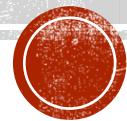


ANDROID

ViewGroups



Layouts

- Interfaces das aplicações
- Armazenadas dentro de **res/layout**
 - Ex.: **principal.xml**
- Podem ser acessados no código como:
 - Ex.: **R.layout.principal**
- Dentro do XML definem-se as interfaces com elementos do tipo View e ViewGroup

ViewGroup

- São elementos agrupadores, que dispõe seu filhos em uma determinada forma
- Conteúdo Estático
 - **LinearLayout**: componentes dispostos linearmente
 - Vertical - um abaixo do outro
 - Horizontal - um ao lado do outro
 - **RelativeLayout**: coloca um componente em uma posição relativa a outro
 - **TableLayout**: organiza os elementos como uma tabela
- Não permitem rolagem, para isso deve-se usar um **ScrollView** e colocar o layout como seu filho
- Conteúdo Dinâmico: usa-se adaptadores
 - **ListView**: mostra elementos verticalmente em uma lista rolável
 - **GridView**: mostra elementos em uma grade rolável

Prof. Razer A N R Montaño

2017

3

4

ScrollView

Prof. Razer A N R Montaño

2017

ScrollView

- Elemento usado para que a tela seja rolável, visto que os layouts não suportam rolagem nativamente
- Link: <https://developer.android.com/reference/android/widget/ScrollView.html>
- Rolagem Vertical (para horizontal, use `HorizontalScrollView`)
- Deve ter somente um filho
- Atributos
 - `android:fillViewport`: (true/false) define se o ScrollView deve esticar seu componente filho para preencher toda a tela, caso ele seja menor. Se o filho for maior, nada é alterado e o ScrollView se comporta como esperado

Prof. Razer A N R Montaño

2017

5

ScrollView

- Exemplo no XML

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ScrollView01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <!-- SEU LAYOUT AQUI -->

</ScrollView>
```

Prof. Razer A N R Montaño

2017

6



LinearLayout

Prof. Razer A N R Montaño

2017

LinearLayout

- Agrupa elementos em uma única coluna (um abaixo do outro) ou em uma única linha (um ao lado do outro)
- Link: <https://developer.android.com/reference/android/widget/LinearLayout.html>
- Atributos
 - **android:orientation:** (horizontal/vertical) se os elementos serão organizados em uma coluna ou em uma linha
 - **android:gravity:** ajusta os elementos filhos:
 - **center**: centralizados na horizontal e vertical
 - **center_horizontal**: centralizados somente na horizontal
 - **center_vertical**: centralizados somente na vertical
 - **left**: alinhados à esquerda
 - **right**: alinhados à direita
 - **top**: alinhados acima
 - **bottom**: alinhados abaixo

Prof. Razer A N R Montaño

2017

8

LinearLayout

- Exemplo no XML

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
  
    ...  
</LinearLayout>
```

10

RelativeLayout

RelativeLayout

- Os elementos filhos podem ser posicionados um relativo ao outro, ou relativos ao próprio layout
- Link: <http://developer.android.com/reference/android/widget/RelativeLayout.html>
- Por default, posiciona todos os elementos filhos à esquerda e no topo
- Nos **elementos filhos**, usam-se atributos para acertar o posicionamento
 - **android:layout_below**: abaixo de qual widget deve ser posicionado
 - **android:layout_above**: acima de qual widget deve ser posicionado
 - **android:layout_alignLeft**: alinha o widget à esquerda com o elemento indicado
 - **android:layout_alignRight**: alinha o widget à direita com o elemento indicado
 - **android:layout_toLeftOf**: posiciona o widget à esquerda do elemento indicado
 - **android:layout_toRightOf**: posiciona o widget à direita do elemento indicado
 - etc

Prof. Razer A N R Montaño

2017

11

RelativeLayout

- Exemplo no XML

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/txt_escolha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/escolha" />
    <SeekBar
        android:id="@+id/seek"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="45"
        android:layout_below="@+id/txt_escolha"/>

</RelativeLayout>
```

Prof. Razer A N R Montaño

2017

12

13

TableLayout

Prof. Razer A N R Montaño

2017

TableLayout

- Organiza os elementos em um esquema de tabela, com linhas e colunas
- Link: <https://developer.android.com/reference/android/widget/TableLayout.html>
- É formado por vários TableRows
 - Link: <https://developer.android.com/reference/android/widget/TableRow.html>
 - Cada linha contém 0 ou mais células, cada célula um elemento
- O número de colunas da tabela é o número de elementos da maior linha
- A largura de uma coluna é definida pela linha que tiver a maior largura
- Atributos
 - **android:shrinkColumns**: índices (início em 0) das colunas que devem ser comprimidas, caso sejam muito grandes, fazendo com que sejam apresentadas integralmente, ex, se um elemento de texto dentro for muito grande, seu conteúdo é quebrado
 - **android:stretchColumns**: índices (início em 0) das colunas que devem ser esticadas, fazendo com que ocupem todo o espaço disponível

Prof. Razer A N R Montaño

2017

14

TableLayout

- Exemplo no XML

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/table1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="0,1">
    <TableRow>
        <android:id="@+id/tr1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >
            <TextView
                android:id="@+id/txt1"
                android:text="Linha 1 Coluna 1" />
            <Button
                android:id="@+id/btn1"
                android:text="Linha 1 Coluna 2" />
        </TableRow>
        <TableRow>
            <android:id="@+id/tr2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" >
                <EditText
                    android:id="@+id/edit1"
                    android:text="Linha 2 Coluna 1" />
            </TableRow>
    </TableLayout>
```

Prof. Razer A N R Montaño

2017

15

16

ConstraintLayout

Prof. Razer A N R Montaño

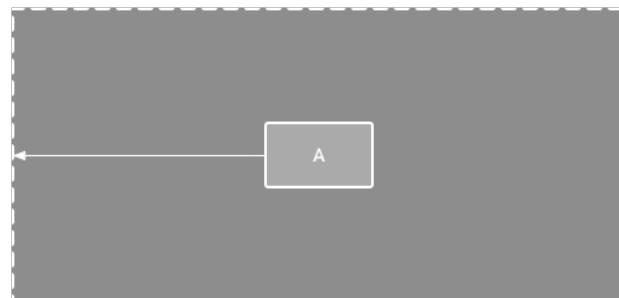
2017

ConstraintLayout

- Os elementos filhos podem ser posicionados de forma relativa entre si
- Link: <https://developer.android.com/training/constraint-layout/index.html>
- Por default no Android Studio 3.0+
- Melhor para construir layouts via editor gráfico
- Baseado em Constraints (restrições):
 - Conexão ou Alinhamento entre Views / Layout Pai
 - Definem a posição do componente: no mínimo vertical e horizontalmente

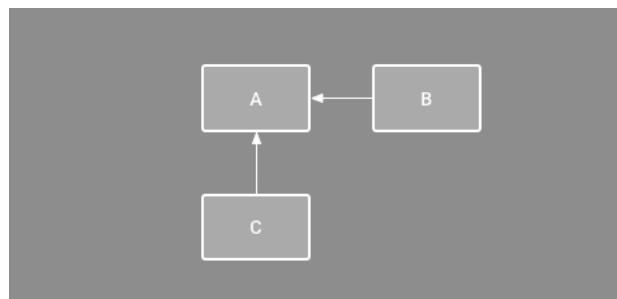
ConstraintLayout

- "A" está conectado à esquerda, pode-se definir essa distância



ConstraintLayout

- "B" estará sempre à direita de "A", mas não há restrição vertical
- "C" estará sempre abaixo de "A", mas não há restrição horizontal



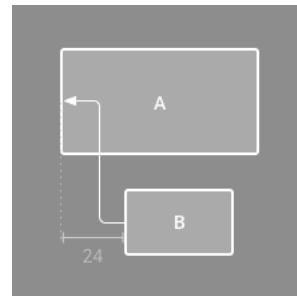
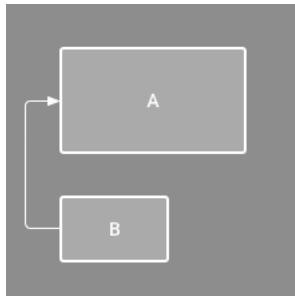
Prof. Razer A N R Montaño

2017

19

ConstraintLayout

- "B" está alinhado horizontalmente a "A"
- "B" está alinhado horizontalmente a "A" com um deslocamento de 24dp (basta arrastar o componente "B" para aumentar ou diminuir o deslocamento)



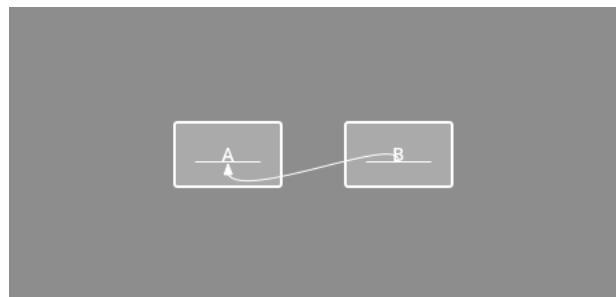
Prof. Razer A N R Montaño

2017

20

ConstraintLayout

- Alinhar conforme a linha de texto (baseline)
- O texto de "B" estará alinhado ao texto de "A"



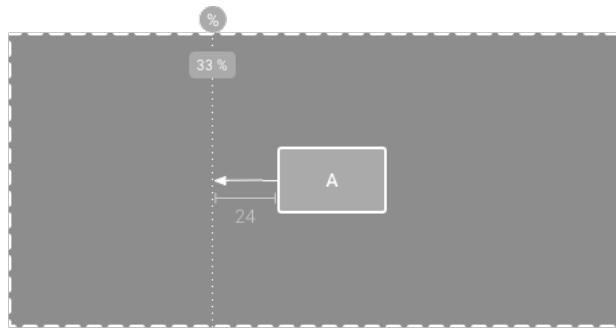
Prof. Razer A N R Montaño

2017

21

ConstraintLayout

- Alinhar conforme uma linha guia (invisível)



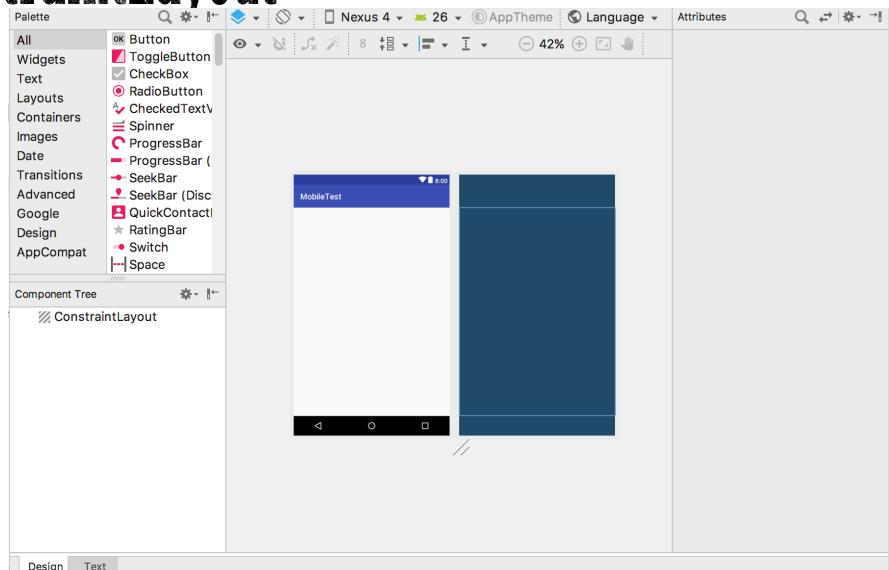
Prof. Razer A N R Montaño

2017

22

ConstraintLayout

- No AS

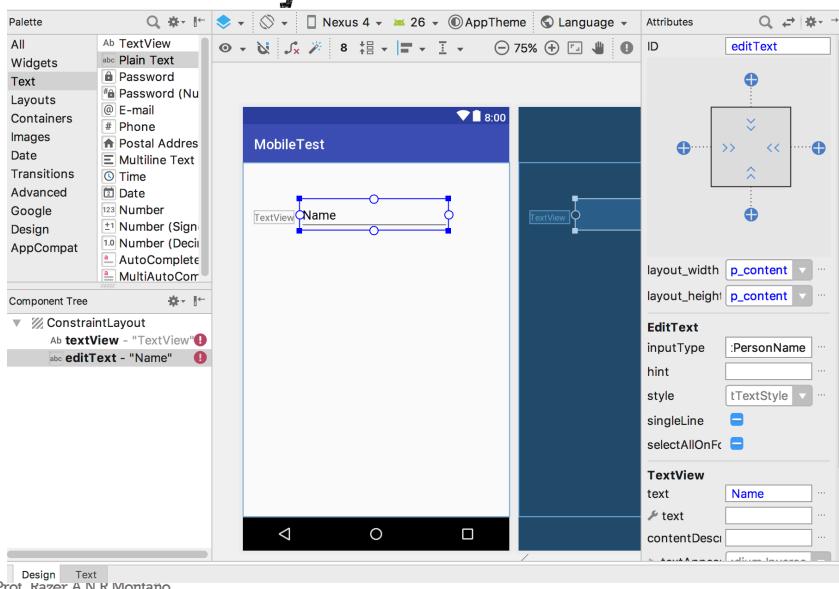


Prof. Razer A N R Montaño

2017

23

ConstraintLayout

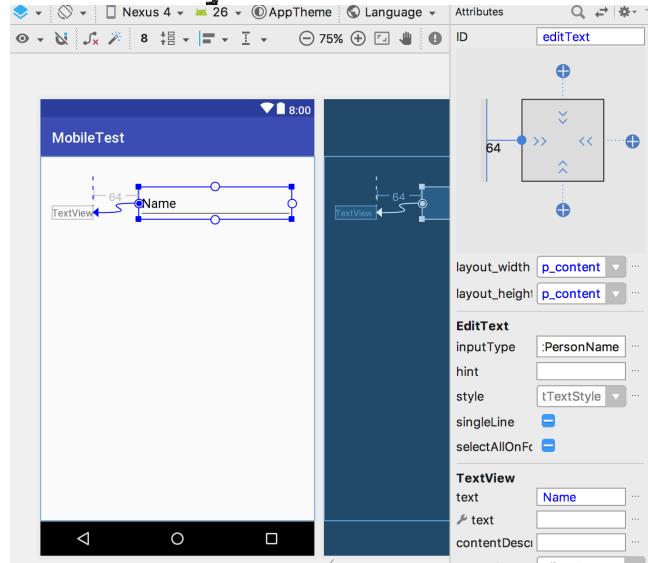


Prof. Razer A N R Montaño

2017

24

ConstraintLayout

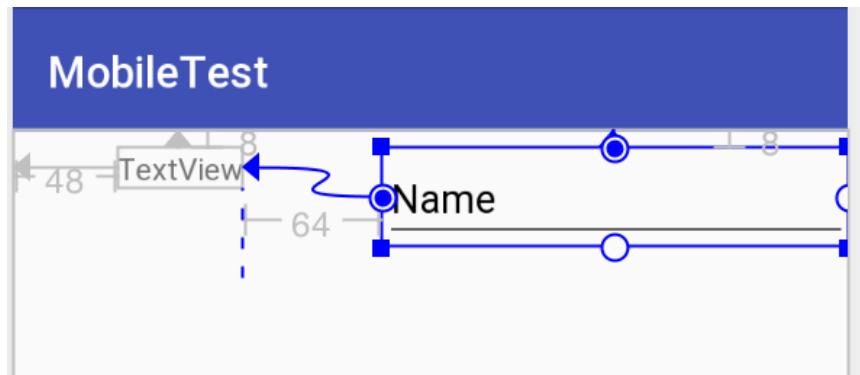


Prof. Razer A N R Montaño

2017

25

ConstraintLayout

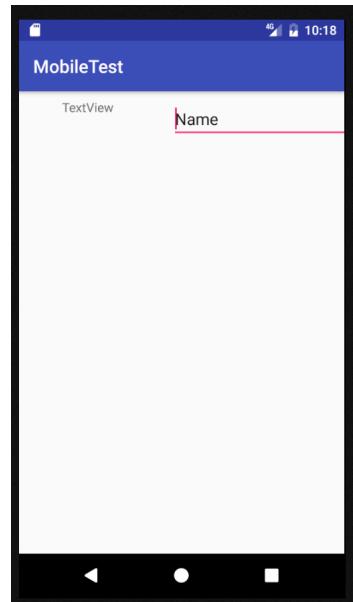


Prof. Razer A N R Montaño

2017

26

ConstraintLayout

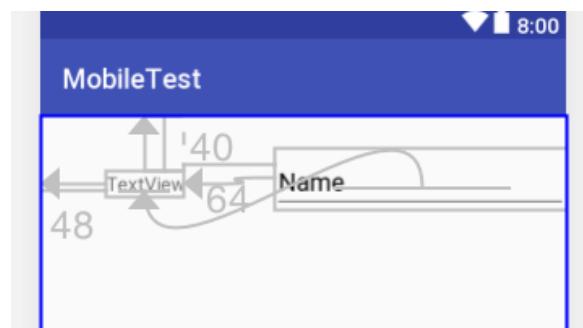


Prof. Razer A N R Montaño

2017

27

ConstraintLayout

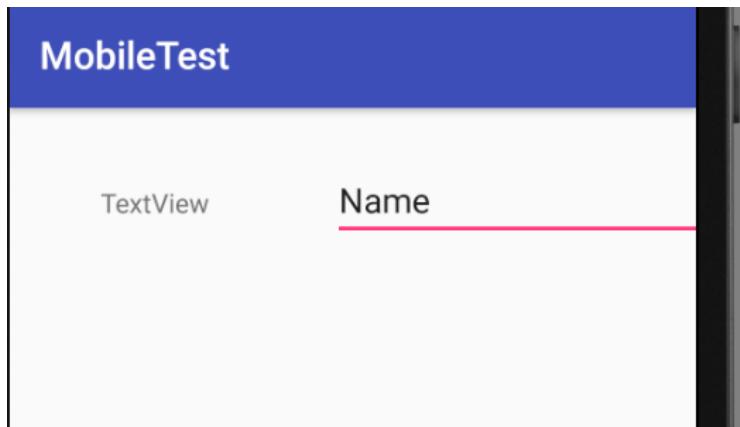


Prof. Razer A N R Montaño

2017

28

ConstraintLayout



Prof. Razer A N R Montaño

2017

29



ListView

Prof. Razer A N R Montaño

2017

ListView

- Elemento usado para mostrar uma lista de elementos de forma vertical, com rolagem.
- Link: <https://developer.android.com/reference/android/widget/ListView.html>
- Os elementos vêm de um ListAdapter
 - Link: <https://developer.android.com/reference/android/widget/ListAdapter.html>

Prof. Razer A N R Montaño

2017

31

ListView

- Exemplo no Layout

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/lista"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent">  
  
</ListView>
```

Prof. Razer A N R Montaño

2017

32

ListView

- Para preencher, deve-se usar um *ListAdapter*
- No exemplo aqui, será usada uma classe Contato (nome, telefone) e um *ArrayAdapter*
 - Link: <https://developer.android.com/reference/android/widget/ArrayAdapter.html>
- Deve-se:
 - Obter uma lista de Contatos
 - Criar um *ArrayAdapter* com um layout e com os dados da lista

```
ArrayAdapter<Contato> adapter = new ArrayAdapter<Contato>(
    this, android.R.layout.simple_list_item_1, contatos);
```

- Onde os parâmetros do construtor são:
 - Contexto, que é a activity (**this**)
 - Layout: **simple_list_item_1** apresenta o conteúdo do **elemento.toString()**. Para outros formatos pode ser necessário criar um *Adapter* customizado
 - Lista de elementos

ListView – Classe Contato

```
public class Contato {
    private String nome;
    private String telefone;
    public Contato() { }
    public Contato(String nome, String telefone) {
        this.nome = nome;
        this.telefone = telefone;
    }
    public String toString() {
        return "Nome: " + this.nome + " - Fone: " + this.telefone;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }
}
```

ListView – Activity

```

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

    List<Contato> contatos = new ArrayList<Contato>();
    Contato c = new Contato("Razer", "9999-1010");
    contatos.add(c);
    c = new Contato("Jaime", "9999-2020");
    contatos.add(c);

    ArrayAdapter<Contato> adapter = new ArrayAdapter<Contato>(
        this, android.R.layout.simple_list_item_1, contatos);

    ListView v = (ListView) findViewById(R.id.lista);
    v.setAdapter(adapter);

}
}

```

Prof. Razer A N R Montaño

2017

35

ListView

- Para tratar o evento de click em um elemento do ListView, adiciona-se um **OnItemClickListener** no método **onCreate**

```

ListView v = (ListView) findViewById(R.id.lista);
v.setAdapter(adapter);
v.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
                               int position, long id) {
            Contato c = (Contato)parent.getItemAtPosition(position);
            clicado(c);
        }
    }
);

```

Prof. Razer A N R Montaño

2017

36



Exercícios

1. Criar uma tela com ListView de Contatos. Carregar com várias informações. Ao ser clicado, deve-se mostrar o nome do contato clicado

38

Outros

Outros ViewGroup's

- GridLayout
- GridView
- AbsoluteLayout
- FrameLayout
- Etc