

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

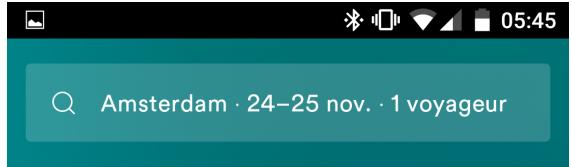
- Permet d'afficher dans une vue de taille limitée un grand nombre de données du même type en se basant sur le recyclage



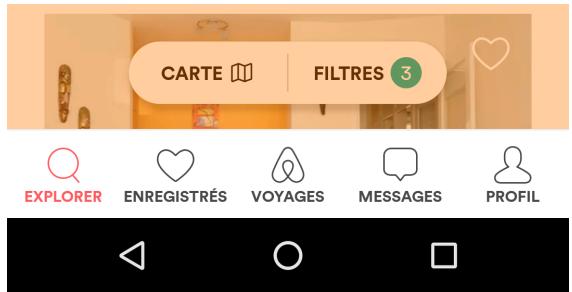
Affichage de logements dans
Airbnb

Recyclage

- Liste constituée d'items



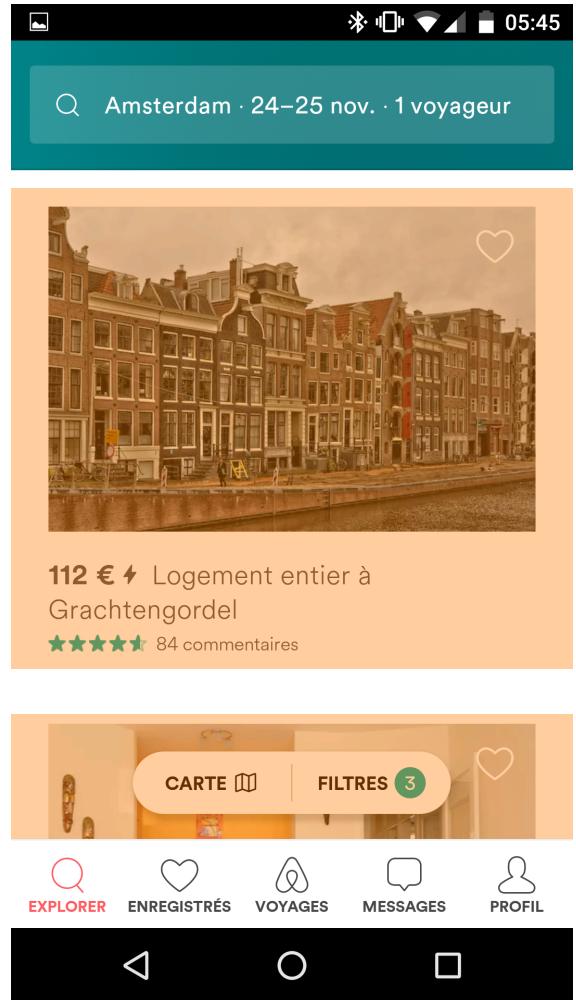
112 € ⚡ Logement entier à
Grachtengordel
★★★★★ 84 commentaires



Recyclage

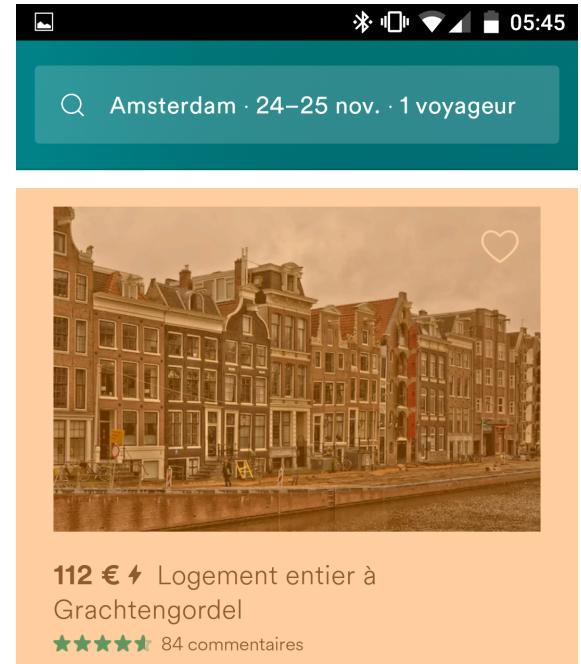
- Liste constituée d'items
- Pour chaque item : vue à parser pour récupérer les différents éléments
 - Instantiation du fichier xml
 - findViewByIds
 - ...

Traitement long !

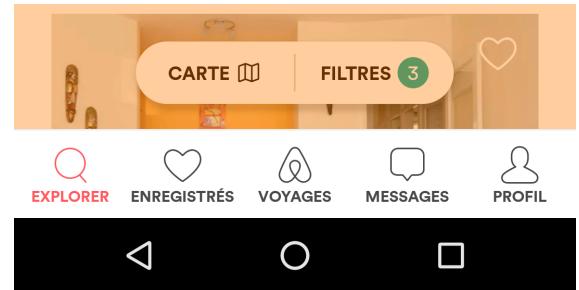


Recyclage

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



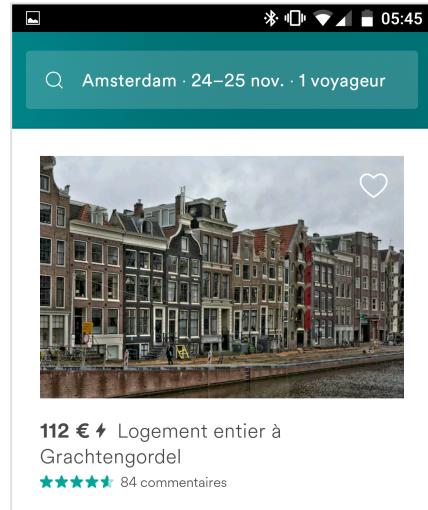
→ Recyclage des anciennes vues parsées



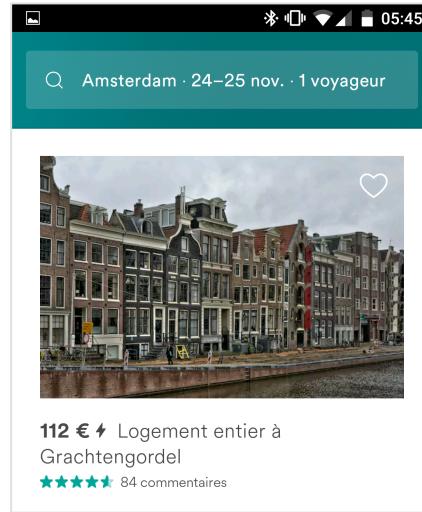
05:45

Q Amsterdam · 24–25 nov. · 1 voyageur

112 € ⚡ Logement entier à
Grachtengordel
★★★★★ 84 commentaires



Création d'un
nouveau layout
pour prochain item
à afficher



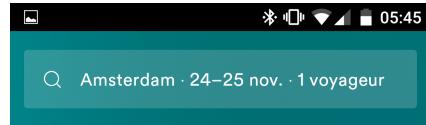
Ancien item invisible →



112 € ⚡ Logement entier à

Grachtengordel

★★★★★ 84 commentaires



Nouvel item affiché →



294 € ⚡ Logement entier à

Rivierenbuurt

★★★★★ 104 commentaires

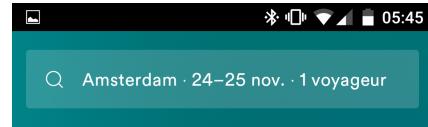
Ancien item invisible →



112 € ⚡ Logement entier à

Grachtengordel

★★★★★ 84 commentaires



Nouvel item affiché →



294 € ⚡ Logement entier à

Rivierenbuurt

★★★★★ 104 commentaires



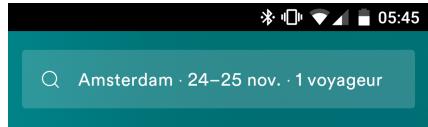
Nouvel item affiché →





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

The screenshot shows a mobile application interface with a dark header bar at the top containing icons for Bluetooth, signal strength, battery level, and the time (06:41). Below the header is a navigation bar with a back arrow and the text "Les articles populaires". The main content area displays a list of news articles in a vertical RecyclerView. Each article card includes a small profile picture, the author's name, a brief description, and engagement metrics (likes, responses, and a bookmark icon).

- Medium Staff et 1442 autres personnes recommandé
J. Alex Halderman
il y a 20 heures · temps de lecture : 7 min

Want to Know if the Election was Hacked? Loo...

You may have read at NYMag
that I've been in discussions w...

1.4K likes · 129 réponses ·
- (🔥) Dans Free Code Camp par Felix Feng
il y a 4 jours · temps de lecture : 6 min

I spent 3 months applying to jobs after a coding boo...

A less-talked about part of the bootcamper's journey is what ...

etDevJob(studying, hardWork){
 prepared = studying && hardWork;
 return prepared;
}
false;

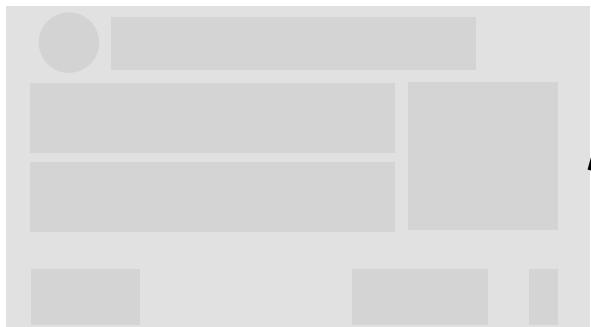
3.7K likes · 131 réponses ·
- R Dans The Ringer par The Ringer
il y a 15 heures · temps de lecture : 11 min

At the bottom of the screen are standard Android navigation icons: a triangle pointing left, a circle, and a square.

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 ...";
```



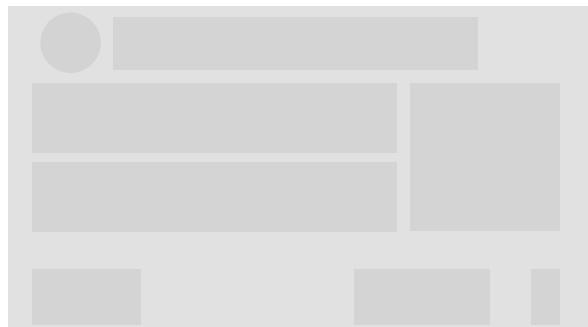
The screenshot shows a list of articles on a mobile device. At the top, there is a navigation bar with icons for signal strength, battery level, and time (06:41). Below the navigation bar is a header with a back arrow and the text "Les articles populaires". The main content area displays three articles:

- Medium Staff et 1442 autres personnes recommandé**
J. Alex Halderman
il y a 20 heures · temps de lecture : 7 min
Want to Know if the Election was Hacked? Loo...
You may have read at NYMag
that I've been in discussions w...
- Dans Free Code Camp par Felix Feng**
il y a 4 jours · temps de lecture : 6 min
I spent 3 months applying to jobs after a coding boo...
A less-talked about part of the bootcamper's journey is what ...
etDevJob(studying, hardWork
epared = studying && hardWork
epared) {
 true;
 false;
- Dans The Ringer par The Ringer**
il y a 15 heures · temps de lecture : 11 min

At the bottom of the screen, there are standard Android navigation buttons: a triangle pointing left, a circle, and a square.

RecyclerView

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



The screenshot shows a list of articles on a mobile device. At the top, there's a navigation bar with icons for signal, battery, and time (06:41). Below it, the title "Les articles populaires" is displayed with a back arrow. The first article is by "Medium Staff et 1442 autres personnes recommandé". It features a profile picture of "J. Alex Halderman", a timestamp ("il y a 20 heures"), and a reading time ("temps de lecture : 7 min"). The title of the article is "Want to Know if the Election was Hacked? Loo...". Below the title, a snippet reads: "You may have read at NYMag that I've been in discussions w...". Below the snippet are engagement metrics: 1.4K likes and 129 responses. A bookmark icon is also present. The second article, by "Felix Feng" from "Free Code Camp", is titled "I spent 3 months applying to jobs after a coding boo...". It includes a snippet of code:

```
etDevJob(studying, hardWork  
epared = studying && hardWork  
epared) {  
    true;  
}  
false;
```

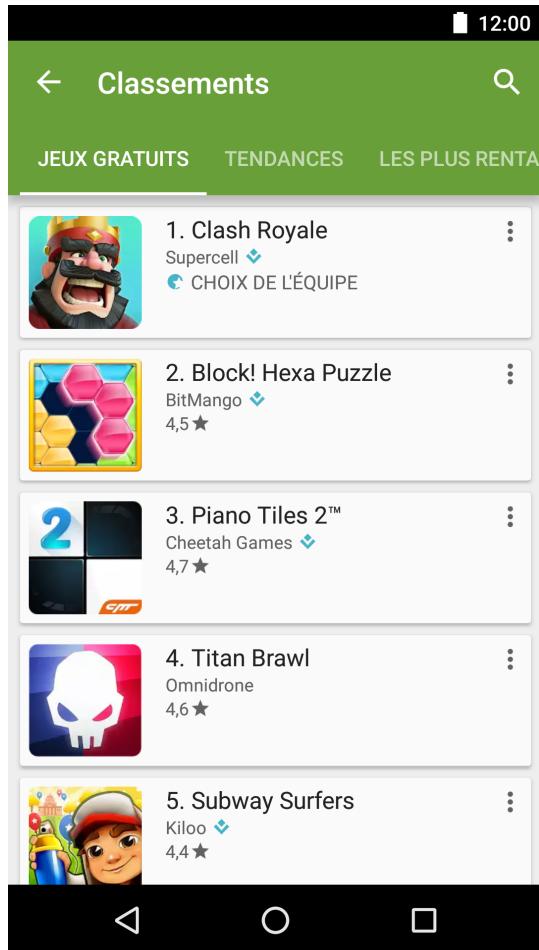
. It has 3.7K likes and 131 responses. The third article, by "The Ringer" from "The Ringer", is partially visible. The bottom of the screen shows standard Android navigation buttons: back, home, and recent apps.

LayoutManager



LinearLayoutManager

LayoutManager

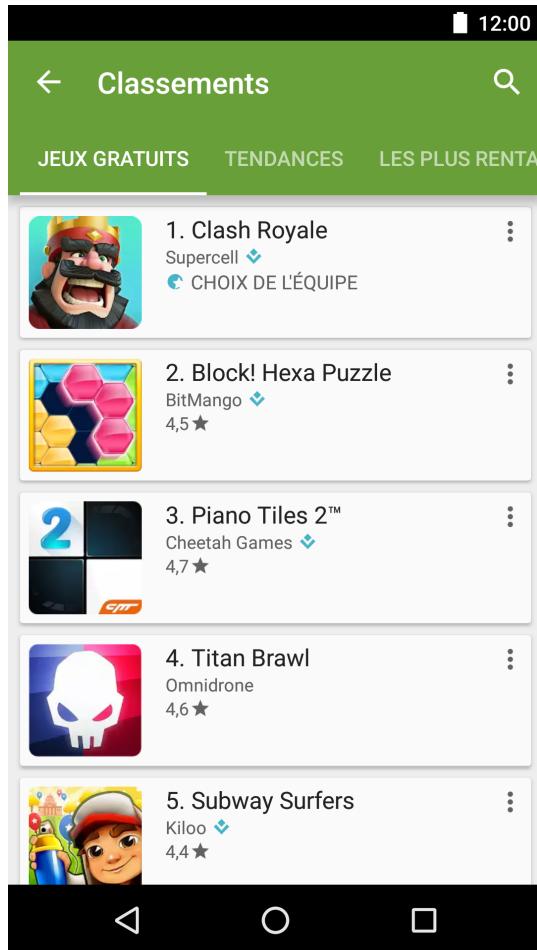


LinearLayoutManager

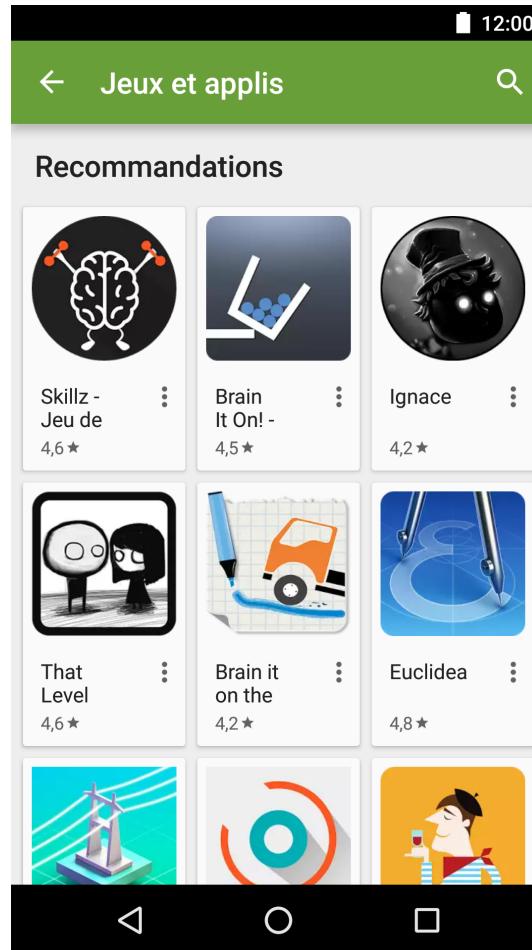


GridLayoutManager

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> list0fData){}  
  
    @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);
```

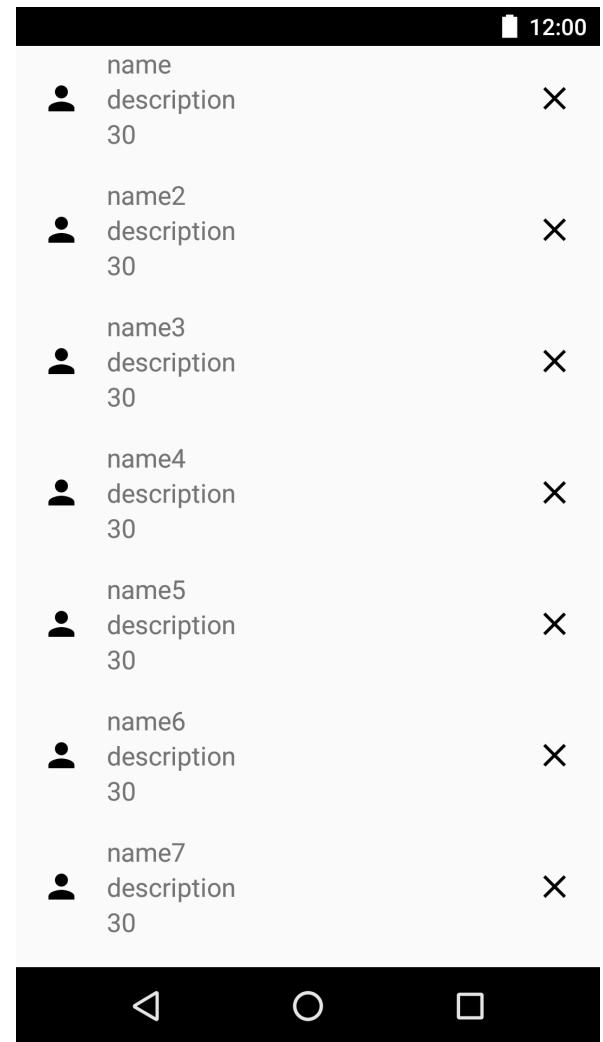
Adapter

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

12:00		
name		x
● description		
30		
name2		x
● description		
30		
name3		x
● description		
30		
name4		x
● description		
30		
name5		x
● description		
30		
name6		x
● description		
30		
name7		x
● description		
30		

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

	name	
1	name1	X
2	name2	X
3	name3	X
4	name4	X
5	name5	X
6	name6	X
7	name7	X

Adapter

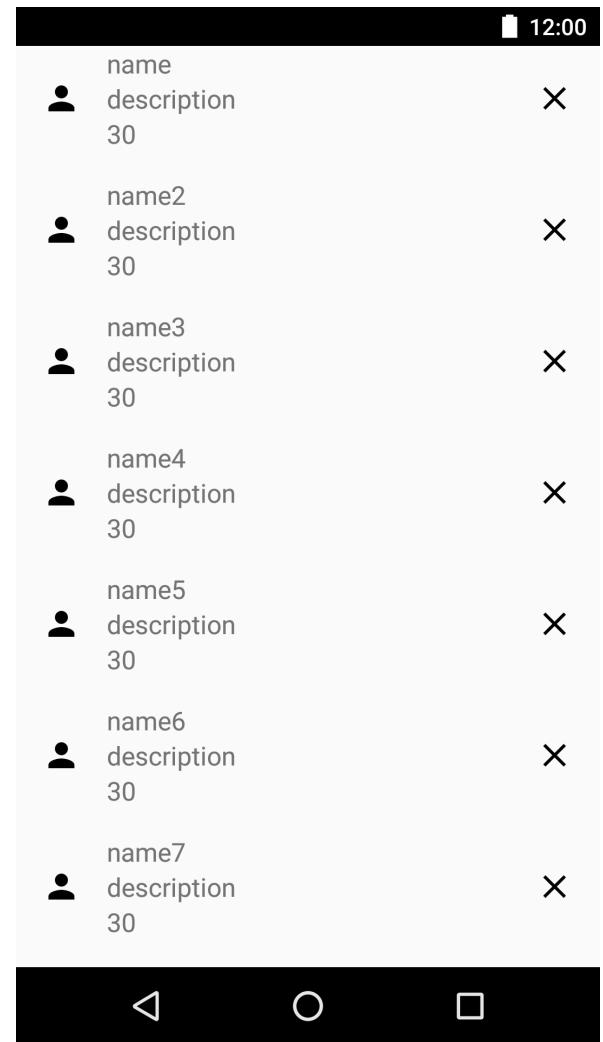
- Création de la liste
 - Calcul du nombre d'item possible à afficher à l'écran
 - onCreateViewHolder() appelée pour chaque item à afficher
 - création également d'un ViewHolder pour chacun d'entre eux
 - appel à onBindViewHolder pour lier chaque item à une donnée

	name	
1	name1	X
2	name2	X
3	name3	X
4	name4	X
5	name5	X
6	name6	X
7	name7	X

Adapter

- Crédit de la liste

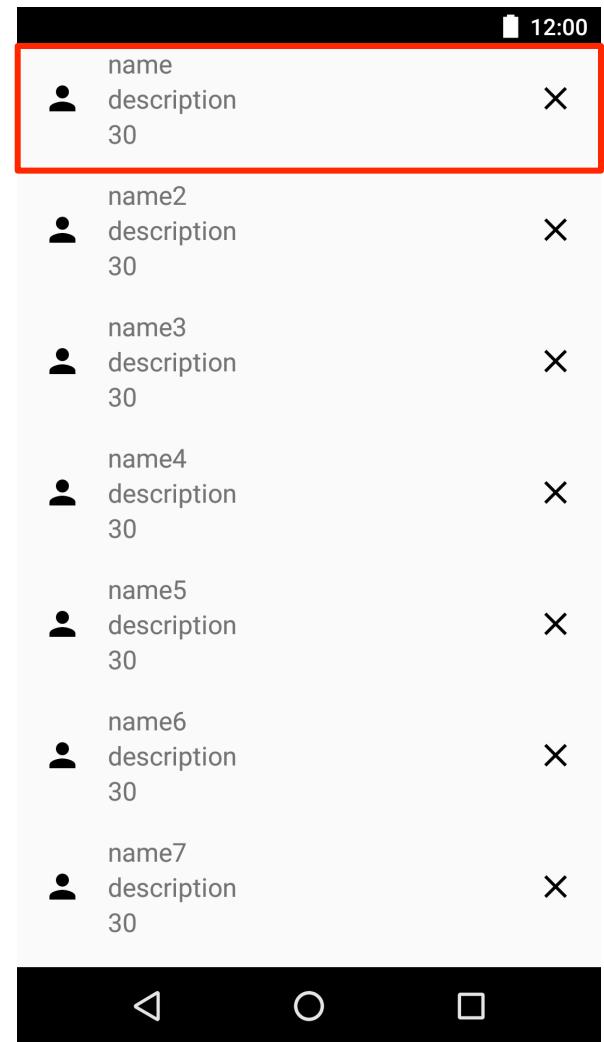
```
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder
```



Adapter

- Création de la liste

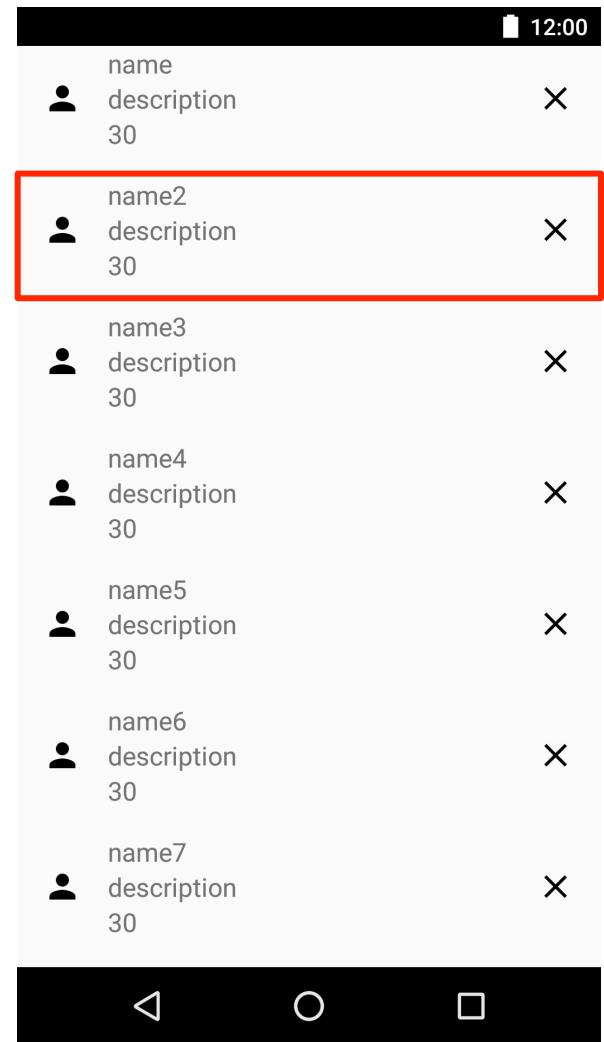
```
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder
```



Adapter

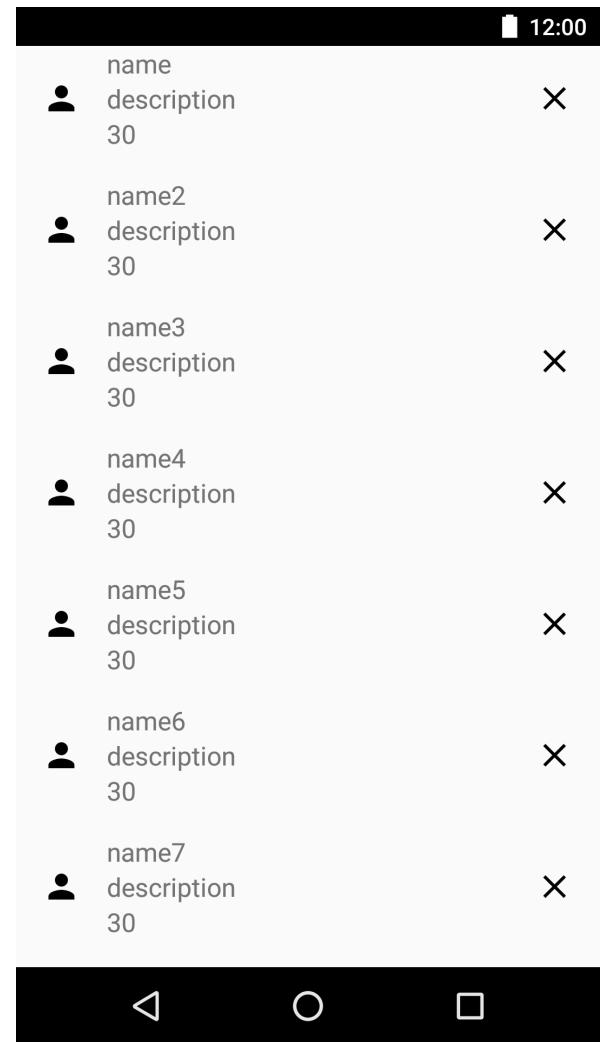
- Création de la liste

```
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder  
com.myapplication I/RecyclerView: onCreateViewHolder  
com.myapplication D/RecyclerView: new MyViewHolder()  
com.myapplication E/RecyclerView: onBindViewHolder
```



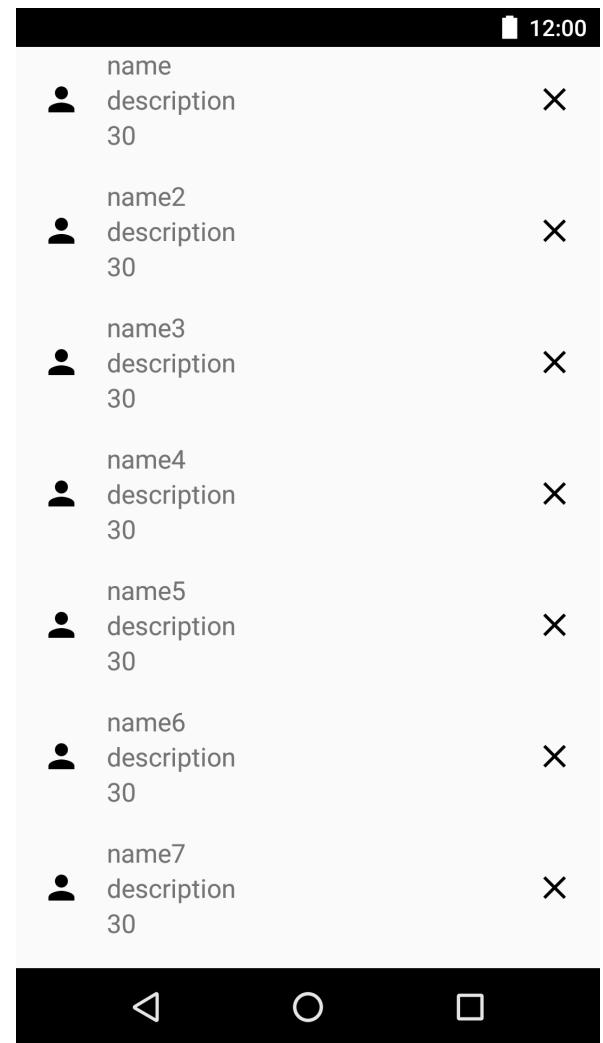
Adapter

- 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



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> list0fData;

public MyAdapter(ArrayList<Object> list0fData) {
    this.list0fData = list0fData;
}

@Override
public int getItemViewType(int position) {
    if (list0fData.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) {  
    LayoutInflator inflater = LayoutInflator.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);
```