P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

# Programação Para Dispositivos Móveis I

## RecyclerView

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

NORTE 2030 Programa Regional do Norte

PORTUGAL 2030

Cofinanciado pela União Europeia

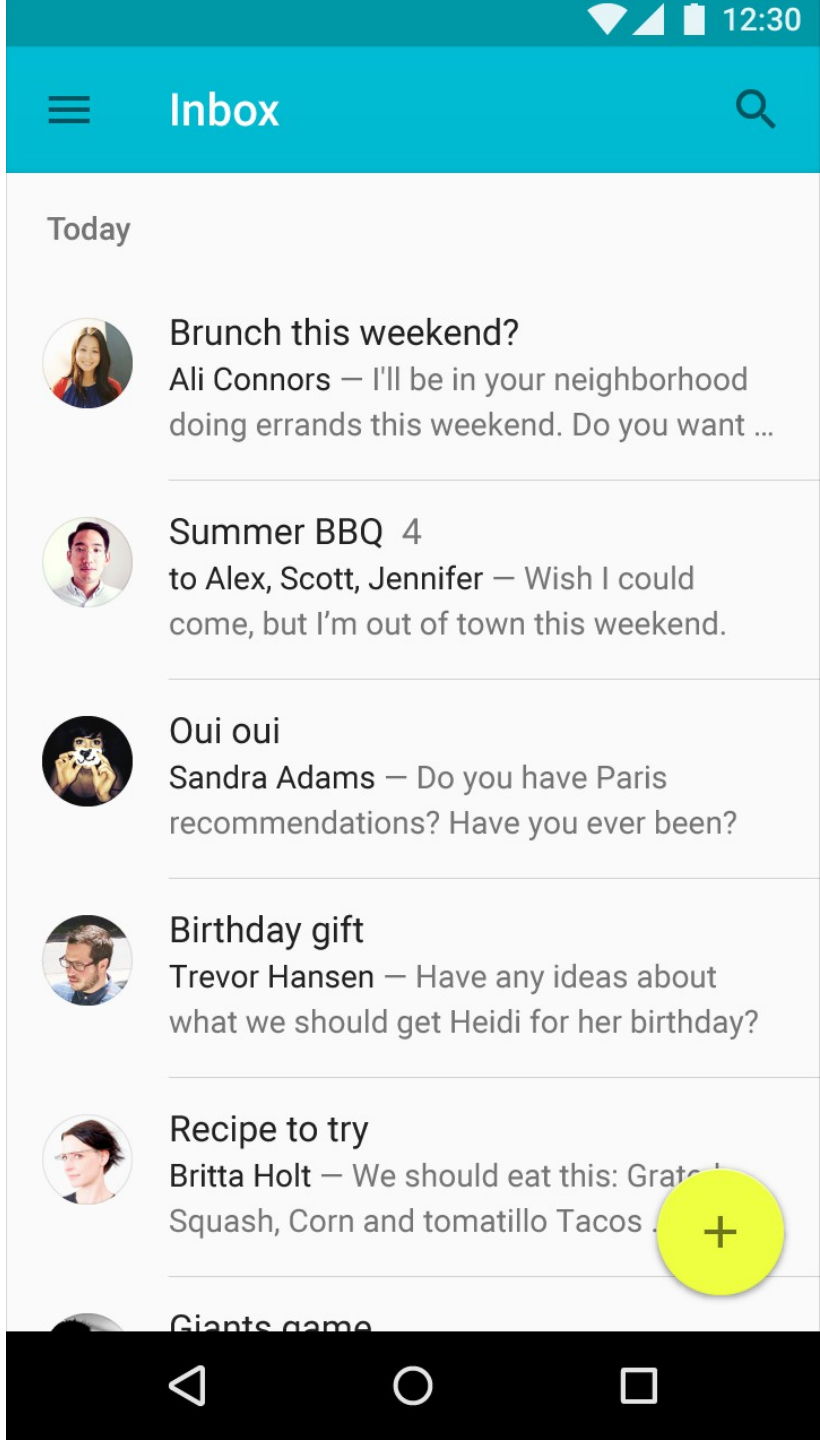
# Índice

- Introdução;
- RecyclerView;
- Arquitetura;
- Implementação;
- Atualizar dados na ReyclerView;
- Leitura Recomendada.

# Introdução

Em certas aplicações é necessário mostrar uma grande quantidade de dados ou apresentar dados que mudam regularmente;

Por exemplo, numa aplicação de SMS a quantidade de mensagens que um utilizador possui pode ascender a milhares. Ao mostrar esta informação ao utilizador **apenas um subset das mensagens serão visíveis no ecrã**.

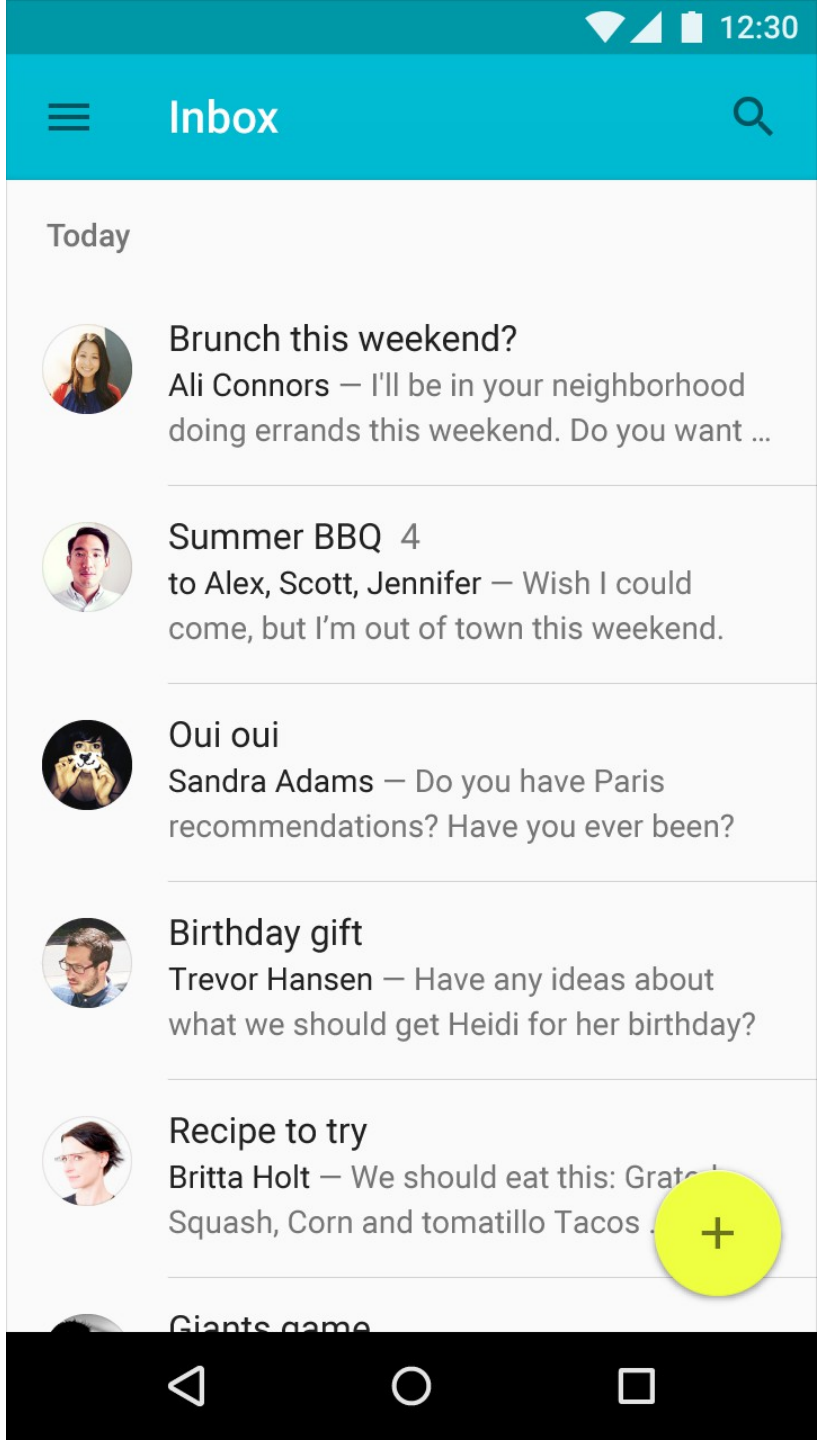



P.PORTO
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
NORTE 2020 Programa Regional do Norte
PORTUGAL 2030
Cofinanciado pela União Europeia

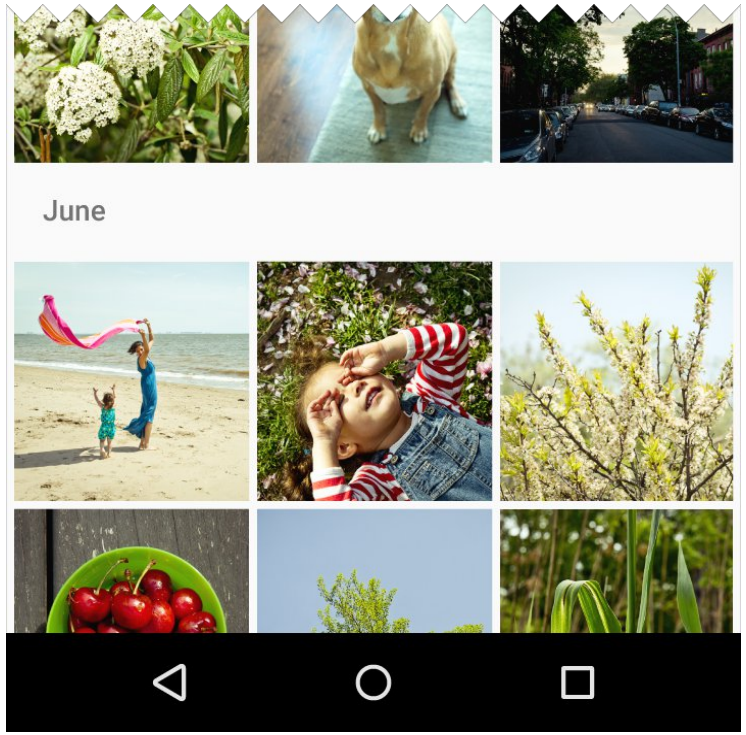
# Introdução
## O problema

Não faz sentido a nível de performance criar uma view  por cada mensagem, visto que podemos ficar sem memoria;

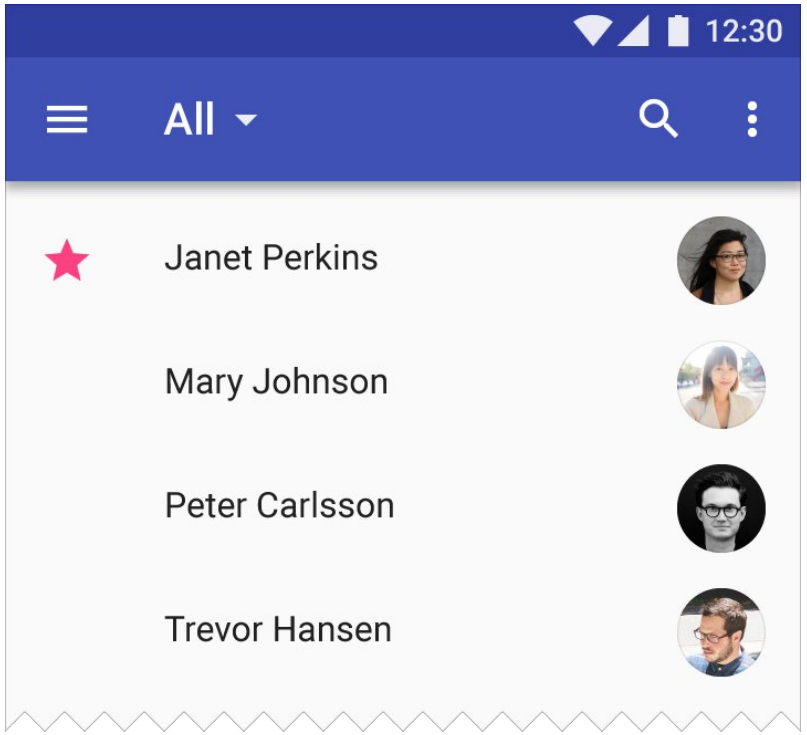
Também não é viável criar apenas as views quando o utilizador faz scroll na lista pois a instanciação de uma view é bastante dispendiosa e vai afetar a performance da aplicação.



# RecyclerView



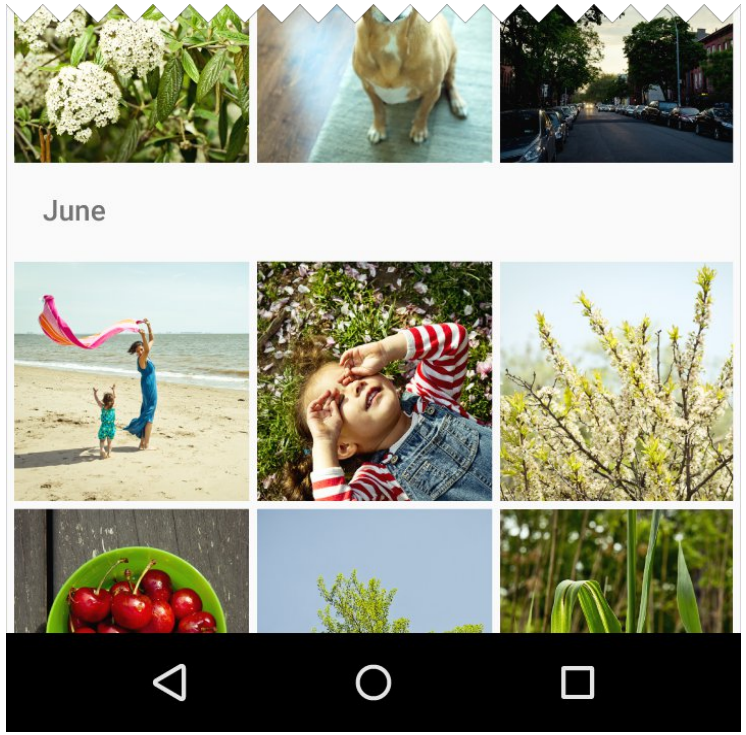

Grid



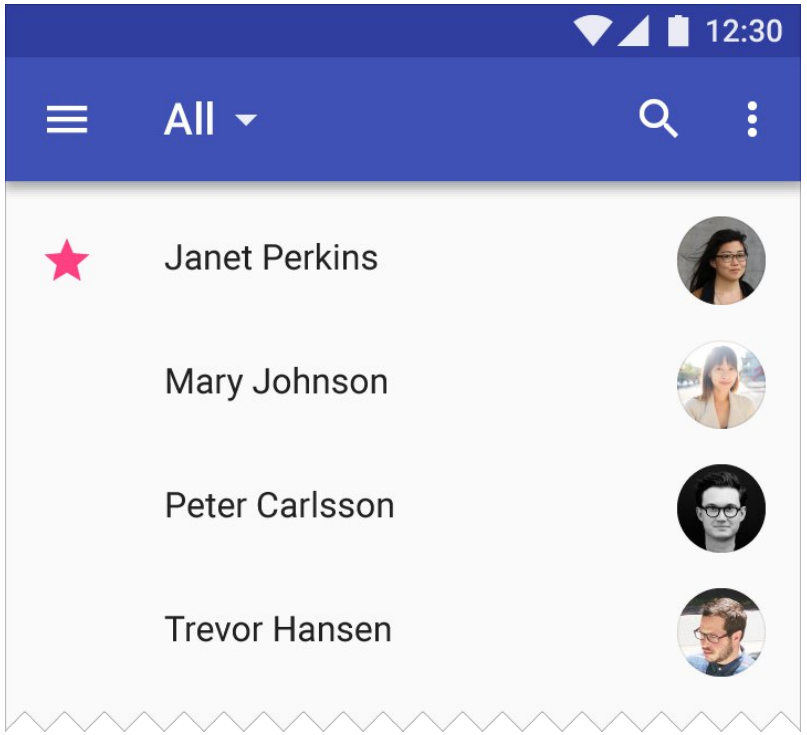

Lista

RecyclerView é um componente da biblioteca de suporte (v7) que facilita a apresentação de grandes quantidades de dados.

# RecyclerView




Grid

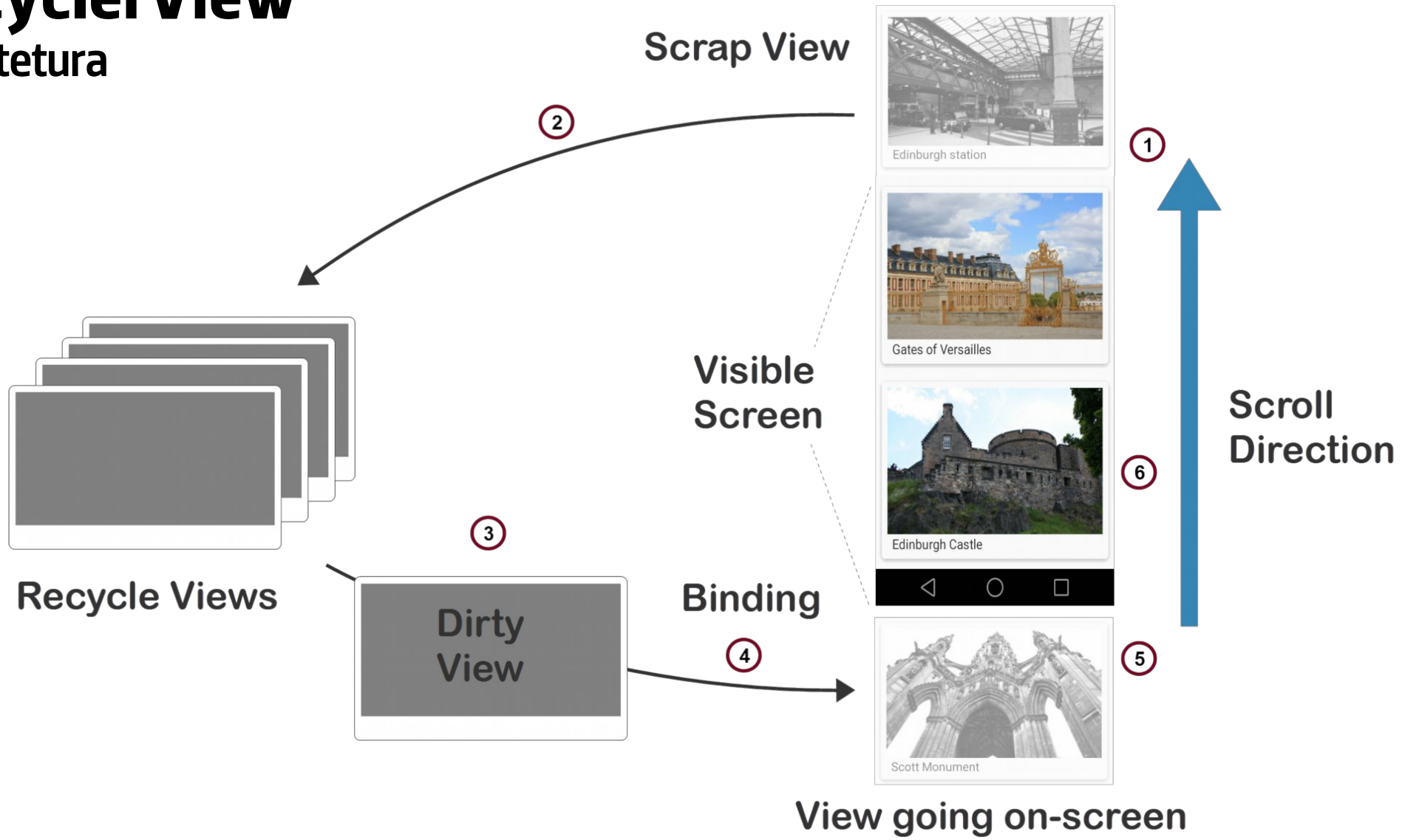



Lista

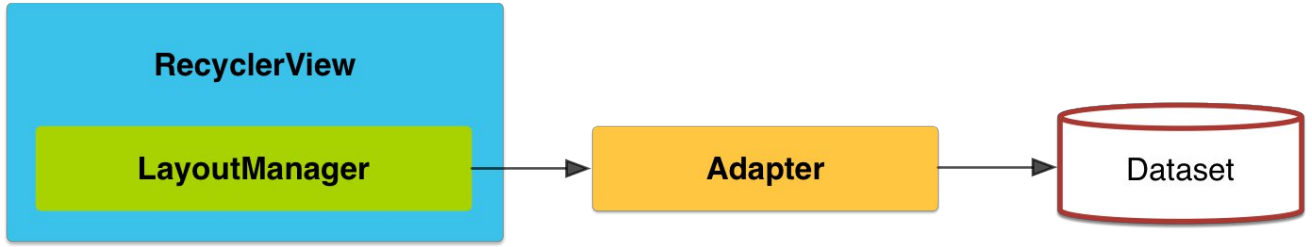
É uma view flexível que permite incluir uma fonte de dados extensa numa "janela" com espaço limitado.

Faz o scroll automático de dados se ficar sem "espaço" para serem mostrados no ecrã.

# RecyclerView
## Arquitetura



# RecyclerView
## Arquitetura




### LayoutManager

- Responsável pelo posicionamento das views, e pela gestão das mesmas;
- São utilizados diferentes LayoutManagers para obter diferentes tipos de layouts: (1) LinearLayoutManager, (2) GridLayoutManager, (3) StaggeredGridLayoutManager.

### Adapter

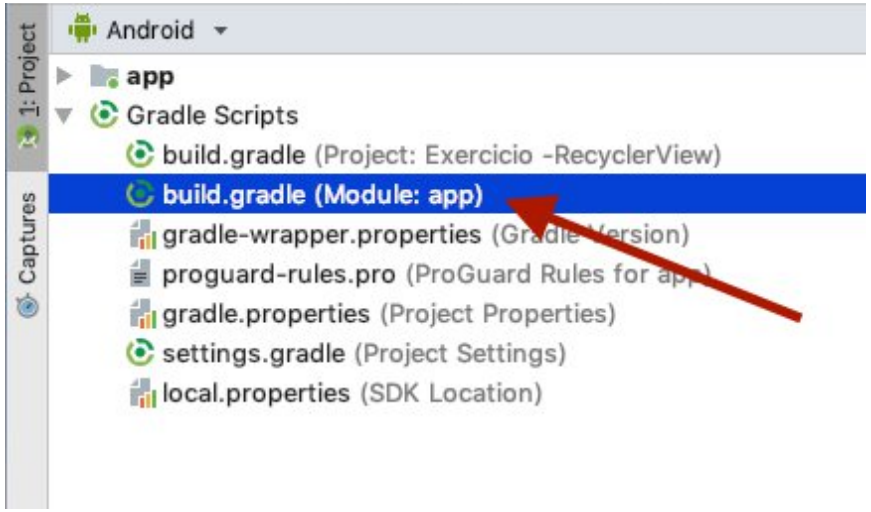
- Possui o dataset que vai ser mostrado;
- Serve de ponte entre o modelo de dados e a RecyclerView;

### ViewHolder

- Possui uma referencia para uma view a ser mostrada no ecrã. É neste componente que fazemos "binding" dos dados à view.

# RecyclerView
## Implementação

1. Adicionar a dependência da RecyclerView ao ficheiro gradle do **módulo**




```
dependencies {
    …
    implementation "androidx.recyclerview:recyclerview:1.4.0“

    // For control over item selection of both touch and mouse driven selection
    implementation "androidx.recyclerview:recyclerview-selection:1.1.0“
    …
}
```

Nota: devem usar a versão mais recente disponível!

https://developer.android.com/jetpack/androidx/releases/recyclerview

# RecyclerView
## Implementação (2)

```
public class Contact {

    private String name;
    private Boolean isOnline;

    public Contact(String name, Boolean isOnline){
        this.name = name;
        this.isOnline = isOnline;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Boolean isOnline() {
        return isOnline;
    }

    public void setOnline(Boolean onlineStatus) {
        isOnline = onlineStatus;
    }
}
```

2. Criar a class que define o modelo de dados (ex. **Contact.java**)

# RecyclerView
## Implementação (3)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical" />

</LinearLayout>
```

3. Adicionar o widget da RecyclerView ao layout da activity

   `activity_main.xml`

# RecyclerView

## Implementação (4)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal">


    <TextView
        android:id="@+id/txt_contact_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="10"
        android:padding="12dp" />

    <Button
        android:id="@+id/btn_contact_isOnline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1" />

</LinearLayout>
```

4. Criar o layout do item da RecyclerView. Este layout será replicado para todos os elementos a serem apresentados.

`item_contact.xml`

# RecyclerView
## Implementação (5)

```
public class ContactAdapter extends RecyclerView.Adapter<ContactAdapter.ContactViewHolder> {

    private ArrayList<Contact> contactList;

    public ContactAdapter(ArrayList<Contact> contactList) {
        this.contactList = contactList;
    }

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

5. Criar o adapter que vai conter os dados.(ex. `ContactAdapter.java`)

# RecyclerView

## Implementação (6)

```
// Provide a reference to the views for each data item
// Complex data items may need more than one view per item, and
// you provide access to all the views for a data item in a view holder
public static class ContactViewHolder extends RecyclerView.ViewHolder{

    public TextView txtView_contact_name;
    public Button btnMessage;

    public ContactViewHolder(@NonNull View itemView) {
        super(itemView);
        txtView_contact_name = itemView.findViewById(R.id.txt_contact_name);
        btnMessage = itemView.findViewById(R.id.btn_contact_isOnline);
    }
}
```

6. Ao extender `RecyclerView.Adapter<ContactAdapter.ContactViewHolder>` somos indicados para a implementação da respetiva class. Nesta classe realizamos a referencia às views presentes no layout para cada elemento (`item_contact.xml`).

# RecyclerView
## Implementação (7)

```
// Create new views (invoked by the layout manager)
@NonNull
@Override
public ContactViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    //Get Layout inflater from Context
    Context context = parent.getContext();
    LayoutInflater layoutInflater = LayoutInflater.from(context);

    //Inflate Layout
    View contactView = layoutInflater.inflate(R.layout.item_contact, parent, false);

    return new ContactViewHolder(contactView);
}
```

7. Implementar o método `onCreateViewHolder` que é invocado pelo Layout Manager escolhido e é responsável pela criação de novas vistas.

# RecyclerView

## Implementação (8)

```
// Replace the contents of a view (invoked by the layout manager)
@Override
public void onBindViewHolder(@NonNull ContactViewHolder viewHolder, int position) {
    //Get the data model based on position
    Contact contact = contactList.get(position);

    //Set name
    TextView textView = (TextView) viewHolder.txtView_contact_name;
    textView.setText(contact.getName());

    //Set button status
    Button button = (Button) viewHolder.btnMessage;
    button.setText(contact.isOnline() ? "Message" : "Offline");
    button.setEnabled(contact.isOnline());
}
```

8. Implementar o método `onBindViewHolder` que é invocado pelo Layout Manager escolhido e é responsável pela associação de dados às views presentes no layout.

# RecyclerView

## Implementação (9)

```
private RecyclerView recyclerView;
private ContactAdapter contactAdapter;
private ArrayList<Contact> contactList;

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

    //Define the LayoutManager
    recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    //Create contact List
    contactList = new ArrayList<>();
    addContacts(contactList);

    //Set the RecyclerView adapter
    contactAdapter = new ContactAdapter(contactList);
    recyclerView.setAdapter(contactAdapter);
}
```

9. Na `MainActivity.class` realizamos a instanciação do `adapter` e do respetivo `LayoutManager`.

Nota: se o conteúdo não alterar o tamanho do layout da RecyclerView devem utilizar a seguinte definição adicional (melhora a performance).

```
recyclerView.hasFixedSize(true);
```

# Atualizar dados na ReyclerView

Sempre que queremos adicionar, atualizar ou remover itens do nosso dataset é necessário notificar os componentes da RecyclerView das alterações.

- Desta forma o Adapter possui os seguintes métodos:

```
notifyItemChanged(int position)     // Item na posiçao foi atualizado
notifyItemInserted(int position)    // Adicionado item na posição
notifyItemRemoved(int position)     // Removido item na posição
notifyDataSetChanged()              // Toda a lista foi atualizada
```

**Exemplo:**

```
// Adiciona um novo contacto na posição 0
contacts.add(0, new Contact("Barney", true));

// Notificamos o adapter que foi adicionado um elemento
// novo na posição 0
adapter.notifyItemInserted(0);
```

# Decorations

- As decorations permitem adicionar certos elementos visuais a cada view de uma RecyclerView.
  - O uso mais comum é o de adicionar os dividers (linhas divisórias) entre as views.
  - Também se usam para modificar o espaçamento entre os elementos da lista.


**Exemplo:**

```
RecyclerView.ItemDecoration itemDecoration = new DividerItemDecoration(this, DividerItemDecoration.VERTICAL);
recyclerView.addItemDecoration(itemDecoration);
```

# Leitura Recomendada

## Recyclerview

https://developer.android.com/jetpack/androidx/releases/recyclerview

## Guia recyclerview

https://developer.android.com/guide/topics/ui/layout/recyclerview

P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

# Programação Para Dispositivos Móveis I

## RecyclerView

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

NORTE 2030 Programa Regional do Norte

PORTUGAL 2030

Cofinanciado pela União Europeia