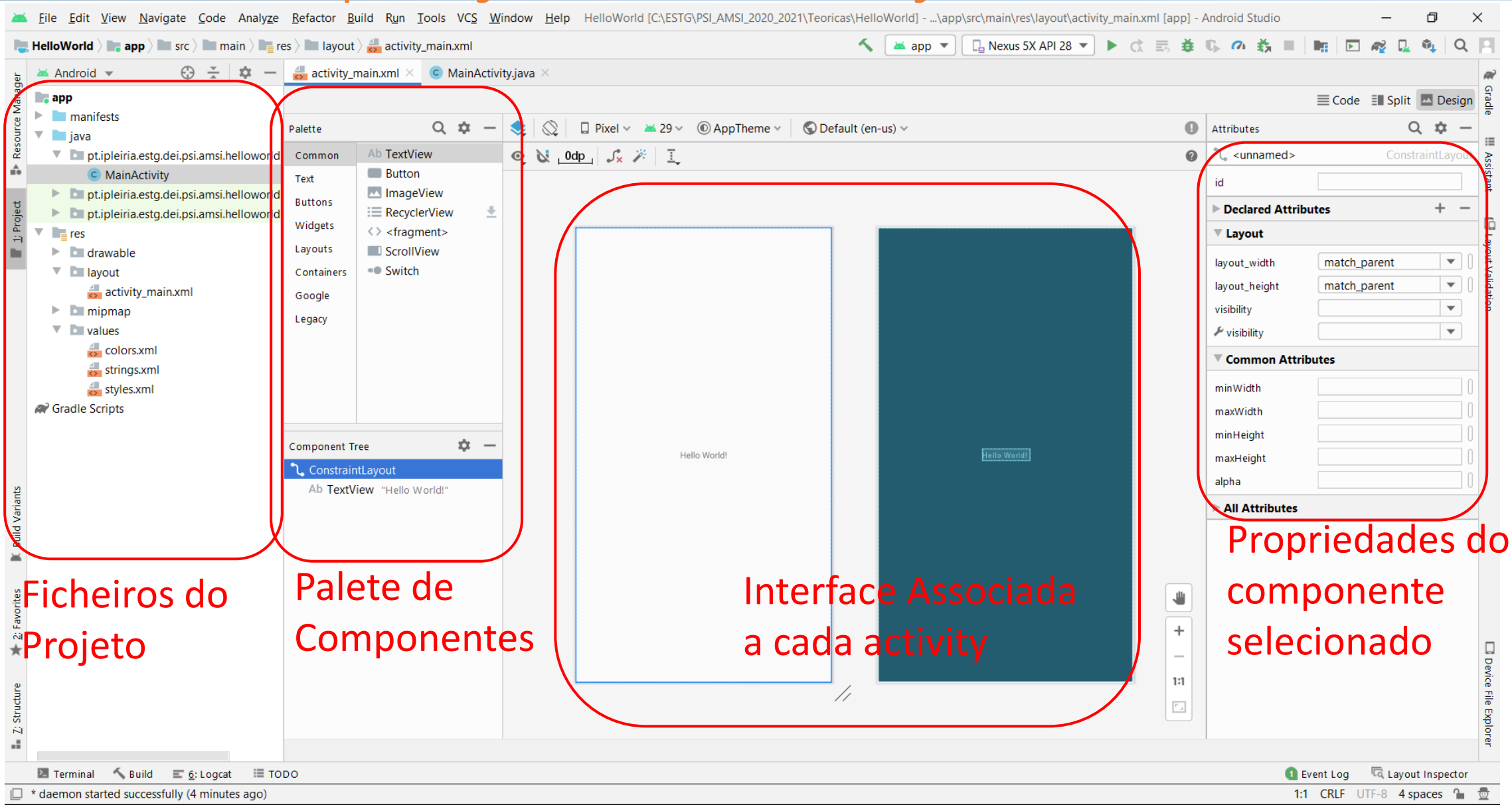
Composição de um Projeto em Android

Desenho de Interfaces Gráficas

Layouts

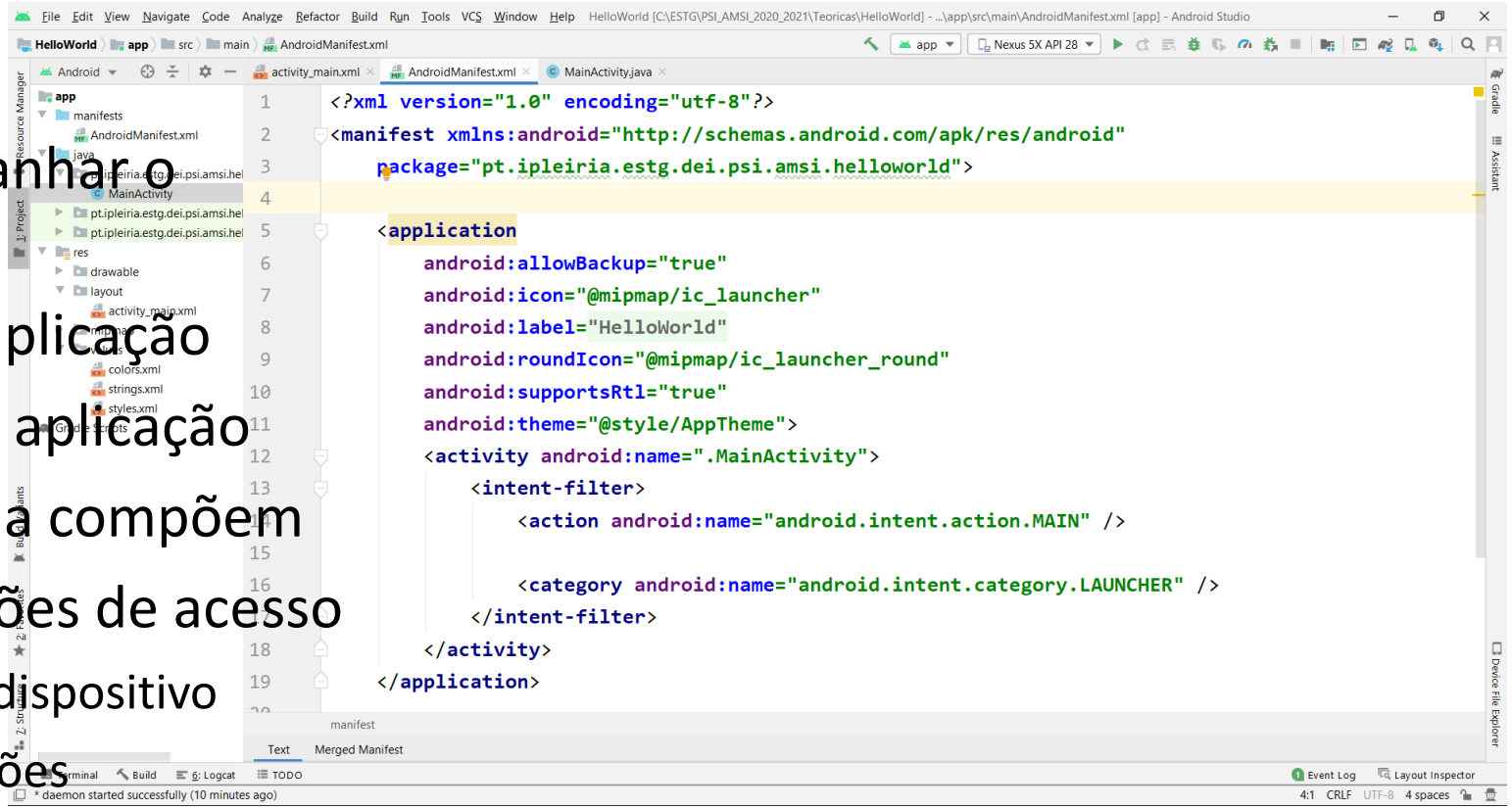
Sónia Luz, sonia.luz@ipleiria.pt
David Safadinho, david.safadinho@ipleiria.pt
Departamento de Engenharia Informática
Escola Superior de Tecnologia e Gestão
Instituto Politécnico de Leiria
1º Semestre - 2021/2022

Composição de um Projeto em Android



Manifests

- AndroidManifest.xml
  - Deve obrigatoriamente acompanhar o projeto
  - Descreve as características da aplicação
  - Contém o nome do *package* da aplicação
  - Identifica os componentes que a compõem
  - Permite a definição de permissões de acesso
    - a funcionalidades protegidas do dispositivo
    - para interagir com outras aplicações
    - de outras aplicações sobre esta
    - (`<permission>`, `<uses-permission>`, …)



Java

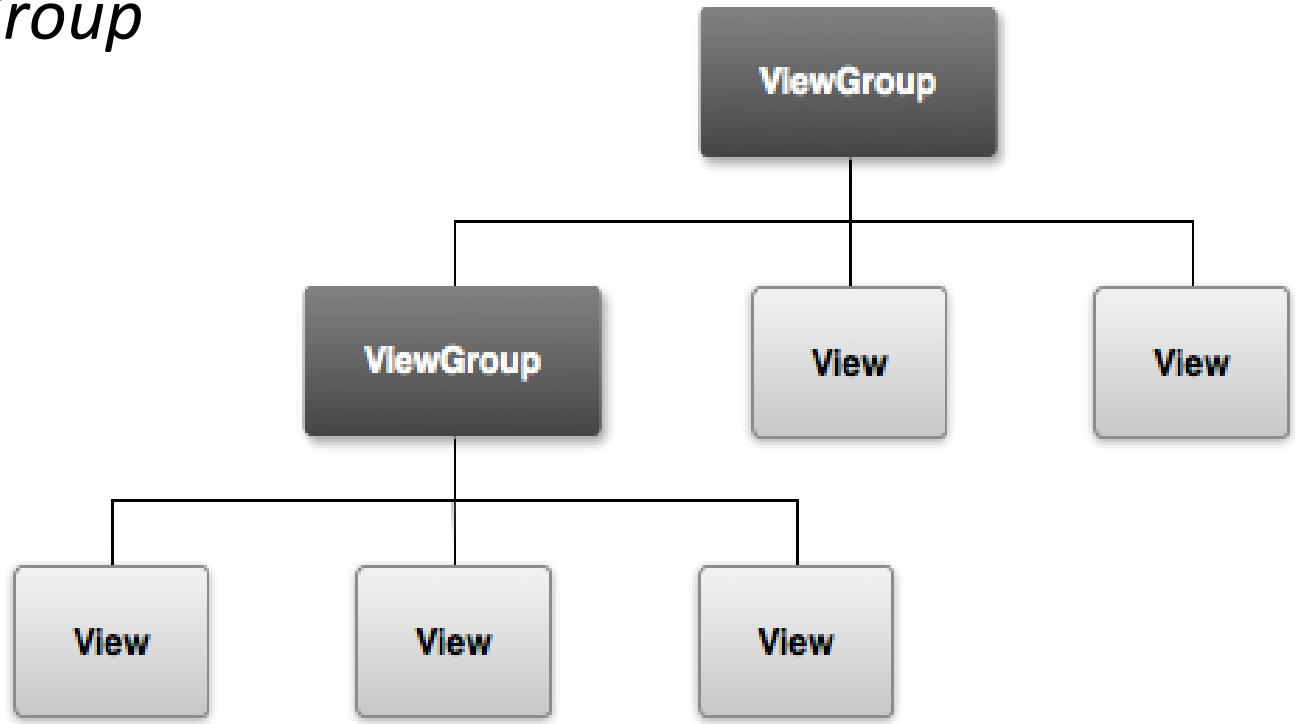
- Diretoria que contém o código da aplicação
  - Ficheiros .java
- Inclui todas as atividades definidas para a aplicação
  - Uma atividade definida como principal e que é executada no lançamento da aplicação
- Inclui o código dos testes automatizados pelo Android
  - Package de testes unitários: *test*
  - Package de testes instrumentados: *androidTest*

Res

- Contém os recursos da aplicação que não são código
  - Drawable
    - Contém as imagens usadas na aplicação
      - É possível separar as imagens em pastas, considerando o tamanho e resolução de cada dispositivo
  - Layout
    - Contém ficheiros .xml que definem o UI (ecrãs) da aplicação - *layouts*
  - Mipmap
    - Contém os ícones da aplicação
  - Values
    - Contém outros ficheiros .xml que definem recursos do tipo *string* e *color*
      - Alguns podem ser utilizados para a internacionalização da aplicação

Desenho de Interfaces Gráficas

- Interface de utilizador (UI) de uma APP Android
  - Obtido através de uma hierarquia de objetos do tipo
    - *ViewGroup*
    - *View*



Desenho de Interfaces Gráficas

- *ViewGroup (layout)*
  - Não são visíveis
  - Contêm outros objetos do tipo *ViewGroup* ou *View*
  - Apenas definem a disposição (horizontal, vertical, …) pelo qual os objetos *view* se encontram dispostos

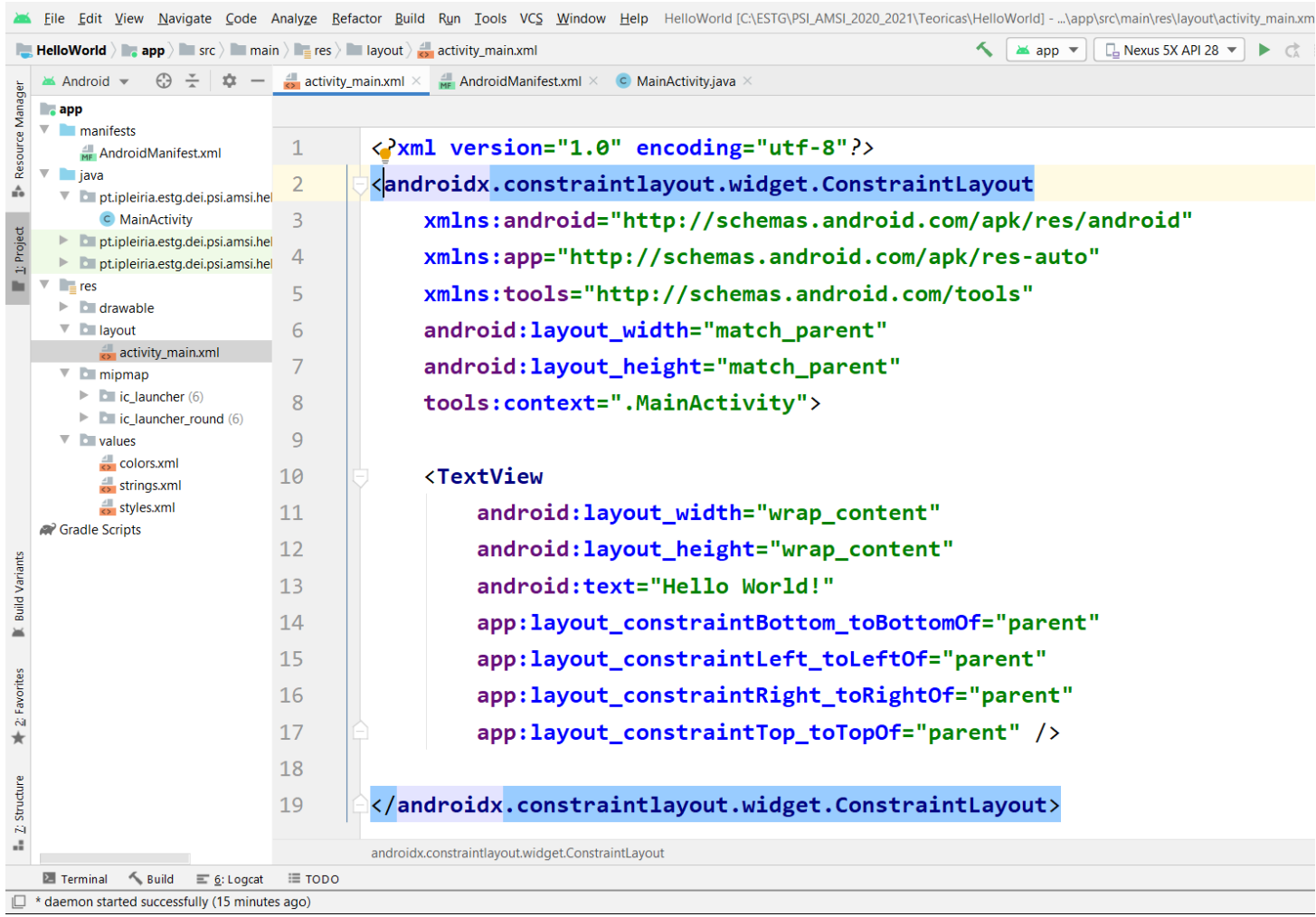
- *View (widgets)*
  - Componentes que já incluem algumas funcionalidades
  - Permitem interação com o utilizador
    - Ex. botões, campos de texto, listas, …

Layouts - Definição

- É uma arquitetura para organizar a UI no Android
- Pode ser construído de duas formas *(ver Android Studio)*:
  - Forma visual
    - utilizando os recursos de *drag and drop* de *views* (*widgets*)
  - Forma declarativa
    - utilizando o ficheiro XML que representa o layout
    - declaração em ficheiros XML permite melhor separação entre a apresentação e o código que implementa os comportamentos
- *Views* e *ViewGroups* podem ser construídos em tempo de execução

Layouts - Localização

- Todos os ficheiros têm extensão XML
- Localizados na pasta ***res/layout*** do projeto
- Registados na classe R



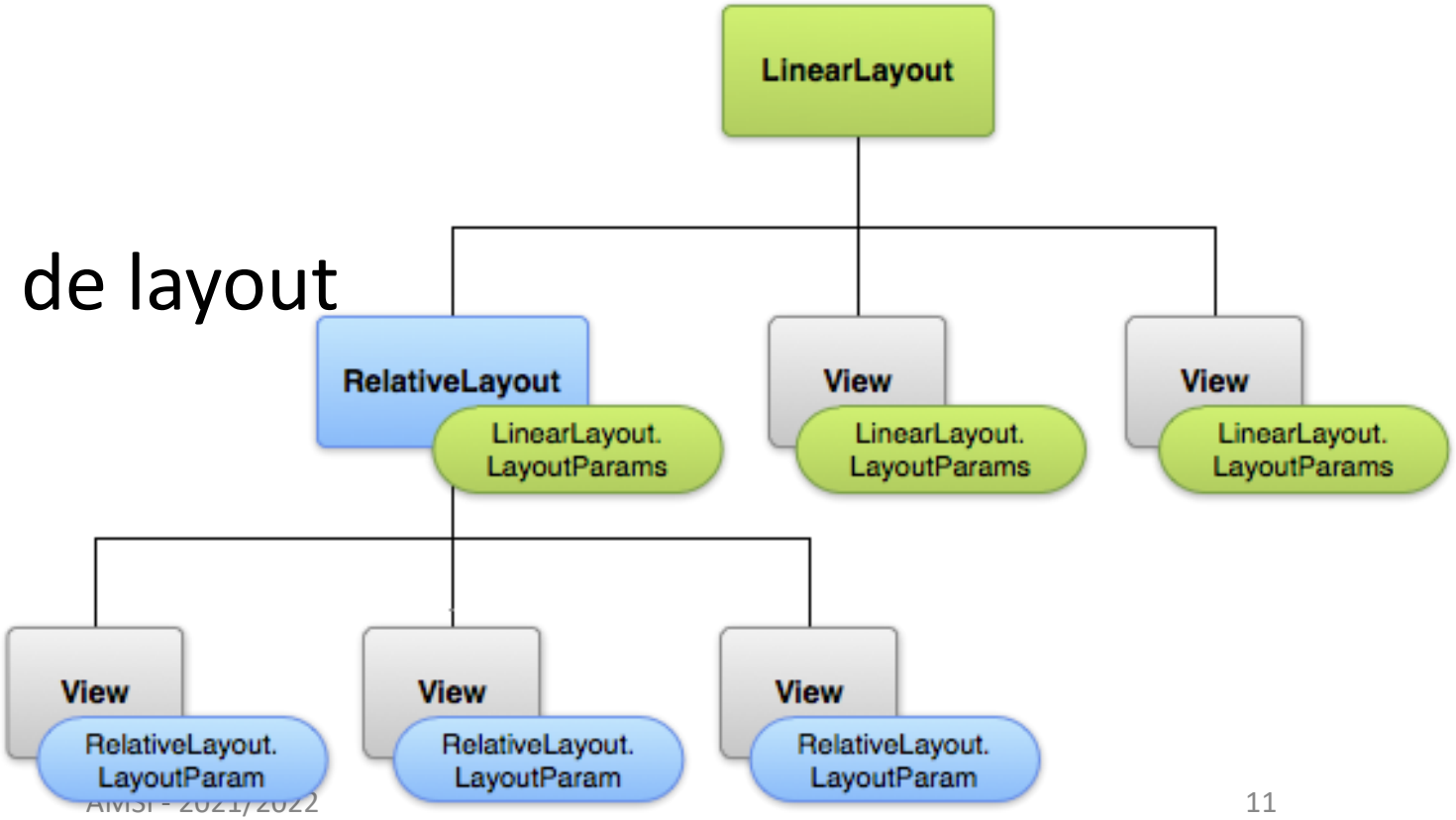
Atribuição do Layout à atividade

- Atribuir o GUI específico para a interface
  - Através do método ***setContentView***()
    - Indicando a referência única definida na classe *R* (*R.java)*
  - Invocado dentro do método ***onCreate***() da atividade
  - Não existe uma relação direta entre o nome do ficheiro e o nome da atividade

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

Parâmetros de um Layout

- Atributos designados *layout_something*
  - Definem parâmetros para a *View* de acordo com o layout onde estão inseridos
- Cada *ViewGroup*
  - define os parâmetros de layout para as *child Views*

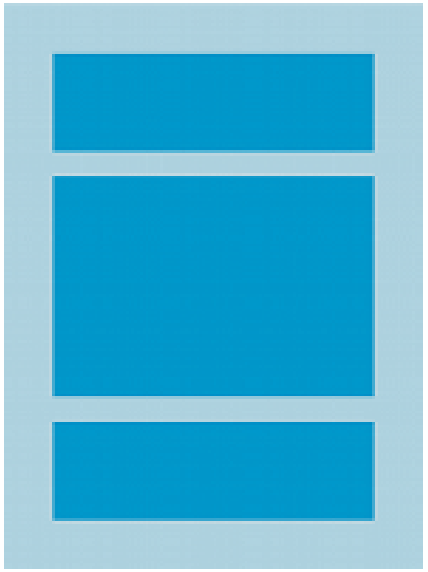


Parâmetros de um Layout

- Entre estes atributos estão os que permitem a definição de largura (*layout_width*) e altura (*layout_height*)
  - Normalmente usam-se as constantes:
    - *wrap_content* – ajustar o tamanho de acordo com o conteúdo
    - *match_parent* – preencher o tamanho máximo permitido pelo layout
    - Valor exato é possível (mas não é recomendado)
- Unidades de medida disponíveis: px (*pixels*), dp (*density-independent pixels*), sp (*scaled pixels*), in (*inches*), mm (*millimiters*)

Tipos de Layouts – 3 tipos/grupos mais comuns

- *Layout Linear*
  - organiza os *childs* numa única linha horizontal ou vertical
  - se o comprimento da janela exceder o tamanho do ecrã é criada uma *scrollbar*



Tipos de Layouts – 3 tipos/grupos mais comuns

- *Layout Relativo*
  - permite especificar a localização de objetos *child*
    - relativos entre si
      - ex.: *child* A à esquerda de *child* B
    - relativos aos *parents*
      - alinhados na parte superior do *parent*

Tipos de Layouts – 3 tipos/grupos mais comuns

- *Visualização Web (Web view)*
  - exibe páginas da Web

Tipos de Layouts

- Tipos de layout específicos, com diferentes características
  - *LinearLayout*
  - *RelativeLayout*
  - *ConstraintLayout*
  - *TableLayout*
  - *GridLayout*
  - CoordinatorLayout
  - …

LinearLayout

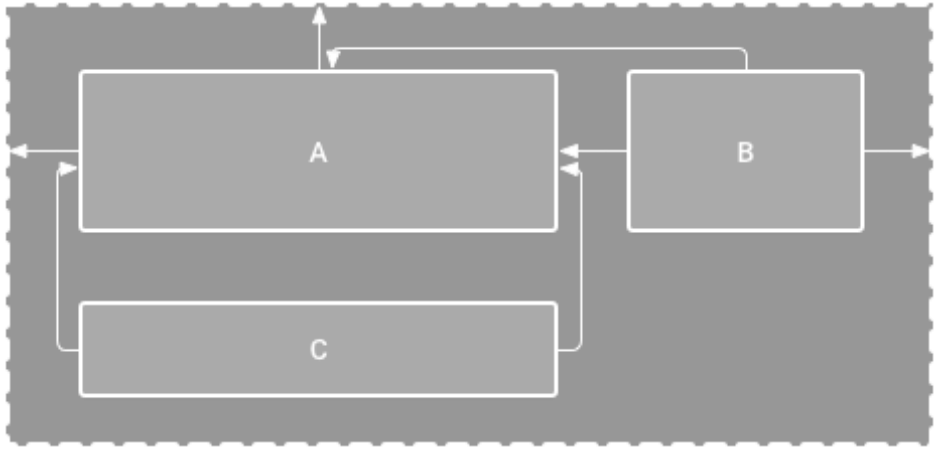
- É um *ViewGroup* que dispõe as *child views* seguidas umas às outras em sequência vertical ou horizontal
  - Atributo `android:layout_orientation`
- Respeita as margens entre as *child views* e alinhamento (ao centro, à esquerda, à direita)
  - Atributo `android:layout_gravity`
- Permite atribuir pesos, individualmente, para que estas *views* possam ocupar o restante espaço do layout
  - Atributo `android:layout_weight`

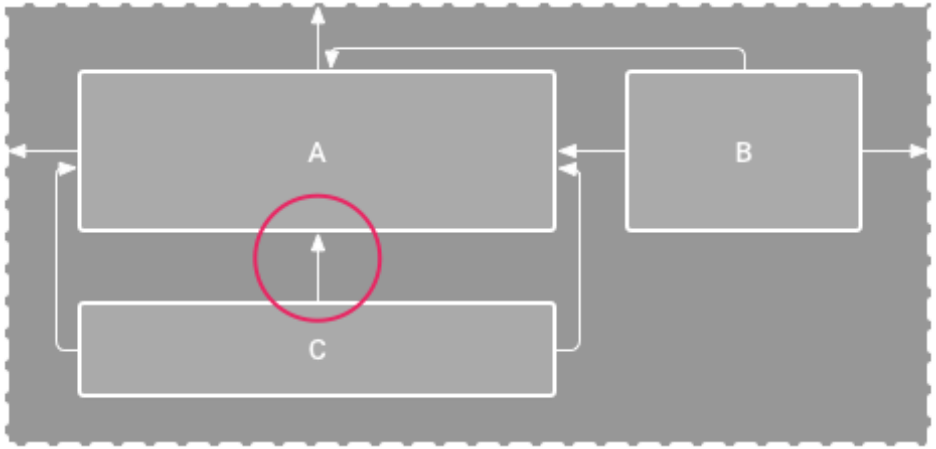
RelativeLayout

- Permite definir a posição dos componentes
  - Relativa a outros componentes presentes no layout
- Relações definidas automaticamente
  - Com base na ordem de adição e local (no Android Studio)
- Propriedades principais para definição em xml
  - `android:layout_below`
    - Posiciona este componente debaixo de outro componente
  - `android:layout_toEndOf`
    - Alinha o início do componente pelo fim de outro componente (prende à esquerda)
  - `android:layout_alignEnd`
    - Alinha o fim do componente pelo fim de outro componente (prende à direita)

ConstraintLayout

- Permite definir layouts grandes e complexos com uma hierarquia *flat* (sem necessidade de sub layouts)
- Similar ao *RelativeLayout*
  - Todas as *views* se relacionam entre si e o seu *parent*





- Usa as vantagem das ferramentas visuais do *Layout Editor*
  - Mais fácil construir o layout por *drag-and-drop* do que pelo xml

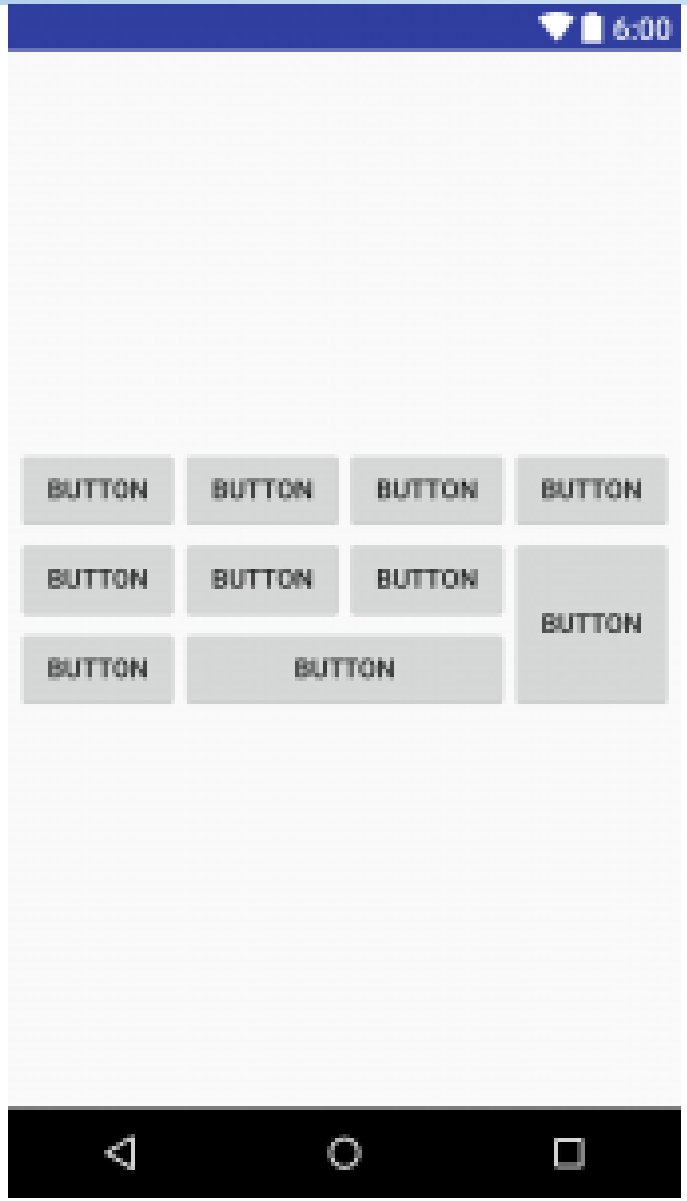
TableLayout

- Organiza as *child views* em linhas e colunas
- Cada *child* é representado pelo componente *TableRow*
  - Permitindo que uma ou mais células sejam adicionadas horizontalmente
  - Cada célula só pode conter uma *view*

- Caso os atributos `android:layout_width` e `android:layout_height` não sejam declarados: (para cada componente)
  - Largura será ajustada para ***match_parent***
  - Altura será ajustada para ***wrap_content***

GridLayout

- Tem o intuito de juntar dois layouts,
  - *LinearLayout* e *TableLayout*
- Composto por um conjunto de linhas e colunas
- Poderá ter orientação
  - na horizontal e na vertical



CoordinatorLayout

- Um novo layout
  - Introduzido na versão 24.1
    - com o Android Design Support Library
- Tem como objetivo e filosofia
  - Coordenar as *views* que contém
- Permite definir diferentes interações entre cada *child*
  - Através de *Behaviors* (comportamentos)
  - É também possível definir âncoras para as *childs*

Layouts

- Vamos experimentar no Android Studio!

Fontes e Mais Informação

- Estrutura de um Projeto Android Studio
  - https://developer.android.com/studio/intro
- Interface do Utilizador no Android
  - https://developer.android.com/guide/topics/ui/index.html
- *ViewGroups* (*Layouts*)
  - http://developer.android.com/guide/topics/ui/declaring-layout.html

Próximo Tema:

Activities e Intents

- Atividades
  - http://developer.android.com/guide/components/activities.html
- Intents e filtros de Intents
  - http://developer.android.com/guide/components/intents-filters.html
- Gerir o ciclo de vida de uma atividade
  - https://developer.android.com/guide/components/activities/activity-lifecycle
- Interagir com outras Apps
  - http://developer.android.com/training/basics/intents/index.html
- Partilhar dados simples
  - http://developer.android.com/training/sharing/index.html