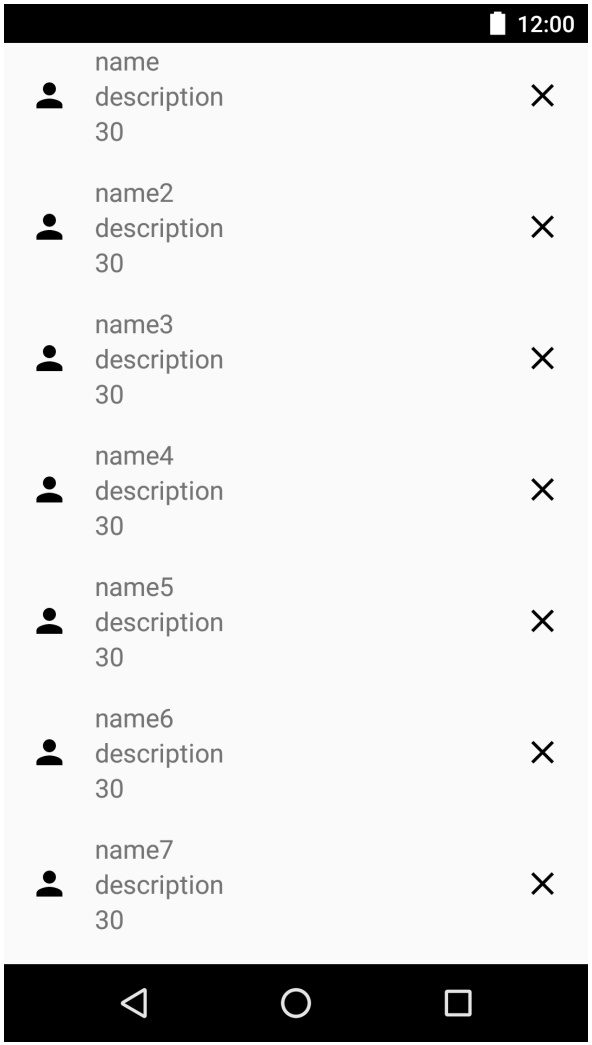
Adapter

- Création de la liste



Adapter

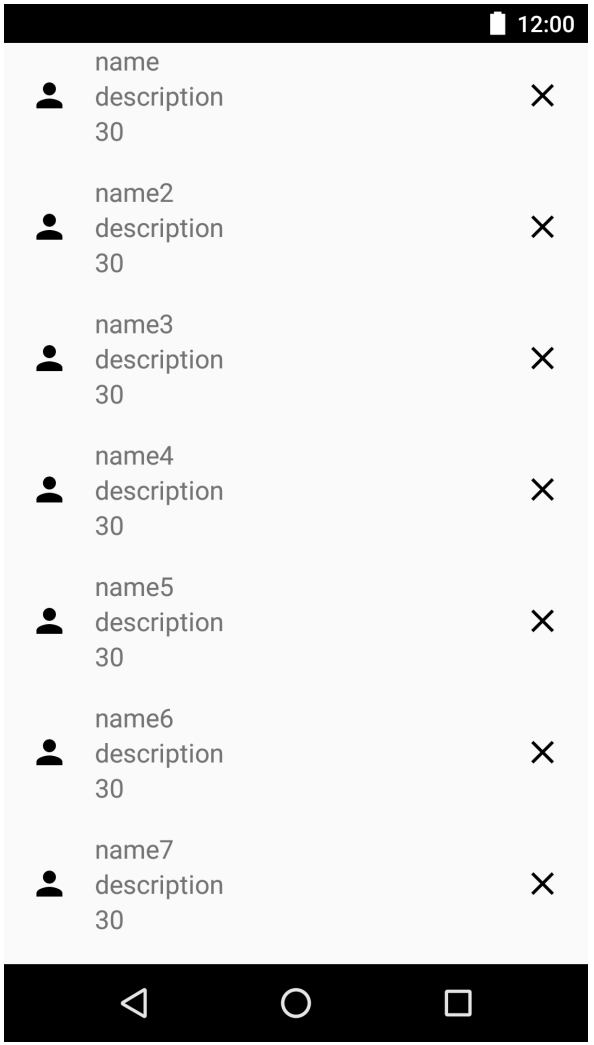
- Scroll de la liste
  - Récupération des ViewHolders plus utilisés par les items cachés
  - appel à `onBindViewHolder` pour lier chaque item aux nouvelles données à afficher



Adapter

- Scroll de la liste

```
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
com.myapplication E/RecyclerView: onBindViewHolder
```



Gestion de plusieurs ViewTypes

Gestion de plusieurs ViewTypes

```
public class ViewHolderHeader extends RecyclerView.ViewHolder {

    public TextView title;

    public ViewHolderHeader(View itemView) {
        super(itemView);
        title = (TextView) itemView.findViewById(R.id.title);
    }
}

public class ViewHolderContent extends RecyclerView.ViewHolder {

    public TextView name;
    public TextView desc;

    public ViewHolderContent(View itemView) {
        super(itemView);
        name = (TextView) itemView.findViewById(R.id.name);
        desc = (TextView) itemView.findViewById(R.id.desc);
    }
}
```

Gestion de plusieurs ViewTypes

```
private static final int TYPE_HEADER = 0;
private static final int TYPE_CONTENT = 1;

private ArrayList<Object> listOfData;

public MyAdapter(ArrayList<Object> listOfData) {
    this.listOfData = listOfData;
}

@Override
public int getItemViewType(int position) {
    if (listOfData.get(position) instanceof String) {
        return TYPE_HEADER;
    } else {
        return TYPE_CONTENT;
    }
}
```

Gestion de plusieurs ViewTypes

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
    LayoutInflater inflater = LayoutInflater.from(parent.getContext());

    if (viewType == TYPE_HEADER) {
        View view = inflater.inflate(R.layout.item_header, parent, false);
        return new ViewHolderHeader(view);
    } else {
        View view = inflater.inflate(R.layout.item_content, parent, false);
        return new ViewHolderContent(view);
    }
}
```

Gestion de plusieurs ViewTypes

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
    LayoutInflater inflater = LayoutInflater.from(parent.getContext());

    if (viewType == TYPE_HEADER) {
        View view = inflater.inflate(R.layout.item_header, parent, false);
        return new ViewHolderHeader(view);
    } else {
        View view = inflater.inflate(R.layout.item_content, parent, false);
        return new ViewHolderContent(view);
    }
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    if (holder.getItemViewType() == TYPE_HEADER) {
        fillHeader();
    } else {
        fillContent();
    }
}
```

Ajout d’un listener pour le click d’un item

```
interface MyItemClickListener {
    void onItemClickListener(int position);
}


private MyItemClickListener listener;


public void setListener(MyItemClickListener listener) {
    this.listener = listener;
}
```

Ajout d'un listener pour le click d'un item

```
interface MyItemClickListener {
    void onItemClickListener(int position);
}


private MyItemClickListener listener;


public void setListener(MyItemClickListener listener) {
    this.listener = listener;
}

public class MyViewHolder extends RecyclerView.ViewHolder {

    public TextView name;

    public MyViewHolder(View itemView) {
        super(itemView);
        name = (TextView) itemView.findViewById(R.id.name);
    }
}
```

Ajout d'un listener pour le click d'un item

```
interface MyItemClickListener {
    void onItemClickListener(int position);
}


private MyItemClickListener listener;


public void setListener(MyItemClickListener listener) {
    this.listener = listener;
}

public class MyViewHolder extends RecyclerView.ViewHolder {

    public TextView name;

    public MyViewHolder(View itemView) {
        super(itemView);
        name = (TextView) itemView.findViewById(R.id.name);
        itemView.setOnClickListener(new View.OnClickListener() {
               @Override
               public void onClick(View v) {
                    // Returns the position of the item that was clicked
                    listener.onItemClickListener(getAdapterPosition());
               }
          });
     }
}
```

Ajout d’un listener pour le click d’un item

```
MyAdapter recyclerAdapter = new MyAdapter();
recyclerAdapter.setListener(new MyAdapter.MyItemClickListener() {
    @Override
    public void onItemClickListener(int position) {
        // Handle click

    }
});
recyclerView.setAdapter(recyclerAdapter);
```