TD5 - Les Listes

Il existe différents ViewGroup Android pour l’affichage d’une liste d’éléments.

* ListView
* GridView
* RecyclerView

Dans ce TP nous aborderons en détail l’utilisation des RecyclerView. En effet, les Recyclerview ont été créés pour améliorer ses deux premiers composants qui sont moins flexibles. L’utilisation de ces deux ViewGroup est donc à remplacer par un RecyclerView qui est un standard Android. C’est pour cela que nous ne les aborderons pas dans ce TP.

NB : L’utilisation des RecyclerView est plus complexe mais bien plus puissant lorsque l’on veut implémenter différentes vues pour un même objet par exemple.

Les composants d’un RecyclerView

## LayoutManager

Un RecyclerView doit avoir un Layout Manager (gestionnaire de disposition) et un Adapteur (adaptateur) pour être instancié. Un Layout Manager positionne les vues d'élément dans un RecyclerView et détermine quand réutiliser les vues d'élément qui ne sont plus visibles par l'utilisateur.

Le RecyclerView fournit ces Linearlayout intégrés:

* LinearLayoutManager affiche les éléments dans une liste déroulante verticale ou horizontale.
* GridLayoutManager affiche les éléments dans une grille.
* StaggeredGridLayoutManager affiche les éléments dans une grille échelonnée.

## RecyclerView.Adapter

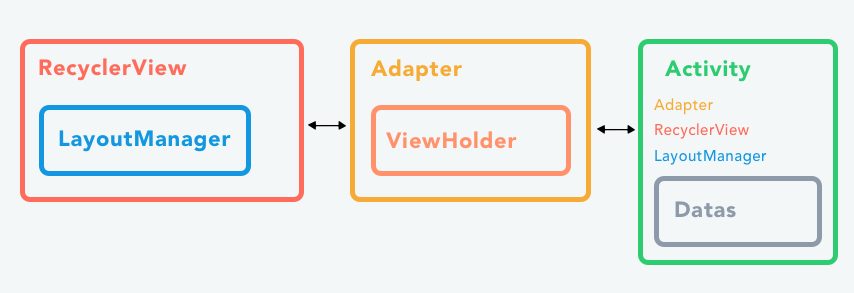
Ce dernier permet de faire la liaison entre le vue et la liste de donnés à afficher.

## ViewHolder

Permet de représenter visuellement un élément de la liste de données dans le RecyclerView (Une ligne).

L'utilisation d'un RecyclerView comporte les étapes clés suivantes:

1. Ajouter la bibliothèque RecyclerView au fichier build.Gradle
2. Définir une classe de modèle à utiliser comme source de données
3. Ajoutez un RecyclerView à votre activité pour afficher les éléments
4. Créer un fichier XML de mise en page personnalisé pour visualiser l'élément
5. Créer un RecyclerView.Adapter et ViewHolder pour fournir les élément



# Implémentons notre premier RecyclerView

* Créer un nouveau projet Android nommé le TP5.
* Ajouter une Empty Activity nommé la ContactActivity
* Ajouter la dépendance suivante dans votre build.gradle (ATTENTION le premier est à utiliser si vous avez déjà des dépendances androidx sinon utilisez le second)

: implementation("androidx.recyclerview:recyclerview:1.2.1") ou

implementation("com.android.support:recyclerview-v7:28.0.0")

## Définir un modèle pour votre donnée

Chaque RecyclerView utilise une unique source de données. Dans ce TP, nous définirons une classe Contact qui représente le modèle de données affiché par le RecyclerView:

* Implémenter une nouvelle classe Contact avec comme attribut un nom(String), un prénom(String), un imageUrl(String).
* remplacer le contenu du xml de votre activité par :

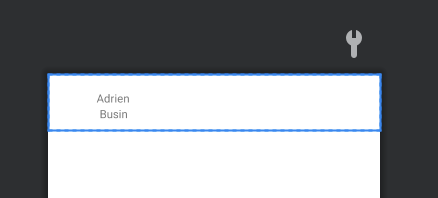
| <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/rvContacts"  android:layout\_width="0dp"  android:layout\_height="0dp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  </androidx.constraintlayout.widget.ConstraintLayout> |
| --- |

Le RecyclerView est maintenant intégré dans notre fichier de mise en page.

Ensuite, nous pouvons définir la mise en page de chaque élément de notre liste.

## Création d’un fichier de mise en page (XML) pour un élément de la liste

Créer un nouveau fichier → res/layout/item\_contact.xml



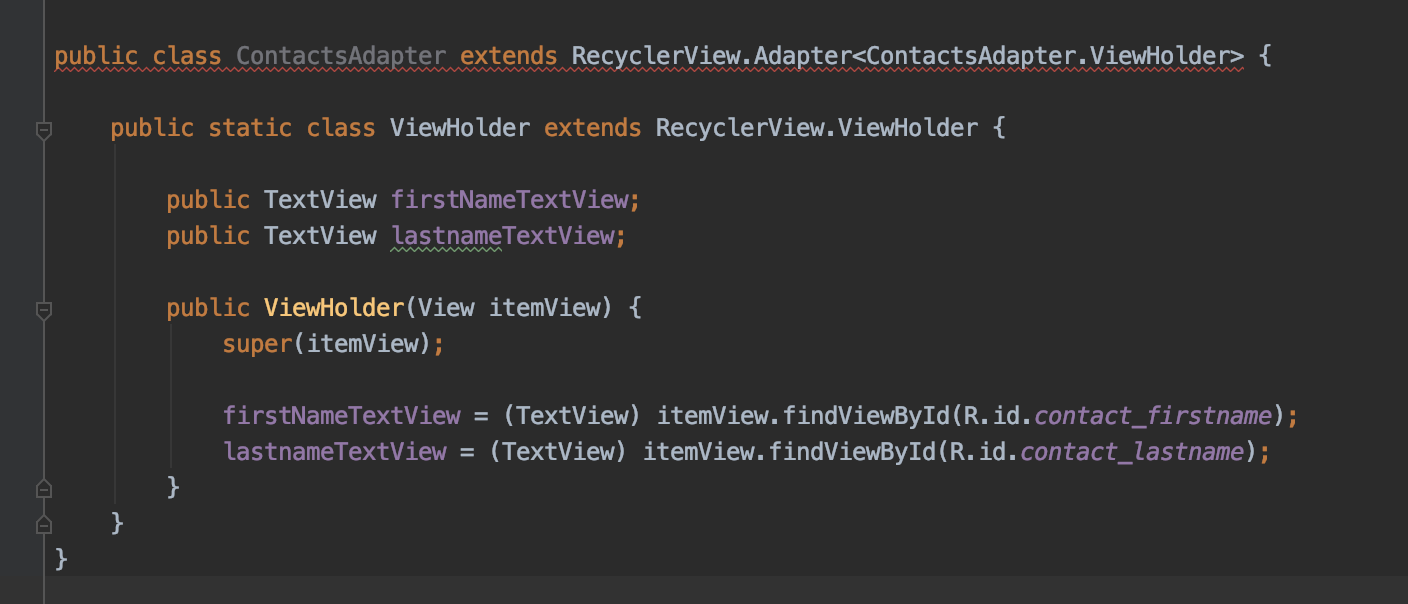
Faites en sorte à l’aide de vos connaissances d’implémenter cette interface graphique comme sur l’image ci-dessus. Deux TextView + 1 ImageView qui seront rempli par les données d’un contact dans notre liste.

Une fois le layout implémenté, nous allons créer l'adaptateur pour remplir les données dans le RecyclerView.

## Création du RecyclerView.Adapter

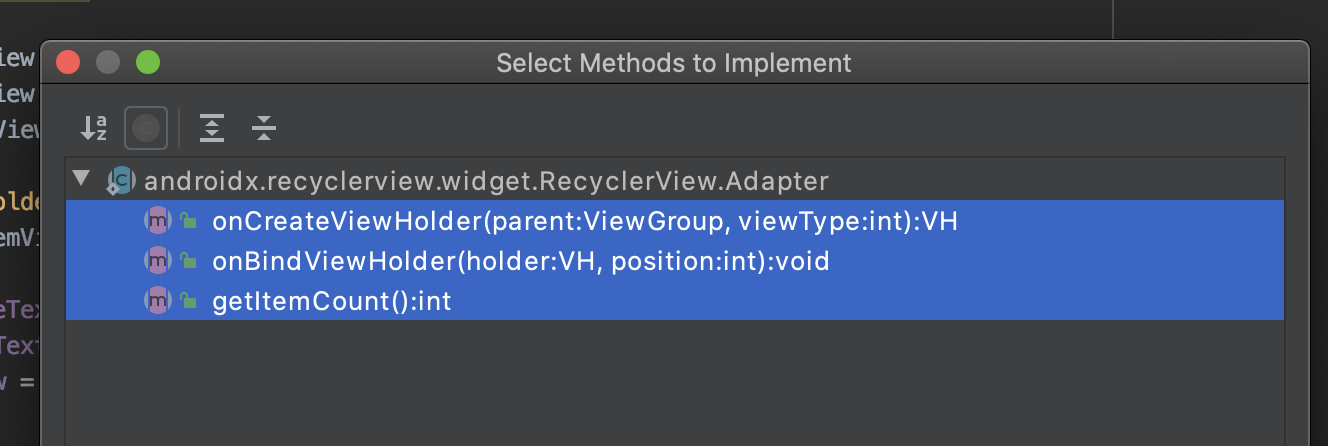
Ici, nous devons créer l'adaptateur qui remplira réellement les données dans le RecyclerView. Le rôle de l'adaptateur est de convertir un objet à une position en un élément de ligne de liste à insérer.

Cependant, avec un RecyclerView, l'adaptateur nécessite l'existence d'un objet "ViewHolder" qui décrit et donne accès à toutes les vues de chaque ligne d'élément. Nous pouvons créer l'adaptateur et le view holder dans ContactsAdapter.java comme suit :



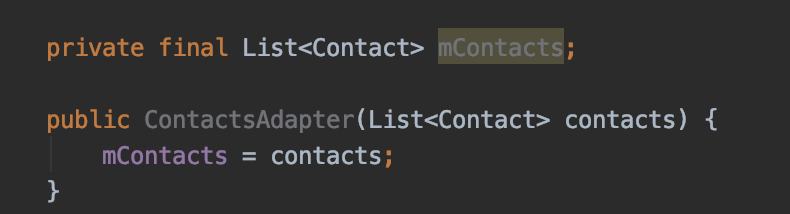
L’IDE affichera alors une erreur sur la classe ContactsAdapter. En effet, chaque Adapter implémente 3 méthodes.

Pour cela, appuyez sur la petite icône en rouge puis ‘impléments méthods’ sélectionnez les 3 méthodes suivantes :



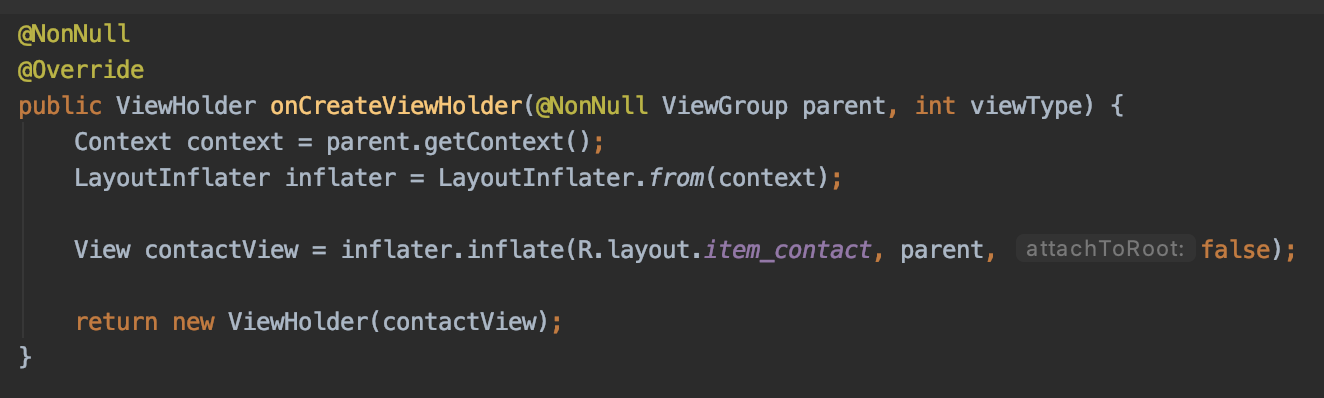
* onCreateViewHolder utilisé pour déclarer le xml de la vue d’un item
* onBindViewHolder utilisé pour afficher les données passées en paramètres de l’adapter
* getItemCount utilisé pour déterminer le nombre d’éléments dans la liste.

Ajouter d’abord dans l’adapter une variable liste de contact et un constructeur qui prend en paramètre un liste de contact :

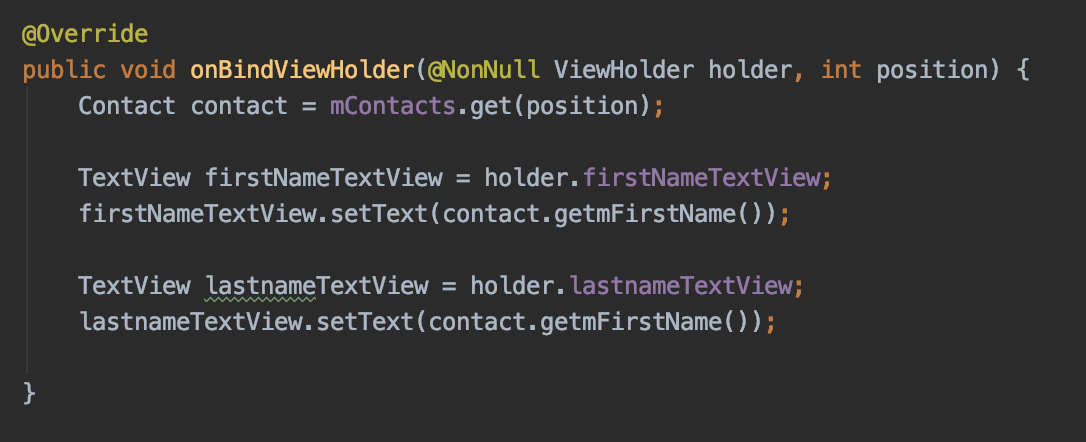


Ensuite implémentez les 3 méthodes ci dessus comme ceci :

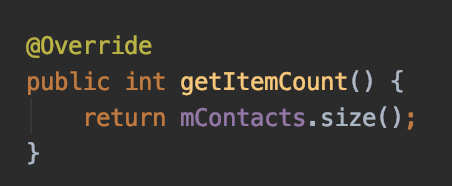
Dans cette méthode on utilise l’objet LayoutInflater pour déclarer quelle vue va décrire un item de la liste. La variable R.layout.item\_contact fait référence avec le fichier créé précédemment.



Dans la méthode onBindViewHolder, nous allons afficher et définir les différentes vues d’un item.



Ainsi que la méthode getItemCount() :

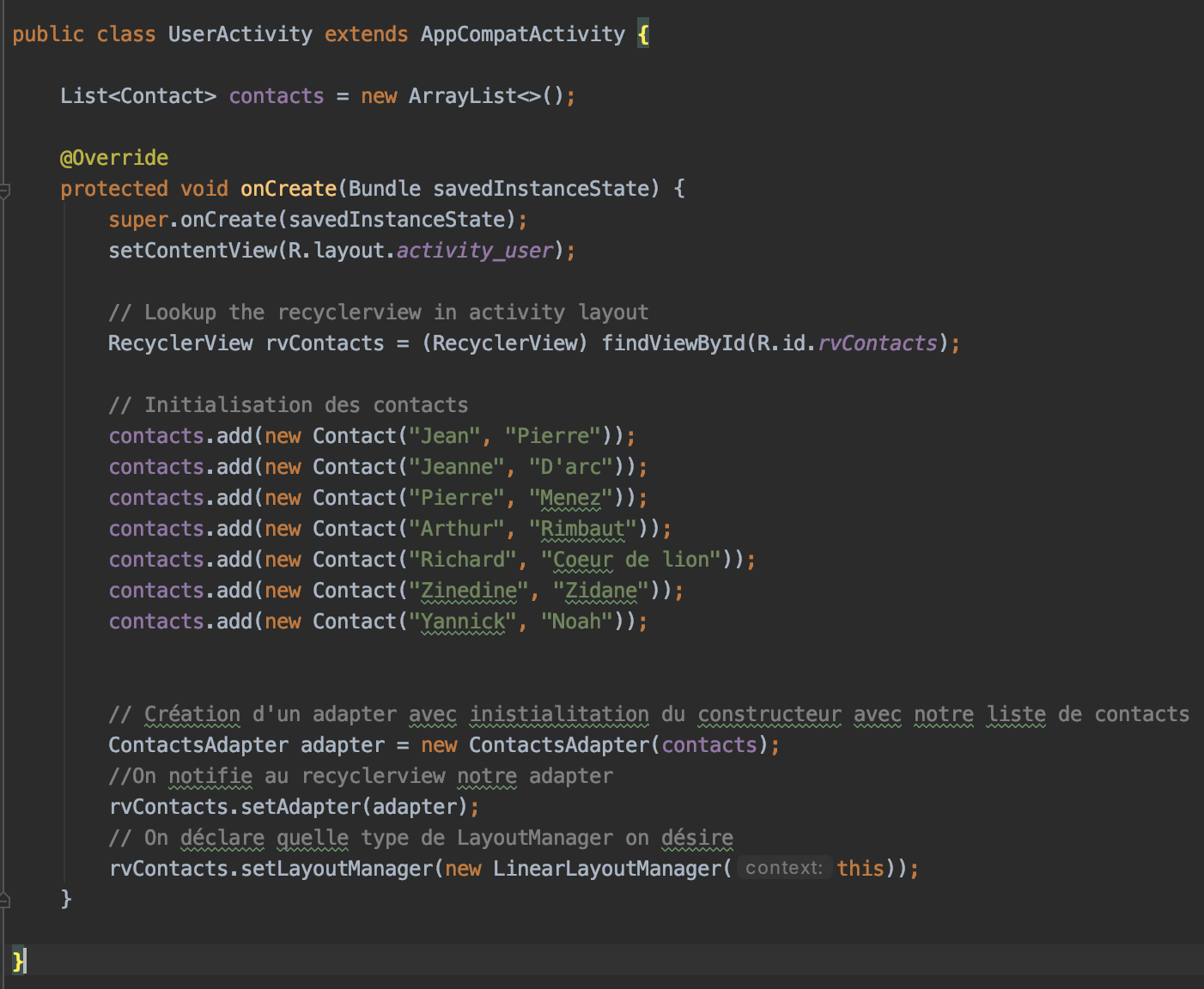


Vous devriez obtenir finalement :



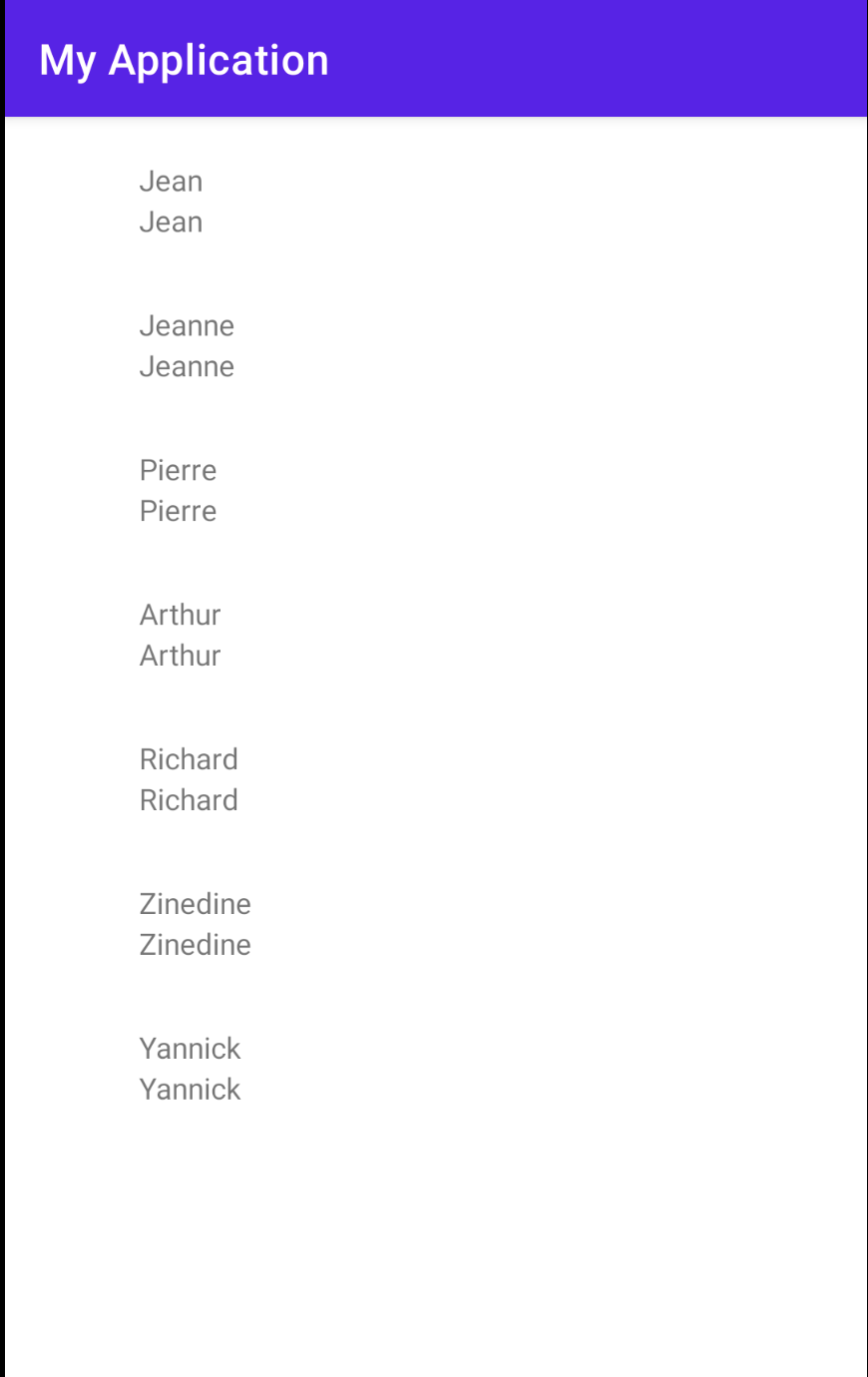
## Lier l'adaptateur au RecyclerView

Pour cela, on crée un nouvel adapter, on l’utilise pour initialiser notre RecyclerView, puis on déclare un type de layoutManager. Et c’est tout !



## Exercice

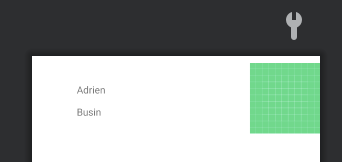
Si vous avez suivi à la lettre le code, vous devriez avoir ceci →



Exercice 1:

Vous remarquerez qu’une erreur simple a été glissée dans le code, le nom n’est pas affiché correctement, à vous de trouver le code et le corriger.

Exercice 2 : Ajouter une image pour chaque contact



Dans l’objet contact ajouter une variable de type String nommez la urlPhoto ajoutez la au constructeur.

Dans votre Activity, lorsque vous ajoutez des contacts, utilisez des url d’image google comme variable pour urlPhoto.

Exemple pour le contact Jeanne D’arc: <https://histoire-image.org/sites/default/jeanne-arc-sacre-charlesvii.jpg>

Vous utiliserez ensuite la librairie Glide → <https://github.com/bumptech/glide> pour afficher pour chaque contact son image.

A vous de lire la documentation Glide pour implémenter cette librairie Android.

Exemple de ce que vous devez obtenir

