Arquitetura de Software MVC

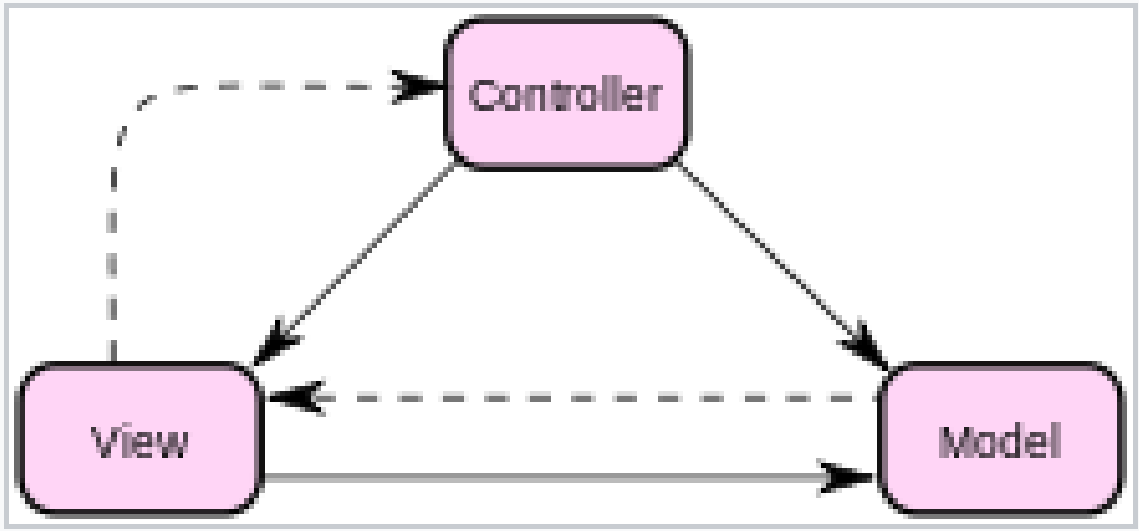
Desenho de Interfaces Gráficas

#ListView #GridView #RecyclerView

Sónia Luz, sonia.luz@ipleiria.pt
David Safadinho, david.safadinho@ipleiria.pt
Departamento de Engenharia Informática
Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Leiria
1º Semestre - 2021/2022

Arquitetura de Software MVC

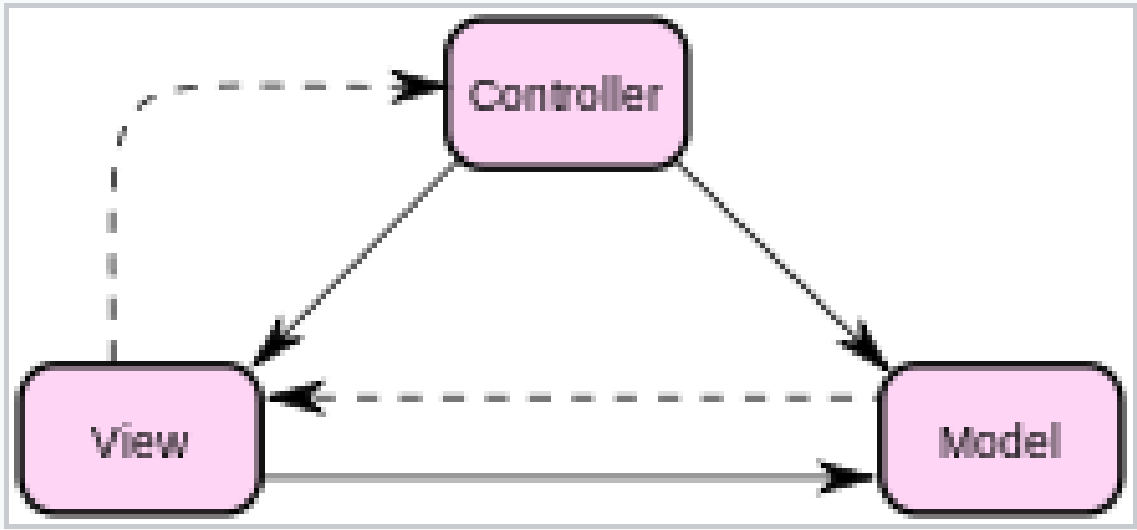
- Model-View-Controller
  - É um padrão de arquitetura de software
    - separa a representação da informação:
    - da interação do utilizador com o software;
  - Ideias centrais por detrás do MVC são:
    - reutilização de código
    - separação de conceitos.



Um diagrama simples exemplificando a relação entre *Model*, *View* e *Controller*. As linhas sólidas indicam associação direta e as tracejadas indicam associação indireta.

Arquitetura de Software MVC

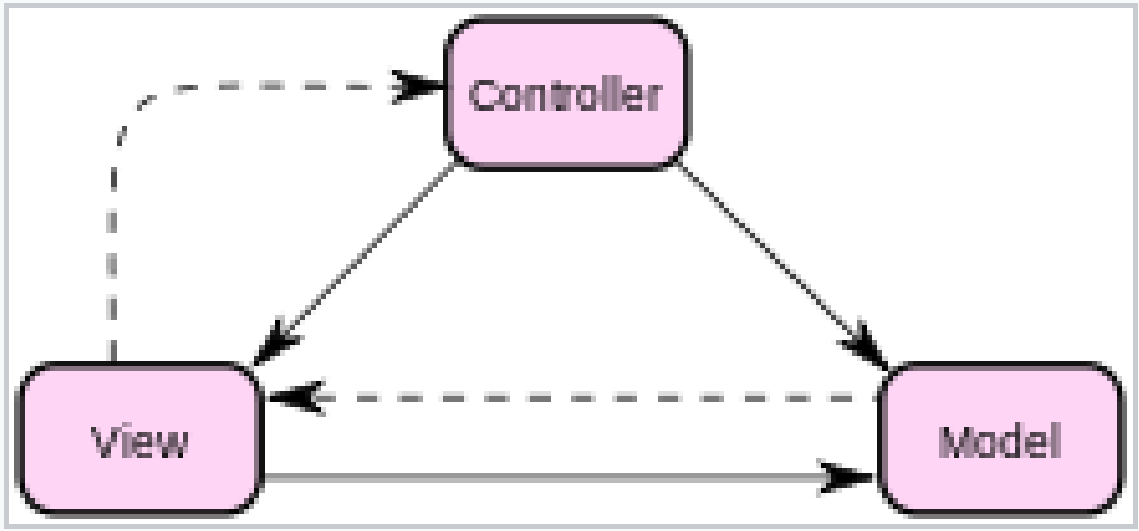
- Modelo (*model*)
  - Consiste nos:
    - dados da aplicação;
    - regras de negócios;
    - lógica e funções.



Um diagrama simples exemplificando a relação entre *Model*, *View* e *Controller*. As linhas sólidas indicam associação direta e as tracejadas indicam associação indireta.

Arquitetura de Software MVC

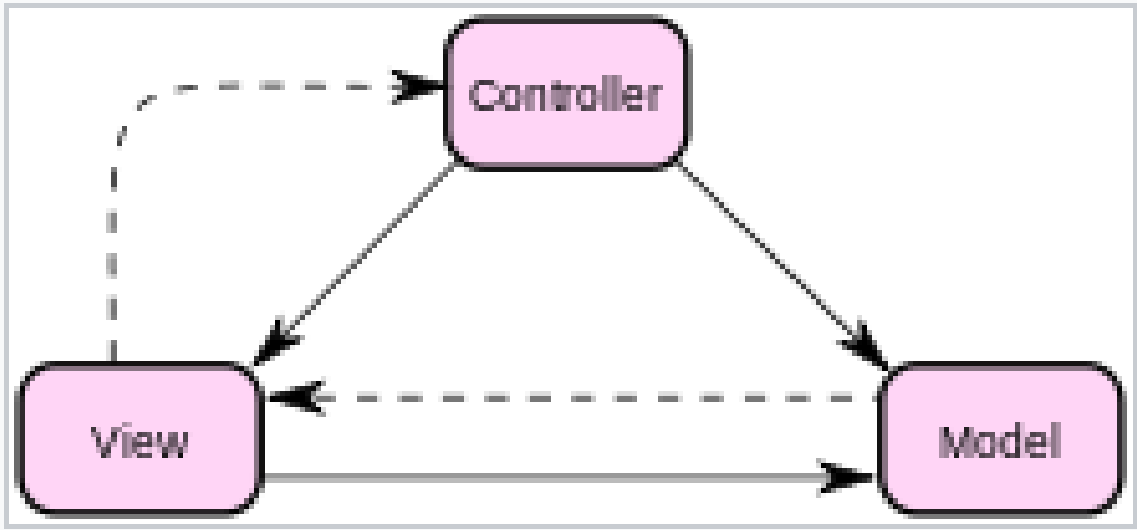
- Vista (*view*)
  - Pode ser qualquer saída de representação dos dados:
    - ex: lista, tabela, diagrama;
  - É possível ter várias visões do mesmo dado:
    - um gráfico de barras;
    - uma visão tabular;



Um diagrama simples exemplificando a relação entre *Model*, *View* e *Controller*. As linhas sólidas indicam associação direta e as tracejadas indicam associação indireta.

Arquitetura de Software MVC

- Controlador (controller)
  - Executa a mediação da entrada;
  - Convertendo-a em comandos
    - para o modelo ou a vista.



Um diagrama simples exemplificando a relação entre *Model*, *View* e *Controller*. As linhas sólidas indicam associação direta e as tracejadas indicam associação indireta.

Layouts: ListView e GridView

- Listas e grelhas são dois elementos de interação com o utilizador bastante usados em dispositivos móveis;

- Por isso, a plataforma Android possui dois layouts/*ViewGoups* dedicados a este tipo de interação:
  - ***ListView***
  - ***GridView***

ListView

- É um *ViewGroup* que permite a disposição de elementos numa lista com deslocamento vertical
  - É responsável pela gestão e controlo:
    - dos elementos visíveis;
    - dos elementos que ficam fora da área visível;
    - da reutilização dos objetos criados;

GridView

- É um *ViewGroup* que permite a apresentação de elementos
  - Com toda a gestão a cargo do componente;
  - Os elementos são dispostos numa grelha bidimensional (linhas e colunas).

ListView e GridView

- Se olharmos para a API de uma ***ListView*** ou uma ***GridView***
  - Não há forma de configurarmos os componentes visuais;
  - Por isso temos de recorrer a um **adaptador** (***Adapter)***
    - Que colocará os itens da lista/matriz automaticamente na lista, ou na grelha;
    - Podemos recorrer a um adaptador já existente
      - ex: *ArrayAdapter*, *CursorAdapter*, *SimpleAdapter* etc.;
    - Ou criar um adaptador personalizado
      - com base na classe *BaseAdapter*

Adapter (Adaptador)

- Adaptadores - O que são?
  - Intermediários entre o modelo e as vistas
    - Representam a ligação entre a *View* e alguma fonte de dados;
    - Irão obter a informação do modelo e fornecer essa informação às vistas.
  - Uma aplicação pode ter vários adaptadores, adequados a cada situação;
  - Permite o acesso aos itens de dados;
    - É responsável por fazer uma exibição para cada item no conjunto de dados.

ListViews e Adapters

- Vamos considerar um projeto **Gestor de Contactos**
  - Que irá conter contactos;
  - E apresentar a informação dos contactos existentes;
- Neste projeto, vamos seguir a arquitetura MVC
  - Assim além da **MainActivity**;
  - Vamos ter um package **modelo** com as classes **Contacto** e **GestorContactos**;
- Para apresentar os dados dos contactos
  - Vamos criar um atividade **ListaContactosActivity**;
  - Com uma *TextView* para apresentação da informação existente;

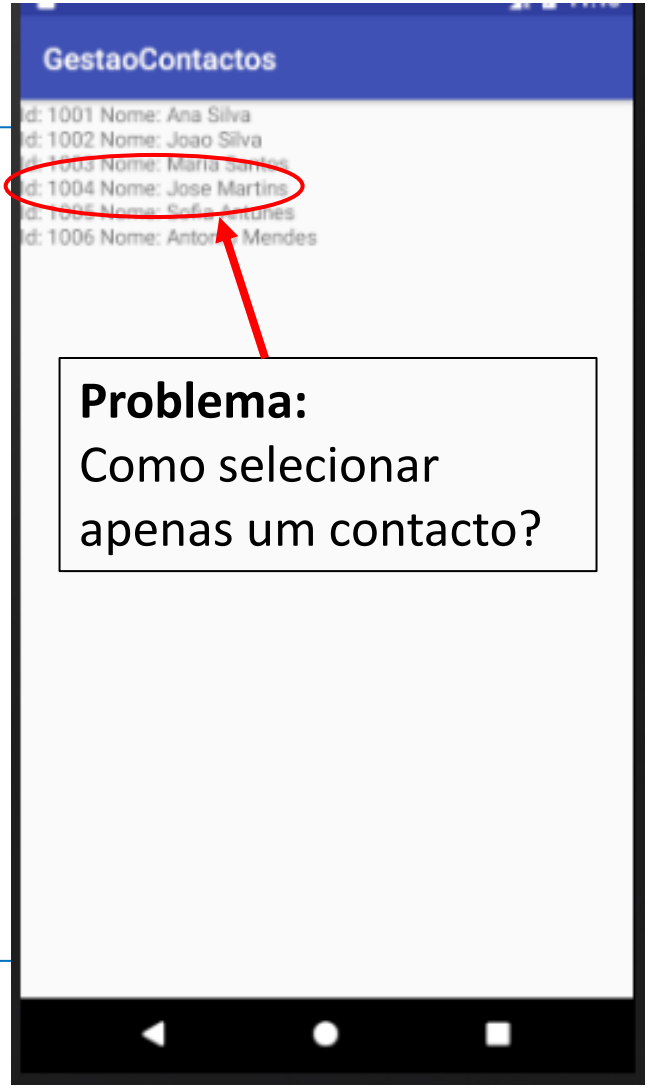
ListViews e Adapte

```java
public class Contacto{
    private long id;
    private String nome;

    public Contacto(long id, String nome) {
        this.id = id;
        this.nome = nome;
    }

    public long getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }

    @Override
    public String toString() {
        return "Id: " + id + " Nome: " + nome;
    }
}
```

```java
public class GestorContactos {

    private LinkedList<Contacto> contactos;

    public GestorContactos(){
        contactos = new LinkedList<>();
        adicionarDadosIniciais();
    }

    private void adicionarDadosIniciais() {
        contactos.add(new Contacto(1001, "Ana Silva"));
        contactos.add(new Contacto(1002, "Joao Silva"));
        contactos.add(new Contacto(1003, "Maria Santos"));
        contactos.add(new Contacto(1004, "Jose Martins"));
        contactos.add(new Contacto(1005, "Sofia Antunes"));
        contactos.add(new Contacto(1006, "Antonio Mendes"));
    }

    public LinkedList<Contacto> getContactos() {
        return new LinkedList<>(contactos);
    }

    public Contacto getContacto(int index){
        return contactos.get(index);
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (Contacto contacto: contactos) {
            sb.append(contacto).append("\n");
        }
        return sb.toString();
    }
}
```

ListViews e Adapters


```
public class ListaContactosActivity extends AppCompatActivity{

    private GestorContactos gestorContactos;

    private TextView textViewContactos;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lista_contactos);

        gestorContactos = new GestorContactos();
        textViewContactos =
          findViewById(R.id.textViewListaContactos);

        apresentarContactos();
    }

    private void apresentarContactos() {
        textViewContactos.setText(gestorContactos.toString());
    }
}
```



Problema:
Como selecionar apenas um contacto?

ListViews e Adapters

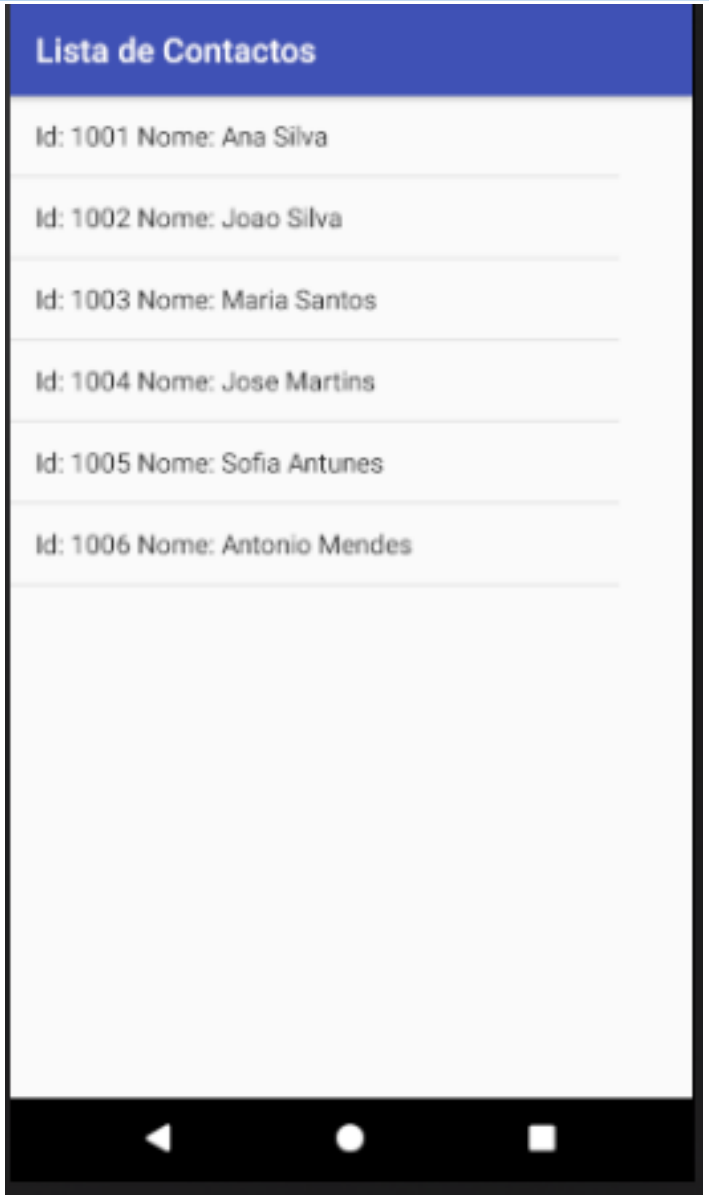
- Problema:
  - Não conseguimos selecionar apenas 1 contacto
- Solução:
  - Utilizar uma *ListView* e respetivo *Adapter;*
    - Assim, vamos alterar o componente *TextView* da **ListaContactosActivity** para uma *ListView;*
    - Criar um adaptador do tipo ***ArrayAdapter*** para que este passe os diversos itens de contactos para a *ListView;*
    - Atribuir o adaptador à *ListView* através do método ***setAdapter***()

ListViews e Adapters

- ArrayAdapter – uma definição possível:
  - `public ArrayAdapter (Context context, int resource, T[] objects);`
  - Significado dos argumentos:
    - *context* → O contexto atual;
    - *resource* → O id para o tipo de layout que contém o layout a ser usado quando instanciarmos uma *View*;
    - *objects* → Os objetos a serem representados na *ListView.*

ListViews e Adapters

```
public class ListViewContactosActivity extends AppCompatActivity{
    private GestorContactos gestorContactos;
    private ListView listViewContactos;
     private ArrayAdapter<Contacto> adaptador;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_view_contactos);
        setTitle("Lista de Contactos");

        gestorContactos = new GestorContactos();
        listViewContactos = findViewById(R.id.listViewContactos);
        atualizarContactos();
    }

    private void atualizarContactos() {
        //criar o arrayAdapter, recebe o contexto onde é usado,
        // o tipo de layout a usar para representar e a lista de dados a apresentar
        adaptador = new ArrayAdapter<Contacto>(this,
                    android.R.layout.simple_list_item_1,
                    gestorContactos.getContactos());
         //atribuir o adapter à listView
        listViewContactos.setAdapter(adaptador);
    }
}
```



ListViews e Adapters

- Neste caso cada item da lista
  - É representado através de uma *TextView;*
  - Porque o *ArrayAdapter* não nos permite alterar a *view* de apresentação de dados;
  - Assim, para ter um layout específico para cada item
    - devemos criar um *Adapter* personalizado, que deve herdar de *BaseAdapter*

ListViews e Adapters

- Como fazer para selecionar um contacto da lista?
  - É necessário criar um *listener* que fique à escuta do *click* sobre o item da lista através do *AdapterView*

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        //Codigo a ser executado aquando do click no Item
    }
});
```

  - E uma atividade **DetalhesContacto** que apresente os detalhes de um contacto

ListViews e Adapters

- Atividade **DetalhesContacto** que apresente os detalhes de um contacto


```
public class DetalhesContacto extends AppCompatActivity {
    public static final String POSICAO = "amsi.dei.estg.ipleiria.pt.POSICAO";
    private GestorContactos gestorContactos;
    private Contacto contacto;
    private TextView textViewId;
    private TextView textViewNome;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detalhes_contacto);
        setTitle("Detalhes de contacto");
        textViewId = findViewById(R.id.textViewId);
        textViewNome = findViewById(R.id.textViewNome);
        gestorContactos = new GestorContactos();
        int posicao = getIntent().getIntExtra(POSICAO,0);
        contacto = gestorContactos.getContacto(posicao);
        if(contacto != null)
            preencherContacto();
    }

    private void preencherContacto() {
        textViewId.setText("" + contacto.getId());
        textViewNome.setText(contacto.getNome());
    }
}
```

ListViews e Adapters

- Criação de um *listener* que fique à escuta do *click* sobre o item da lista através do *AdapterView*

```
public class ListViewContactosActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //...
        atualizarContactos();

        listViewContactos.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Intent intent = new Intent(getApplicationContext(), DetalhesContacto.class);
                intent.putExtra(DetalhesContacto.POSICAO, position);
                startActivity(intent);
            }
        });
    }
}
```

GridViews e Adapters

- Para representar a informação dos contactos através de uma *GridView*
  - Devemos criar uma atividade **GrelhaContactosActivity**
    - Com um componente *GridView*
      - para apresentar a informação em forma de grelha;
    - No entanto, por omissão, a informação aparece como lista;
    - É necessário configurar o atributo **numColumns** para "autofit"
      - no xml: `android:numColumns="auto_fit"`
      - ou diretamente nas propriedades do componente
    - Podemos criar um *ArrayAdapter*, tal como fizemos para a atividade Lista de Contactos;
    - Ou definir um novo package **adapters** e criar aí o *Adapter* personalizado;

GridViews e Adapters

- Como se define um *Adapter* personalizado?
  - Criamos uma classe que herde de ***BaseAdapter***
    - Automaticamente terá de implementar um conjunto de 4 métodos;
    - Definir os atributos que representam o contexto e a lista de dados a apresentar;
    - Definir o construtor de acordo com os atributos definidos;
    - Implementar os métodos;

```
public class ContactoAdapter extends BaseAdapter{
    @Override
    public int getCount() {
        //devolve o total de elementos da lista
        return 0;
    }

    @Override
    public Object getItem(int position) {
        //devolve o item da lista nessa posição
        return null;
    }

    @Override
    public long getItemId(int position) {
        //devolve o Id do item da lista
        return 0;
    }

    @Override
    public View getView(int position, View convertView,
          ViewGroup parent) {
        //criar uma view para cada item
        //através do método inflate() criar uma view
        // a partir do layout especifico do item
        return null;
    }
}
```

Layouts: RecyclerView

- *ListView* e *GridView* são considerados os *ViewGroup* base
  - Mas com a evolução dos componentes para Android
    - estão considerados *Deprecated*

- Por isso, a plataforma Android possui um novo layout/*ViewGoup*:
  - ***RecyclerView***

RecyclerView

- Uma versão mais avançada e mais flexível do que a *ListView*
  - Introduzido com Android Lollipop (versão 5.0 - API 21)
- Diversos componentes trabalham em conjunto

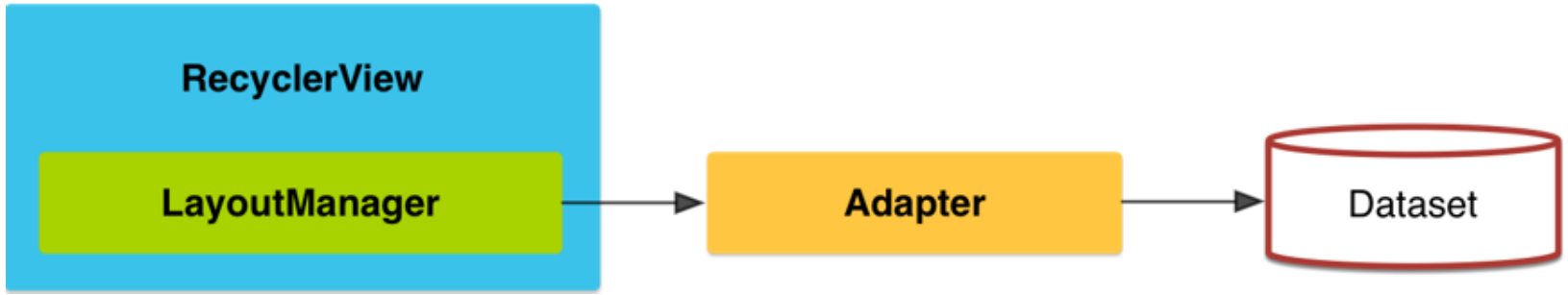


Diagrama de funcionamento do *RecyclerView*

  - É um *container* que encapsula um *LayoutManager* e um *ItemAnimator* e comunica com *RecyclerView.Adapter*

RecyclerView

- É necessário adicionar a dependência
  - Nas versões anteriores (até API 28) :

```
dependencies {
    ...
    implementation 'com.android.support:recyclerview-v7:28.0.0'
}
```

  - A partir do SDK de compilação Android 9.0 (API nível 28 ou superior) – passa a estar incluído na biblioteca Android Jetpack

```
dependencies {
    ...
    implementation "androidx.recyclerview:recyclerview:1.2.1"
    // For control over item selection of both touch and mouse driven selection
    implementation "implementation "androidx.recyclerview:recyclerview-selection:1.1.0"
}
```

RecyclerView

- Não possui responsabilidade de posicionar elementos na lista (permitindo utilizar o RecyclerView em vários *layouts*)
  - Delega nos ***LayoutManager***
    - ***LinearLayoutManager*** – para construção de listas horizontais ou verticais
    - ***GridLayoutManager*** – para construção de *Grids*
    - ***StaggeredGridLayoutManager*** – para construção de *Grids*, com itens de tamanhos variados (efeito mosaico)

RecyclerView

- É necessário criar um adaptador personalizado
  - Deve herdar de ***RecyclerView.Adapter***
    - Implica o uso de um *ViewHolder*
      - Melhora a performance das listas porque cria uma "cache" das *views*
  - E implementar 3 métodos:
    - ***getItemCount*** – devolve o número de elementos na lista de dados
    - ***onCreateViewHolder*** – devolve uma nova instância do *viewHolder*
    - ***onBindViewHolder*** – inclui os dados em cada item da lista
- Atribuir o *adapter* à *RecyclerView através do* ***setAdapter***

RecyclerView

- É possível animar as notificações enviadas ao *adapter*
  - Ex: eventos de remoção e adição de itens à lista
  - Através do ***ItemAnimator***

- *RecyclerView* já prevê algumas animações padrão
  - Sendo necessário substituir o método ***notifyDataSetChanged*** do *adapter*
  - ***notifyItemInserted(int position)*** – para indicar a inserção de um novo item na posição indicada;
  - ***notifyItemRemoved(int position)*** – para indicar a remoção de um item existente na posição indicada.

Desafio:

- Criar uma aplicação de gestão de contactos:
  1. Adicionar a cada contacto uma imagem;
     - **Nota:** O tipo de dados a guardar irá corresponder ao Id de cada imagem *drawable* armazenada;
  2. Criar uma atividade que represente os contactos através de uma *GridView*;
  3. Criar um *layout* específico para um item de contacto;
  4. Implementar um *Adapter* personalizado que utilize o *layout* por item para representar cada *view/item* da *gridview.*

Fontes e Mais Informação

- Arquitetura de Software MVC
  - https://pt.wikipedia.org/wiki/MVC
- *ViewGoups - ListView*
  - https://developer.android.com/reference/android/widget/ListView
- *ViewGoups - GridView*
  - https://developer.android.com/reference/android/widget/GridView
- Adaptadores em Android
  - https://developer.android.com/reference/android/widget/Adapter.html
- Criação de Layouts com Adaptadores
  - https://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews
- *ViewGoups - RecyclerView*
  - https://developer.android.com/jetpack/androidx/releases/recyclerview
  - https://developer.android.com/guide/topics/ui/layout/recyclerview

Próximo Tema:

Desenvolvimento Móvel em Android: Menus e Settings

- Menus
  - https://developer.android.com/guide/topics/ui/menus.html
- *Action Bar*
  - https://developer.android.com/training/appbar/index.html
- *Adicionar Acção Up (Voltar Atrás)*
  - https://developer.android.com/training/appbar/up-action.html#declare-parent
- *SharedPreferences*
  - https://developer.android.com/training/basics/data-storage/shared-preferences.html