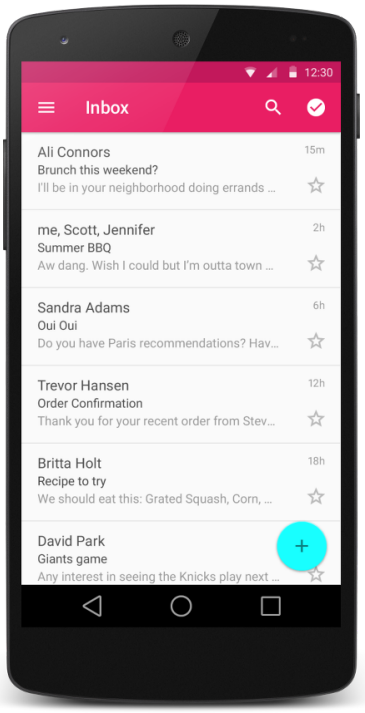
Tutorial. RecyclerView

Objectiu



El widget RecyclerView és un contenidor que ens permet mostrar un conjunt de dades.

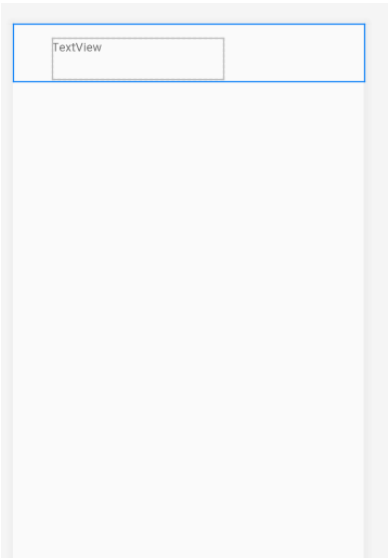
Nosaltres el farem servir per llistar els elements de la nostra app.

Demo

Descarrega la demo aquí

Pas a pas. Preparem les vistes.

- Creem una nova empty Activity que li direm *Llista*
- Instal·larem el RecyclerView al projecte. Al xml a Containers → RecyclerView →
- A l'xml afegim el RecyclerView en un Constraint Layout i li posarem l'id recyclerView
- Afegim un xml que es digui item_list.xml a la carpeta Res → Layout (Botó dret → New → Layout resource file)



A la vista item_list.xml el que farem serà detallar el que anirà a cada element de la llista del RecyclerView. De moment només hi afegim un camp de text (id: userName) en un Constraint Layout. Aquest Constraint Layout **NO** ha d'ocupar tota l'alçada de la pantalla ja que sinó només veurem un element. Per a fer-ho modifica el layout height per una mida fixa (ex: 80dp) o mitjançant el wrap_content (que s'adaptarà a la mida del que hi hagi dins)

Pas a pas. Preparem el codi.

- Una vegada tinguem la llista de dades creada el que farem serà cridar a la construcció del RecyclerView.

```
RecyclerView recyclerView = findViewById(R.id.recyclerView);
RecyclerViewAdapter adapter = new RecyclerViewAdapter(array_noms);
recyclerView.setAdapter(adapter);
recyclerView.setLayoutManager(new LinearLayoutManager((this)));
```

- El programa no coneixerà la classe `RecyclerViewAdapter` per tant el que haurem de fer és crear-la (botó dret → New → Java class) i posar-li el nom RecyclerViewAdapter

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.ViewHolder> {
    private ArrayList<String> array_noms;

    public RecyclerViewAdapter(ArrayList<String> arrN){
        array_noms = arrN;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_list, parent,
false);
        ViewHolder holder = new ViewHolder(view);
        return holder;
    }



    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        holder.etiquetaNom.setText(array_noms.get(position));
    }

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

    public class ViewHolder extends RecyclerView.ViewHolder{
        TextView etiquetaNom;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            etiquetaNom = itemView.findViewById(R.id.userName);
        }
    }
```

```
}
```

- onBindViewHolder. Afegeix el contingut als elements declarats al ViewHolder
- getItemCount. Ens diu quants elements haurà d'iterar de la llista
- ViewHolder. Enllaça els elements de l'item_list amb el RecyclerView

Si volem afegir una linia divisòria entre els elements haurem d'afegir aquesta línia de codi en el MainActivity.java

```
recyclerView.addItemDecoration(new DividerItemDecoration(this,
DividerItemDecoration.VERTICAL));
```

Pas a pas. Fem click a un element del recycler i obrim un nou fragment

Al mètode onBindViewHolder afegirem un setOnClickListener que en fer onClick ens obri un nou Fragment.

```
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AppCompatActivity app = (AppCompatActivity) v.getContext();
        BlankFragment blankFragment = new BlankFragment();

       app.getSupportFragmentManager().beginTransaction().replace(R.id.fr
       agment_container, blankFragment).commit();
    }
});
```