

Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



android

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

 Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Android

---

- Sistema Operacional e Plataforma de Desenvolvimento para Dispositivos Móveis com maior taxa de crescimento
  
- FOSS (*Free Open Source Software*)
  - Usa o kernel do Linux
  - Usa diversas bibliotecas livres
    - *SQLite, LibC, WebKit*
    - *OpenSSL, OpenGL*
    - *zLib, FreeType*



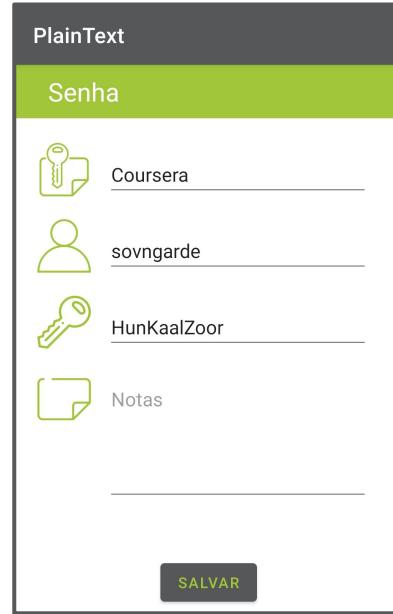
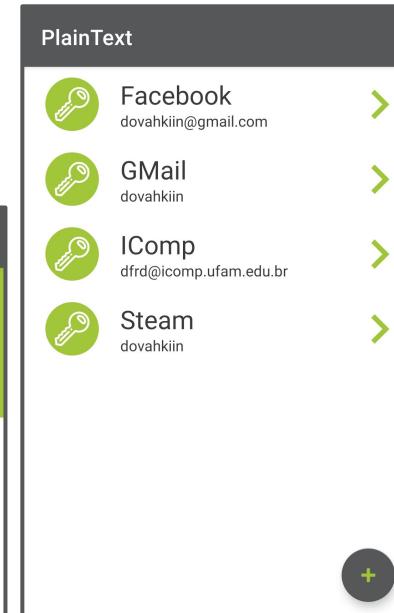
# Desenvolvimento

- Aplicativos Android podem ser desenvolvidos nas linguagens:
  - Java
    - Recomendada para quem está aprendendo a programar
    - Além de ser usado pelo Android, é usado em milhares de outros projetos
    - Comunidade imensa de programadores. Maior documentação e oportunidades
  - Kotlin
    - Linguagem oficial do Android, desde 2017
    - Possui uma sintaxe mais limpa que o Java, dentre outras vantagens
    - Fácil de aprender, principalmente se você já conhece Java
    - Compilado para o mesmo bytecode do Java, executado pela máquina virtual java (JVM)
    - Usado basicamente apenas pelo Android
  - C++
    - Código é executado diretamente pelo processador do dispositivo
    - Permite utilizar bibliotecas existentes desenvolvidas em C/C++

# Sumário

- Android Studio
  - Instalação
  - Hello World
- Interface Gráfica
- Activities e Intents
  - Preferências
  - Listas
- Banco de Dados

## □ Aplicativo de Exemplo



# Projeto DevTITANS

---

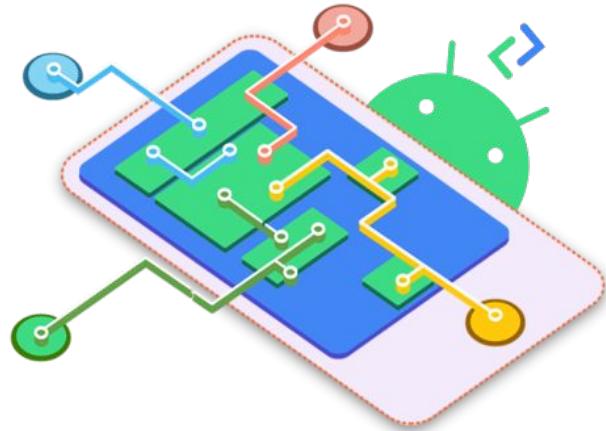


- ▣ Capacitação e Desenvolvimento em Tecnologias Android para Sistemas Embarcados
  - Aprender não só a criar um app
  - Mas aprender a criar/personalizar um Android novo
- ▣ Projeto do IComp/UFAM em parceria com a **Motorola** e **Flextronics**
- ▣ Objetiva a formação de **recursos humanos de alta qualidade**
  - Com conhecimentos técnicos avançados e especializados
  - Na criação, personalização e manutenção da plataforma Android
  - Em dispositivos embarcados

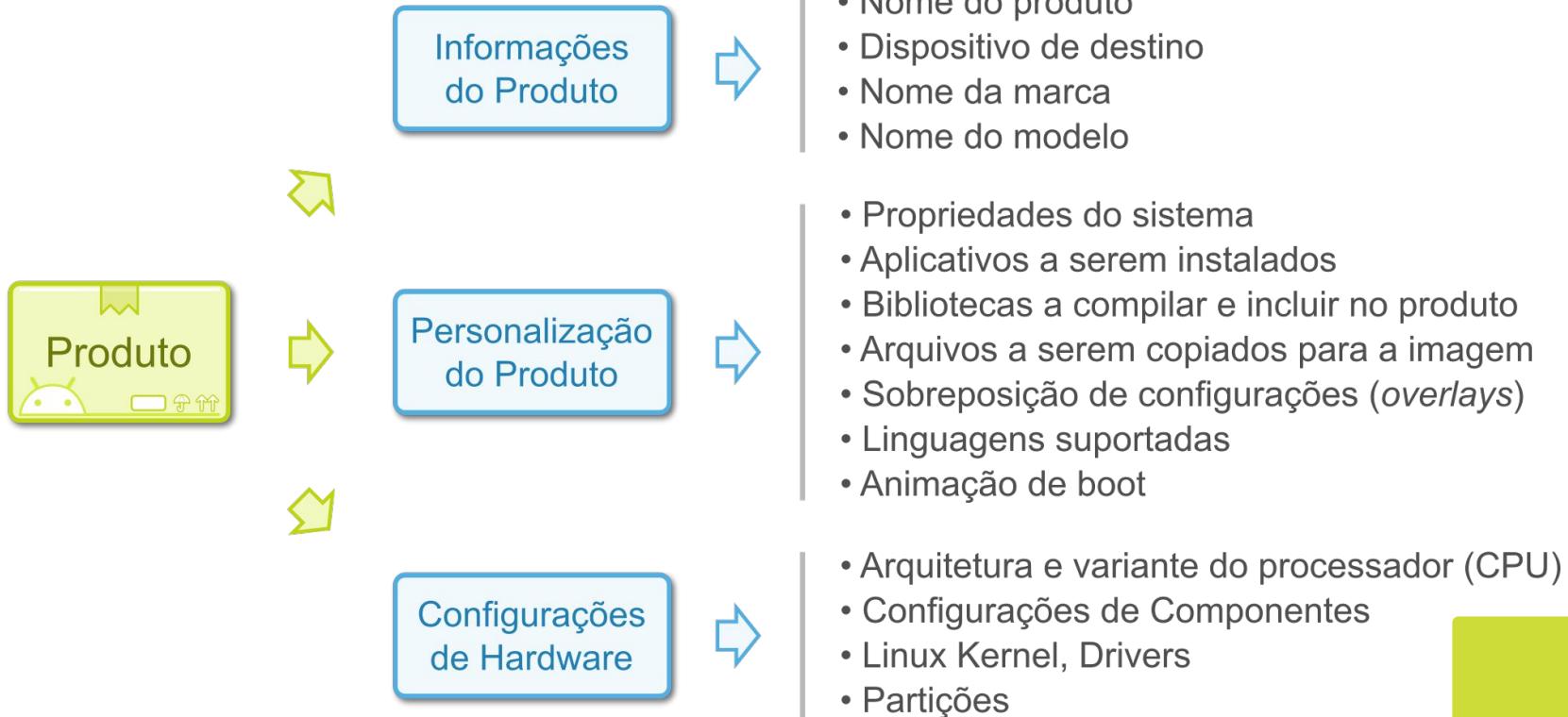
# Android Open Source Project – AOSP



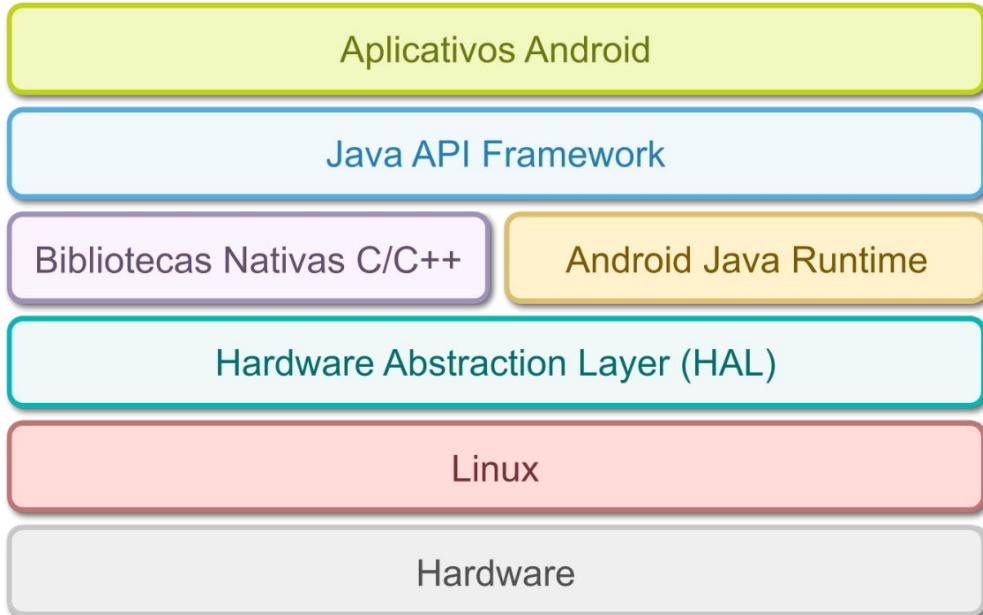
- O código-fonte do Android é **aberto** e permite personalizá-lo
- O AOSP é **ponto de partida** para a personalização e criação de um novo sistema baseado em Android
  - Informações e documentações
  - Repositório de código-fonte
  - Testes de validação e de compatibilidade
- Principal responsável por garantir a **compatibilidade** entre os dispositivos
  - Todos os fabricantes se baseiam no mesmo código-fonte



# AOSP – Produto: Personalização do Sistema

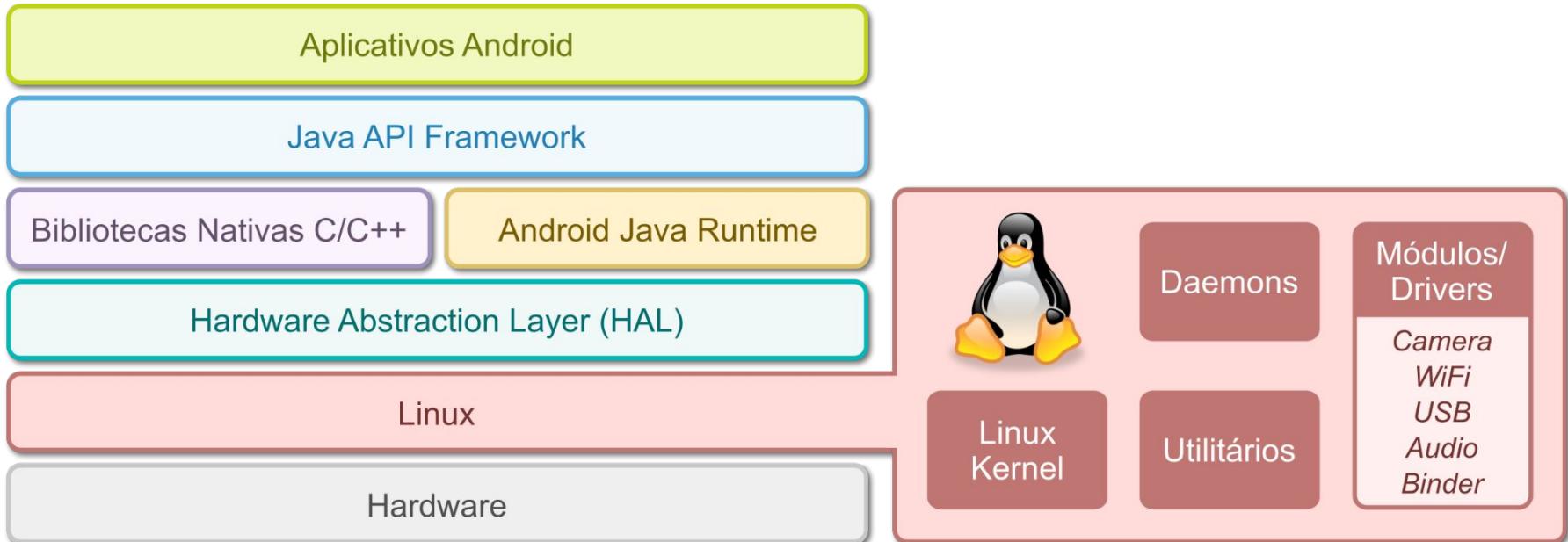


# Camadas do Android

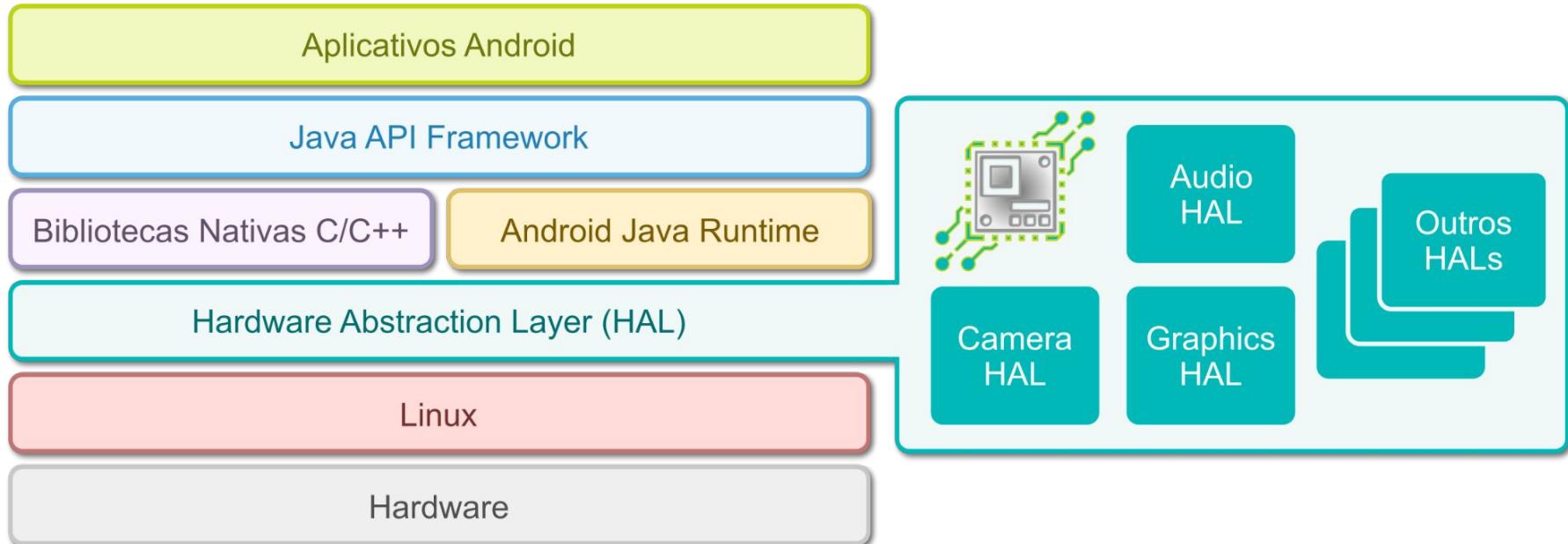


- Para personalizar um Android, é preciso conhecê-lo bem
- O Android é dividido em Camadas
  - A primeira delas, de mais baixo nível, é o *hardware*
    - Parte física do dispositivo

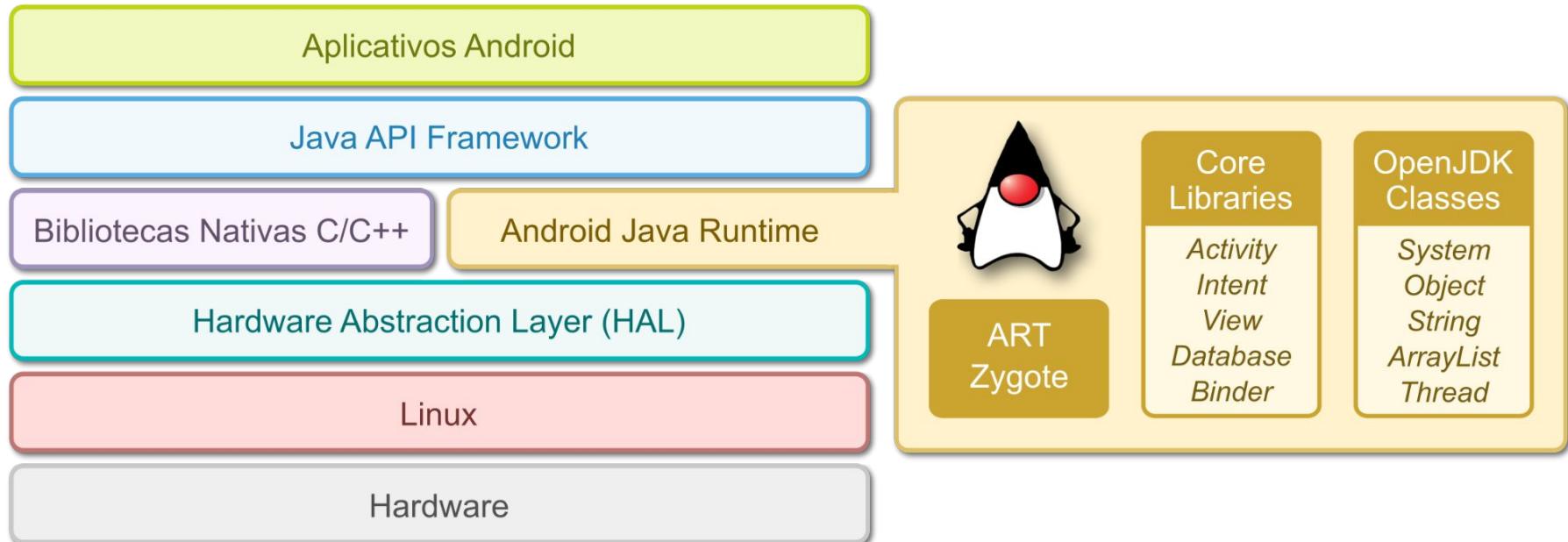
# Camadas do Android



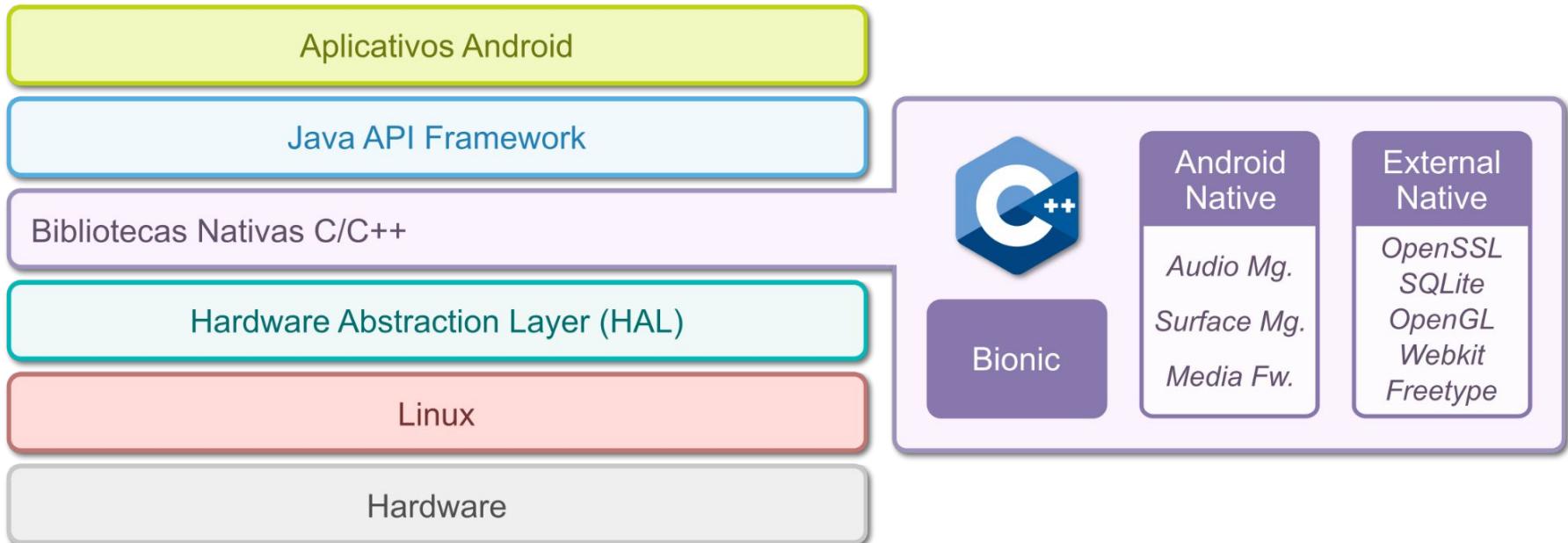
# Camadas do Android



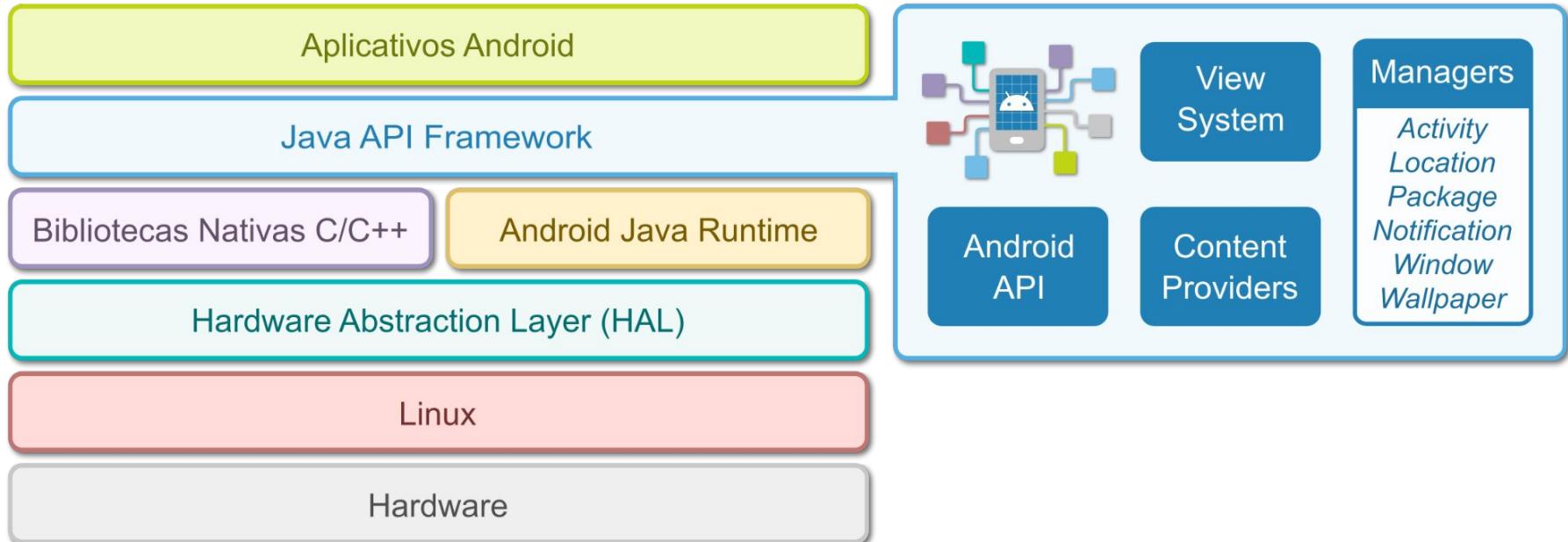
# Camadas do Android



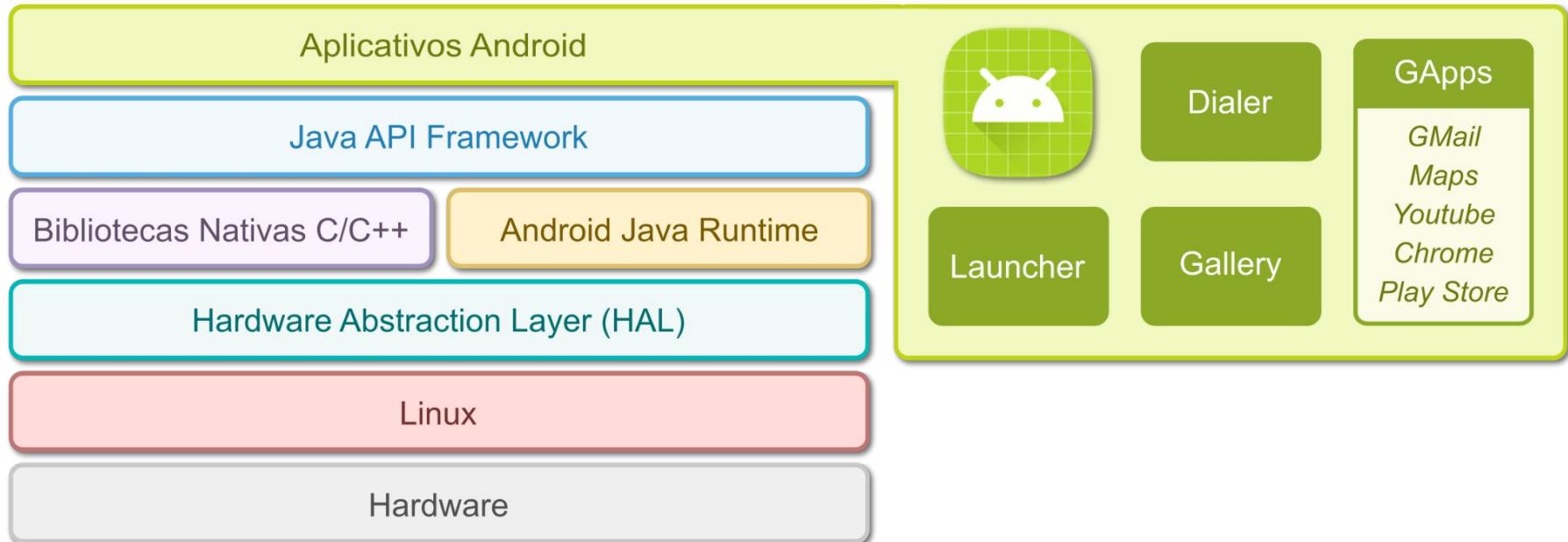
# Camadas do Android



# Camadas do Android



# Camadas do Android



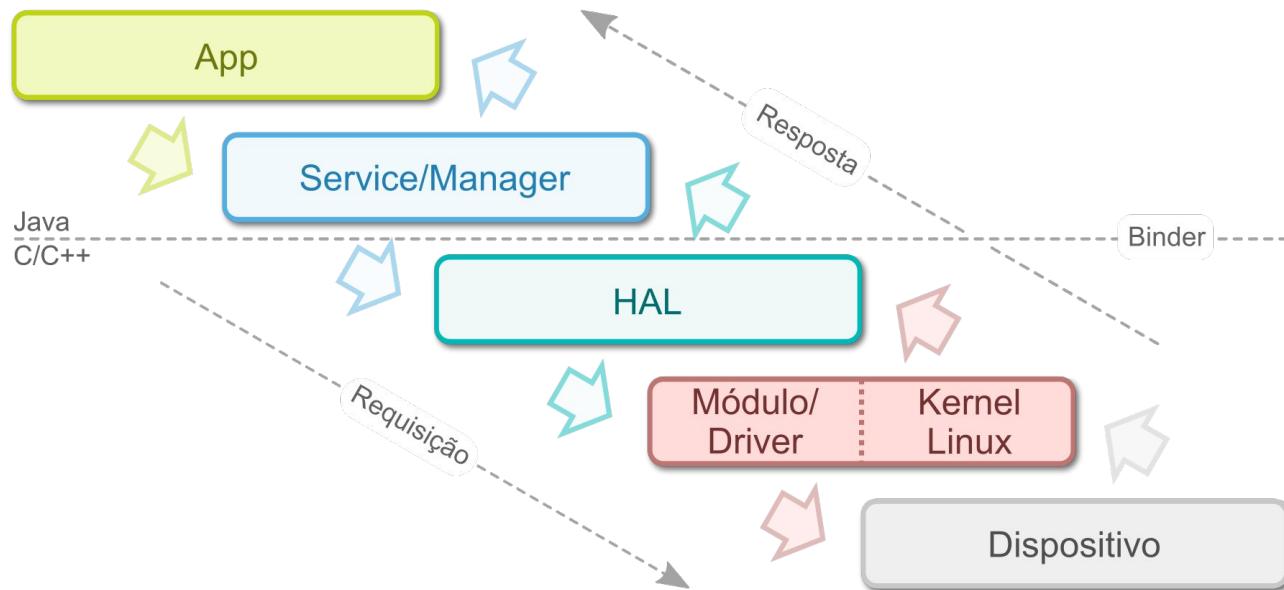
# Do Hardware ao App



- Prática



# Do App ao Hardware, via HAL e Binder



# Inicialização do Android

---



Power  
On

# Projeto DevTITANS – 1a Turma!



# Projeto DevTITANS

---



- Aluno de PP/TAP pode?
  - Sim
  - Quando chegarmos na parte que precisa de Android, vocês já estarão prontos como conteúdo da disciplina até o momento (Java)
  
- Inscrições
  - Início de Fevereiro
  - Aulas começam no final de Fevereiro

Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



# android studio

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

 Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Android Studio

---

- Modo recomendado de desenvolvimento desde 12/2014
  - Substitui Eclipse ADT
- Contém:
  - IntelliJ IDE
  - Android SDK Tools
  - Android Platform-tools
  - Imagem do Sistema Android para o Emulator

# Download

- <https://developer.android.com/studio>

The image shows two side-by-side screenshots. On the left is the official Android Studio Flamingo download page, featuring a large 'Android Studio' logo and a 'Download Android Studio Flamingo' button. On the right is a screenshot of the Android Studio IDE showing a code editor with Kotlin code for a Composable function named 'TopicSelection'. Below the code editor is the 'App Quality Insights' dashboard, which displays various app issues and their details.

Android Studio

Get the official Integrated Development Environment (IDE) for Android app development.

Download Android Studio Flamingo ↓

Read release notes

```
nowinandroid main
```

```
@Composable
private fun TopicSelection(
    onboardingUiState: OnboardingUiState.Shown,
    onTopicCheckedChanged: (String, Boolean) -> Unit,
    modifier: Modifier = Modifier,
) = trace( sectionName = "TopicSelection" ) {
    val lazyGridState = rememberLazyGridState()
    val topicSelectionTestTag = "forYou:topicSelection"

    TrackScrollJank(scrollableState = lazyGridState, stateName = topicSelectionTestTag)

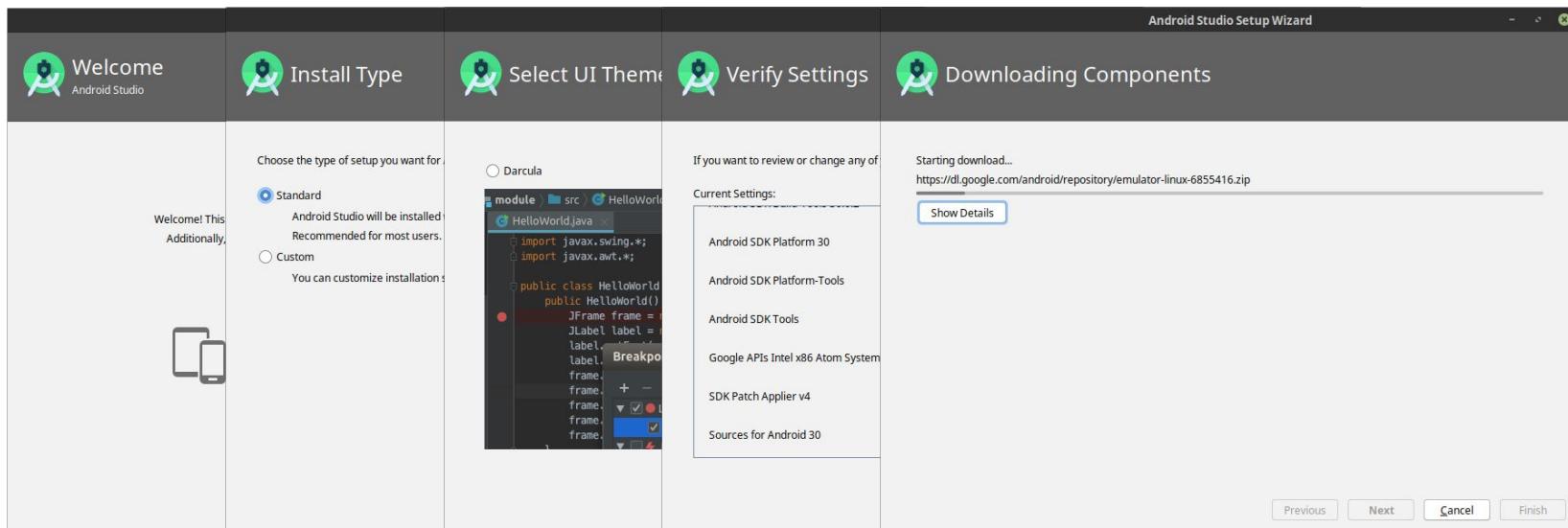
    LazyHorizontalGrid(
        state = lazyGridState,
        rows = GridCells.Fixed( count = 5 ),
        horizontalArrangement = Arrangement.spacedBy(12.dp),
        verticalArrangement = Arrangement.spacedBy(12.dp),
        contentPadding = PaddingValues(24.dp),
        modifier = modifier
            .heightIn(max = max(240.dp, with(LocalDensity.current) { 240.sp.toDp() } ))
            .fillMaxWidth()
            .testTag(topicSelectionTestTag)
    )
}
```

App Quality Insights

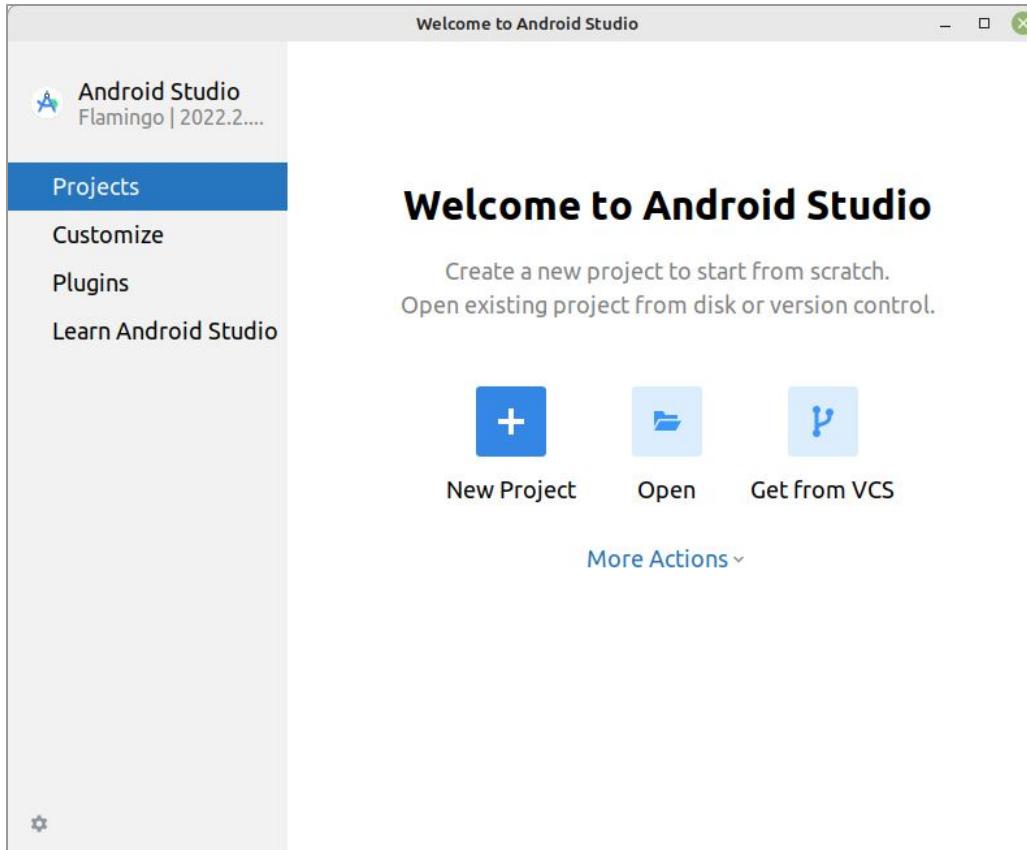
Issue	Events	Users	Details
↳ (has extras)	24	7	Google 29%
↳ ..)	12	4	Other 8%
✗ apps.nowinandroid/u0a	10	4	
ⓘ ...ed to system_server and system apps only, unless they are annotated with @Readable.	7	3	Android Versions
✗ lang.IndexOutOfBoundsException - Index: 4, Size: 4	1	1	Android (12) 58%
✗ sqlite.SQLiteFullException - database or disk is full (code 13 SQLITE_FULL)	1	1	

# Instalação (Linux)

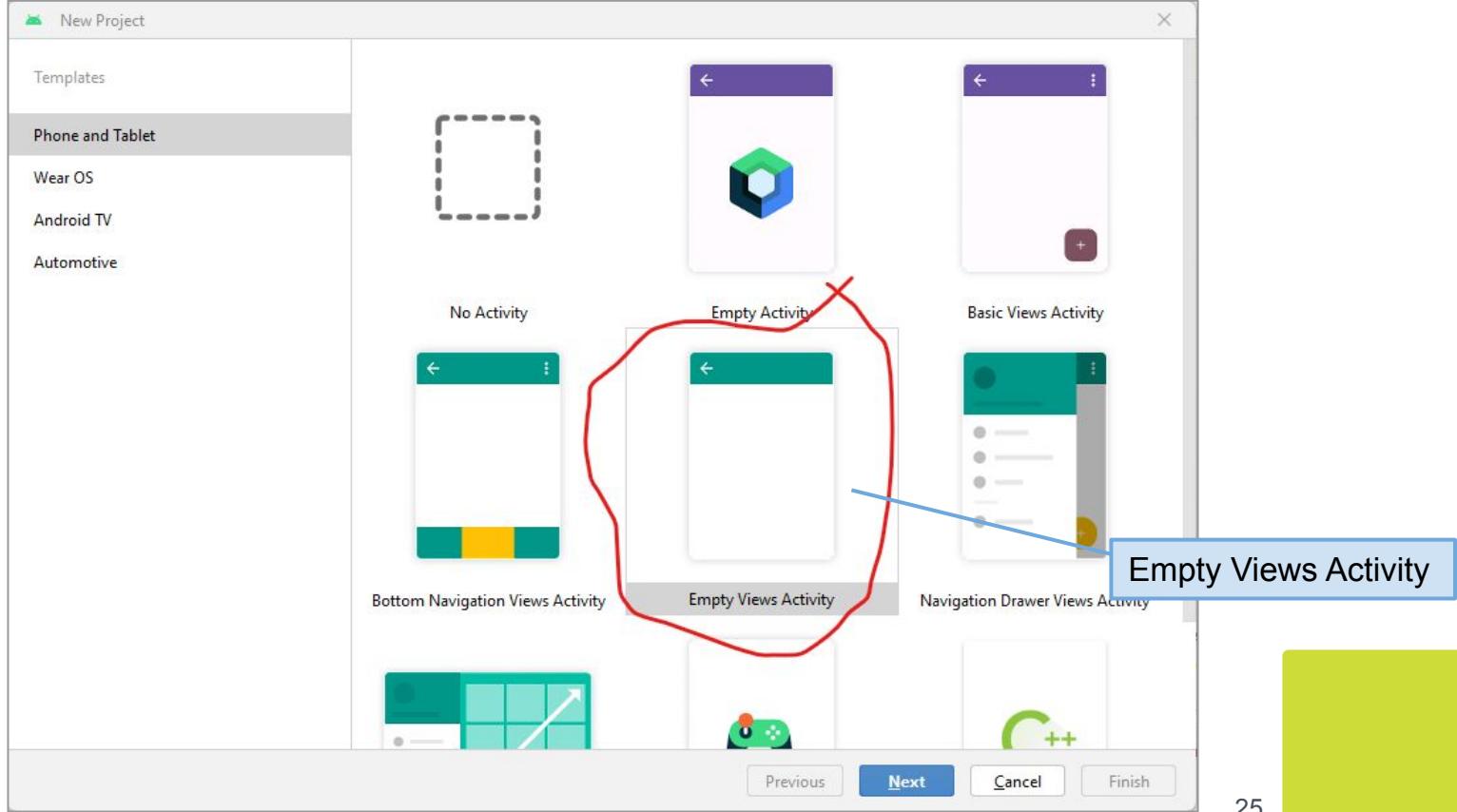
```
$ tar -xvf Downloads/android-studio-202X.X.X.XX-linux.tar.gz  
$ cd android-studio/bin/  
$ ./studio.sh  
→ Na primeira execução, serão baixados e instalados componentes  
→ No total, serão baixados aproximadamente 1.6GB
```



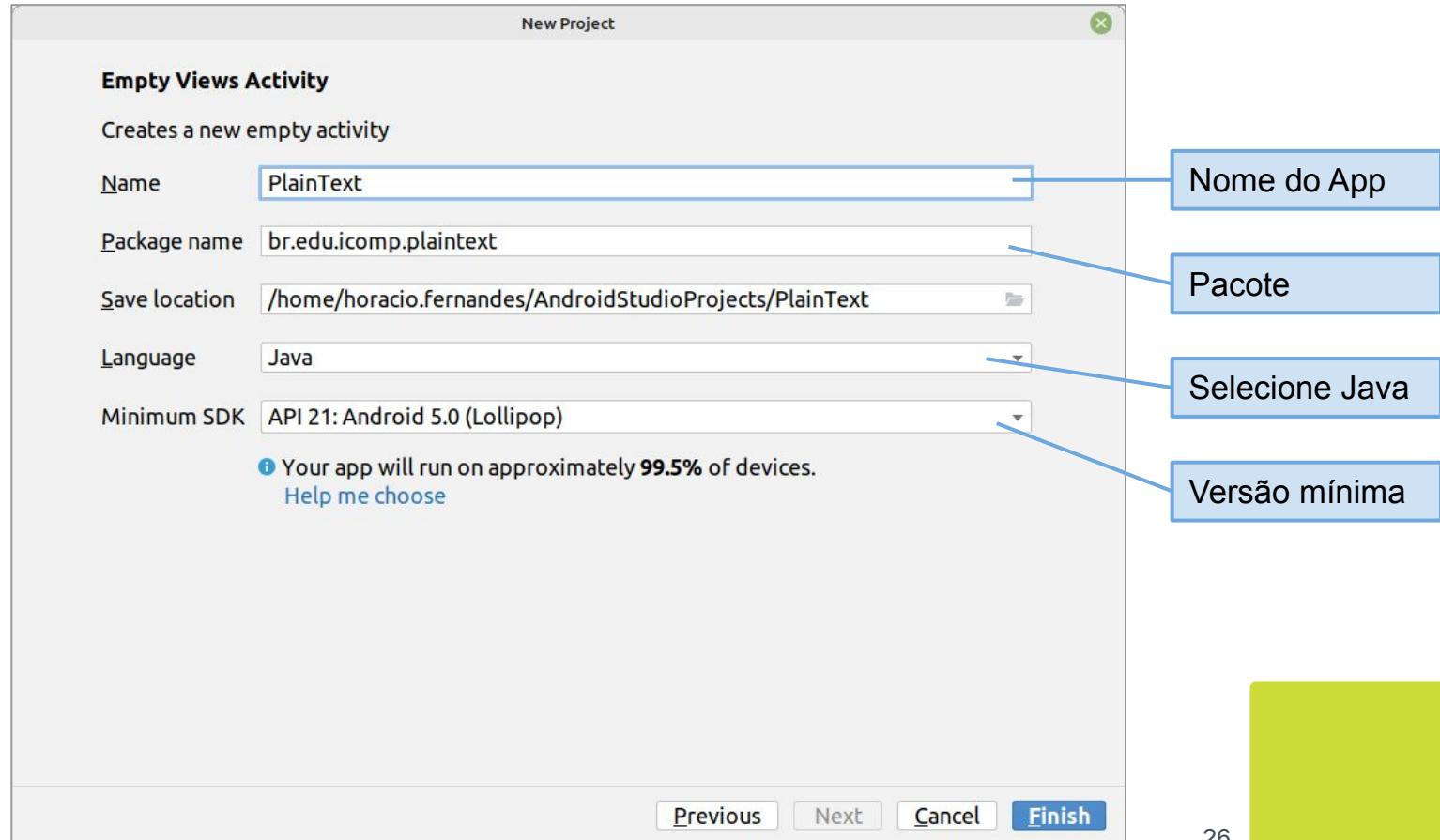
# Tela Inicial, Novo App



# Novo App



# Novo App



Android

- app
- manifests
- java
  - br.edu.icomp.plaintext
  - MainActivity
  - br.edu.icomp.plaintext (andr)
  - br.edu.icomp.plaintext (test)
- res
- Gradle Scripts

activity\_main.xml MainActivity.java

```
1 package br.edu.icomp.plaintext;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12 }
```

# Android Studio

Arquivos

Editor

Logcat, usado para mensagens de debug

Logcat

No connected devices

No debuggable processes

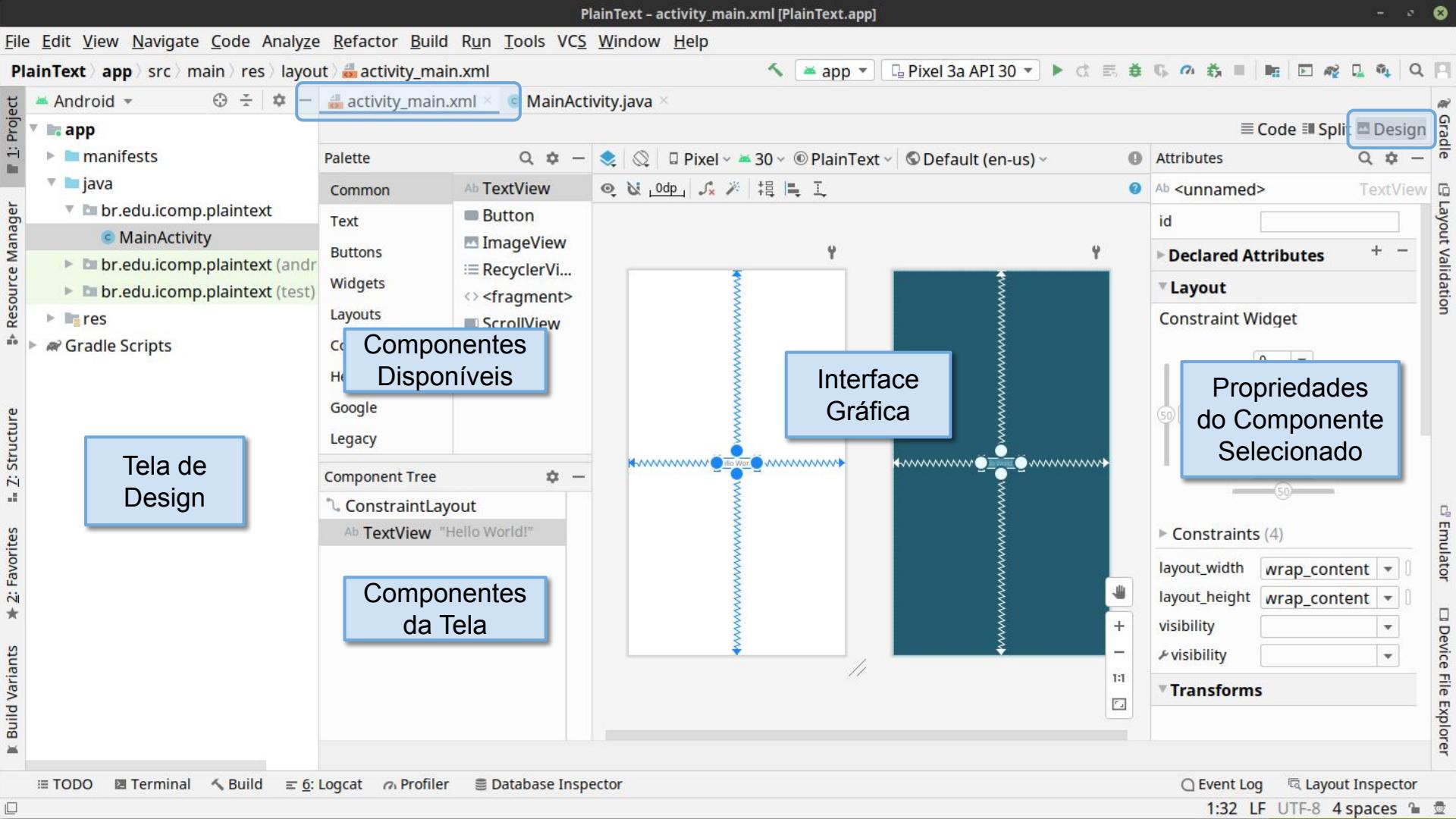
Verbose

Regex

Show only selected application

loqcat





Code Split Design

1: Project

- app
  - manifests
  - java
    - br.edu.icomp.plaintext
    - MainActivity
  - br.edu.icomp.plaintext (andr)
  - br.edu.icomp.plaintext (test)
  - res
- Gradle Scripts

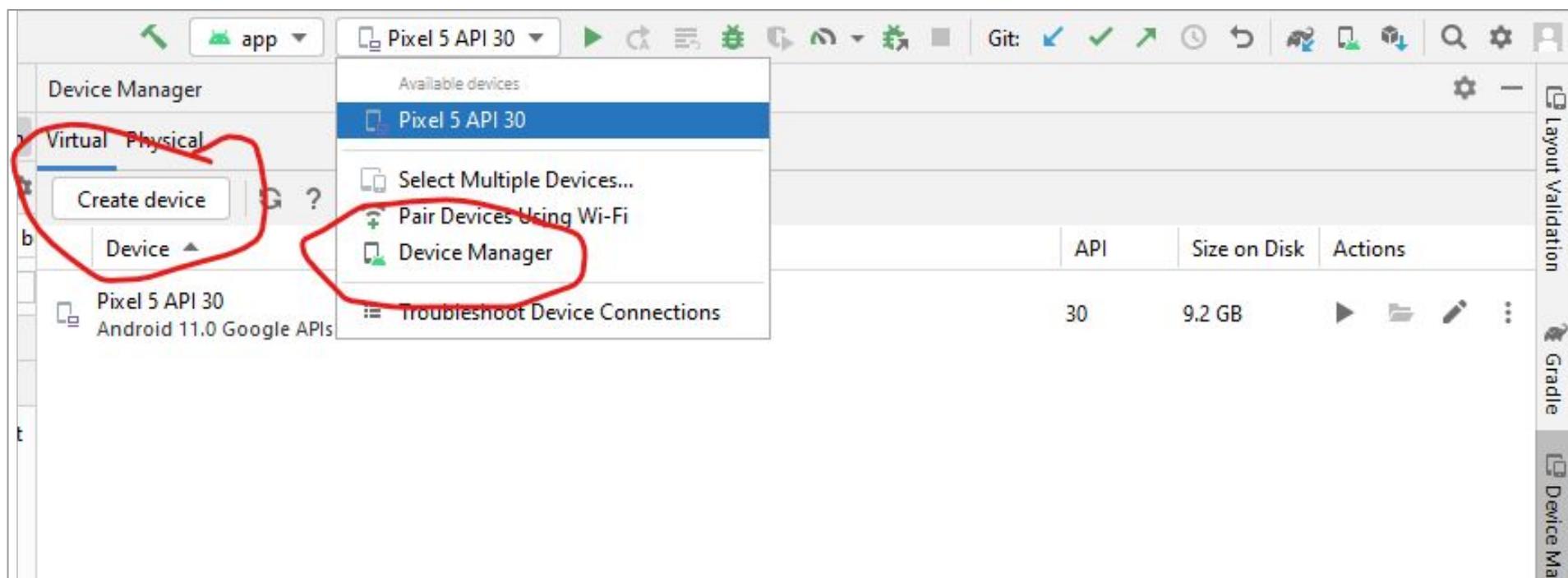
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hello World!"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent" />
17
18  </androidx.constraintlayout.widget.ConstraintLayout>
```

Editor XML  
da Interface  
Gráfica

# AVD - Android Virtual Device

- Emulador que permite executar os aplicativos em sua máquina
  - Ele emula um celular e executa uma versão completa do Android
  - O mesmo do seu celular, mas sem os aplicativos da operadora e do fabricante
  - Permite testar o aplicativo em celulares de configurações e Android diferentes
- Para abrir o gerenciador de AVDs:
  - Tools → Device Manager
    - *Talvez seja necessário criar um AVD novo (próximos slides)*

# AVD - Android Virtual Device



# AVD - Android Virtual Device

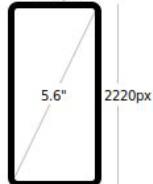
Virtual Device Configuration

Select Hardware

Choose a device definition

Category	Name	Play St...	Size	Resolu...	Density
TV	Pixel XL		5.5"	1440...	560d...
Phone	Pixel 4 XL		6.3"	1440...	560d...
Wear OS	Pixel 4	►	5.7"	1080...	440d...
Tablet	Pixel 3a XL		6.0"	1080...	400d...
Automo...	Pixel 3a	►	5.6"	1080...	440d...
	Pixel 3 XL		6.3"	1440...	560d...

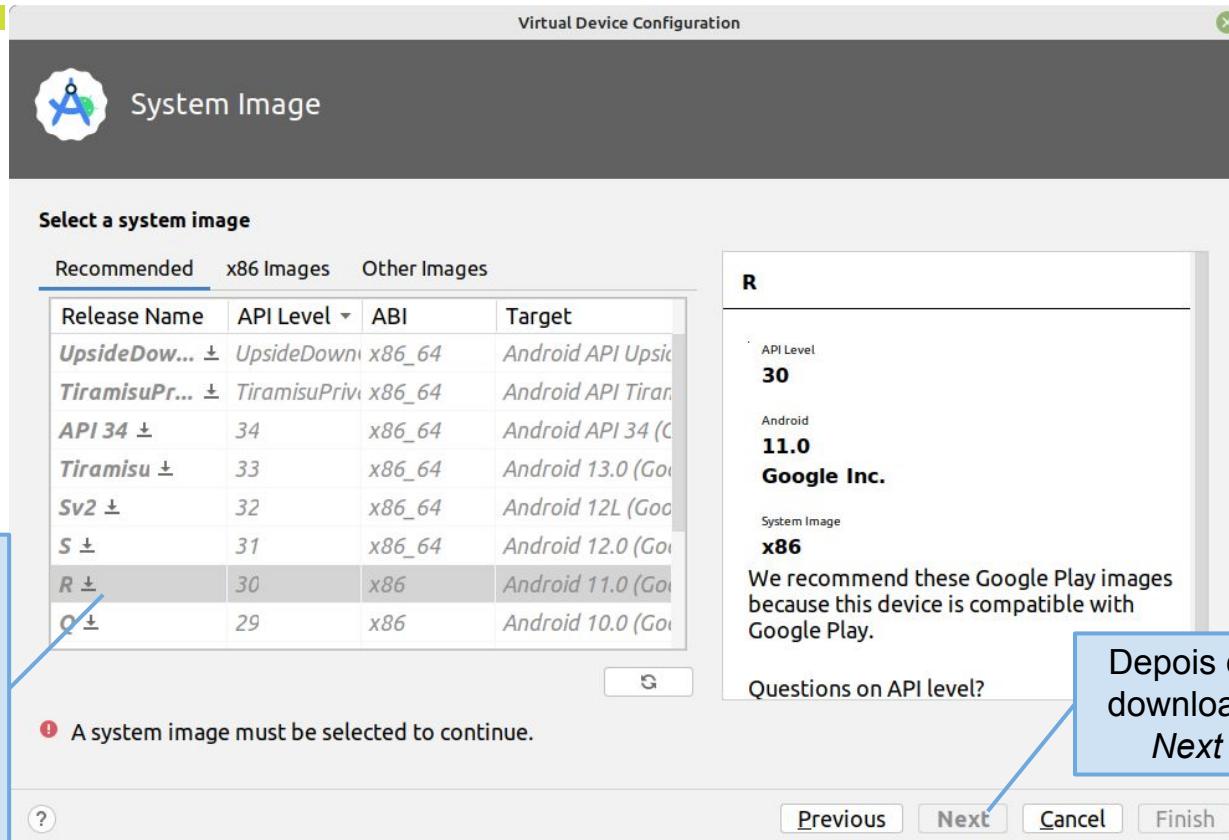
**Pixel 3a**



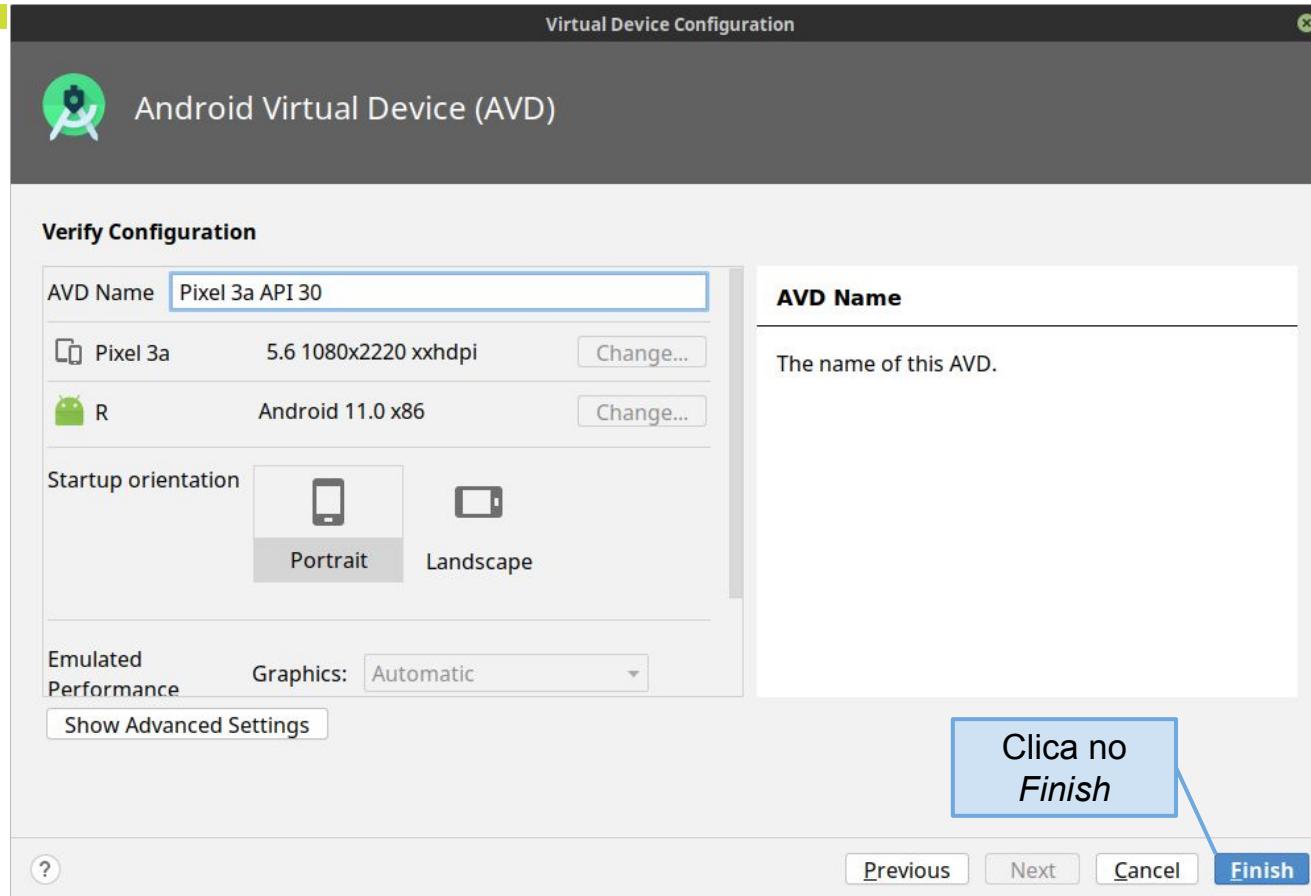
— 1080px —  
5.6" 2220px  
Size: large  
Ratio: long  
Density: 440dpi

New Hardware Profile Import Hardware Profiles Clone Device... ? Previous Next Cancel Finish

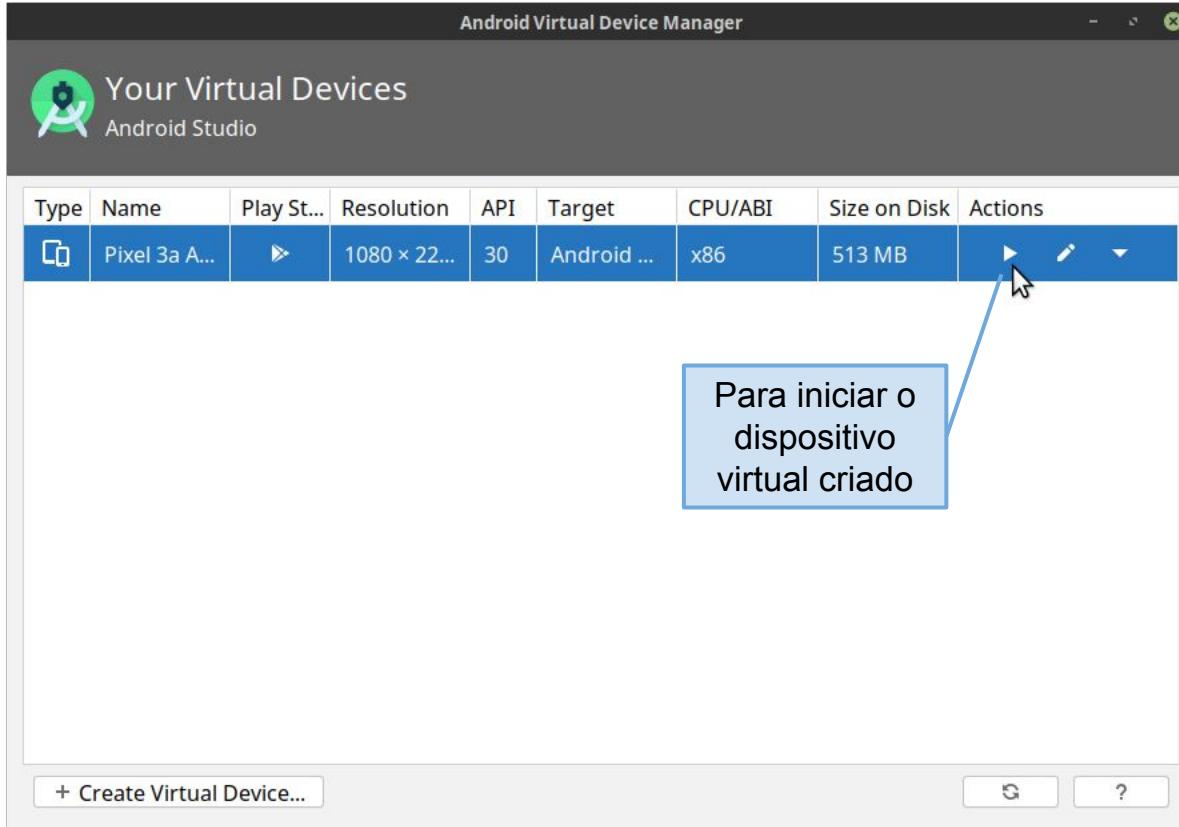
# AVD - Android Virtual Device



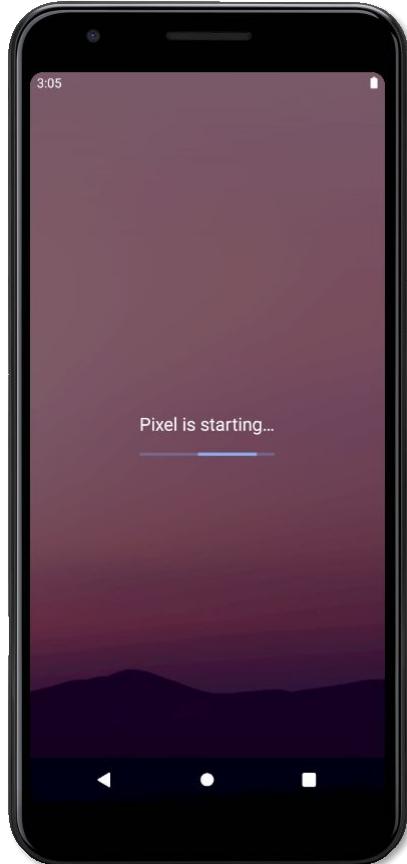
# AVD - Android Virtual Device



# AVD - Android Virtual Device

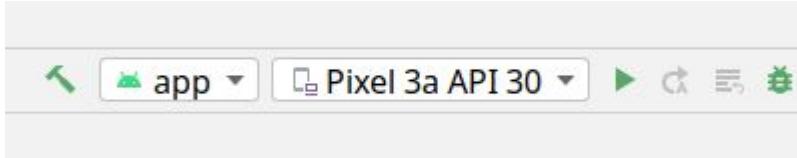


# AVD - Android Virtual Device



# Executando o App no AVD

- Clicando no botão “Play”



- No Menu

- Run → Run ‘App’
  - Shift + F10

- Teclado

- Botão ESC → Voltar
  - Botão HOME → Home
  - F2 → Menu



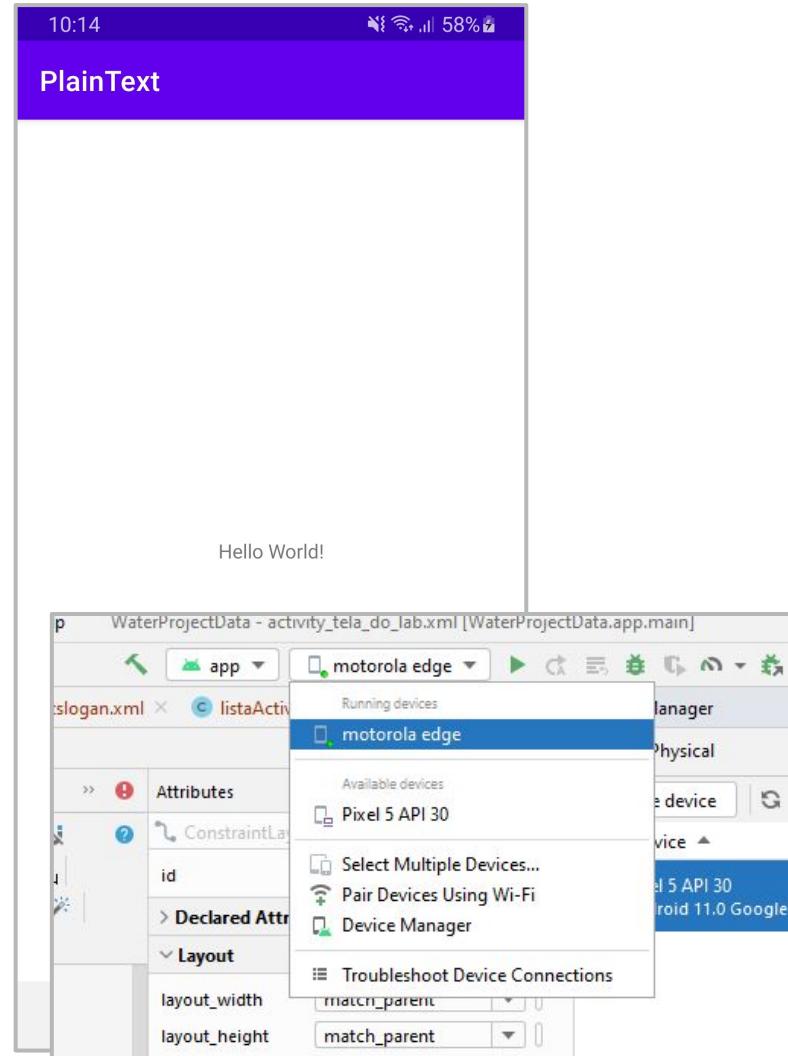
# Executando no Celular

---

- Dependendo da configuração da sua máquina de desenvolvimento, o AVD pode ficar muito lento ou mesmo nem executar
- Além disso, o AVD não tem suporte a todos os recursos
  - Bluetooth
  - Movimentos realistas (para testar sensores de movimento)
  - Câmera do celular (que, normalmente, é melhor que as webcams)
- Outras vezes, você quer ver seu App executando em um celular real

# Executando no Celular

- Passos para executar no celular
  - Conecta celular via USB
  - Habilita “Depuração de USB” no dispositivo
    - *Android 7 em diante: Opções → Sobre o dispositivo → Info. Software → Clique no N. de Compilação diversas vezes → Voltar → Voltar → Opções do Desenv. → Depuração de USB*
    - *Em outros celulares, este passo pode variar*
  - No Android Studio, Run → Run
    - *O Android Studio detecta seu celular conectado, instala e inicia o App no celular*



Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



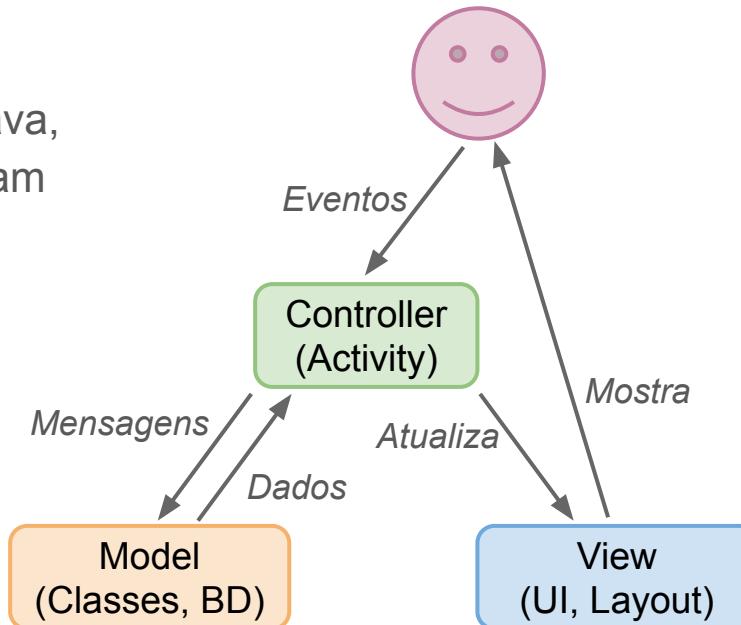
# Interface Gráfica

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

 Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Modelo MVC

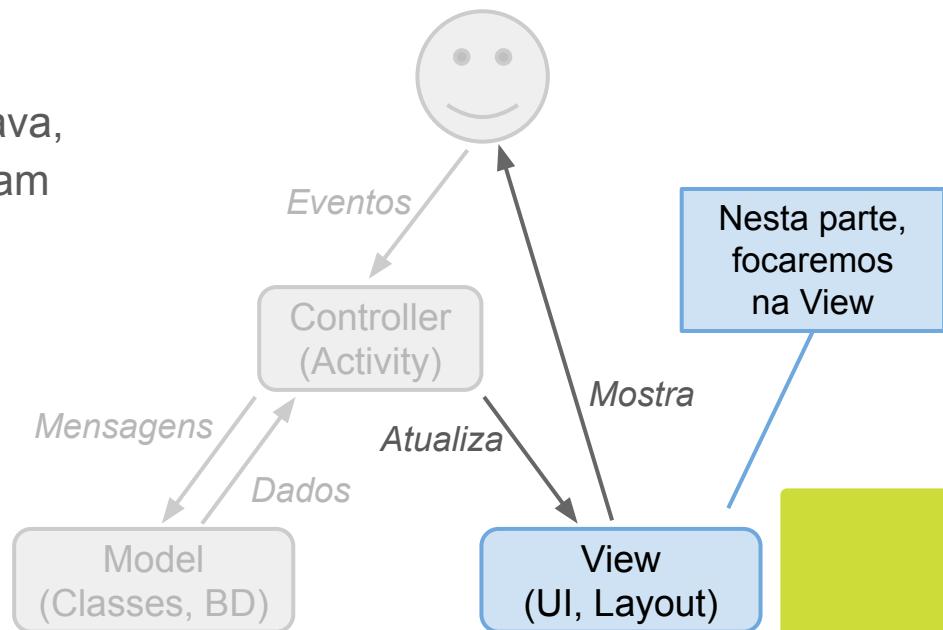
- O Android usa o modelo MVC – *Model, View, Controller*
  - *Model*: classes implementadas em Java
  - *View*: componentes da Interface normalmente feitos em XML
  - *Controller*: implementados em Java, instanciam os modelos a controlam as views. São conhecidas como “activities” no Android



# Modelo MVC

- O Android usa o modelo MVC – *Model, View, Controller*

- *Model*: classes implementadas em Java
- *View*: componentes da Interface normalmente feitos em XML
- *Controller*: implementados em Java, instanciam os modelos a controlam as views. São conhecidas como “activities” no Android

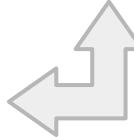
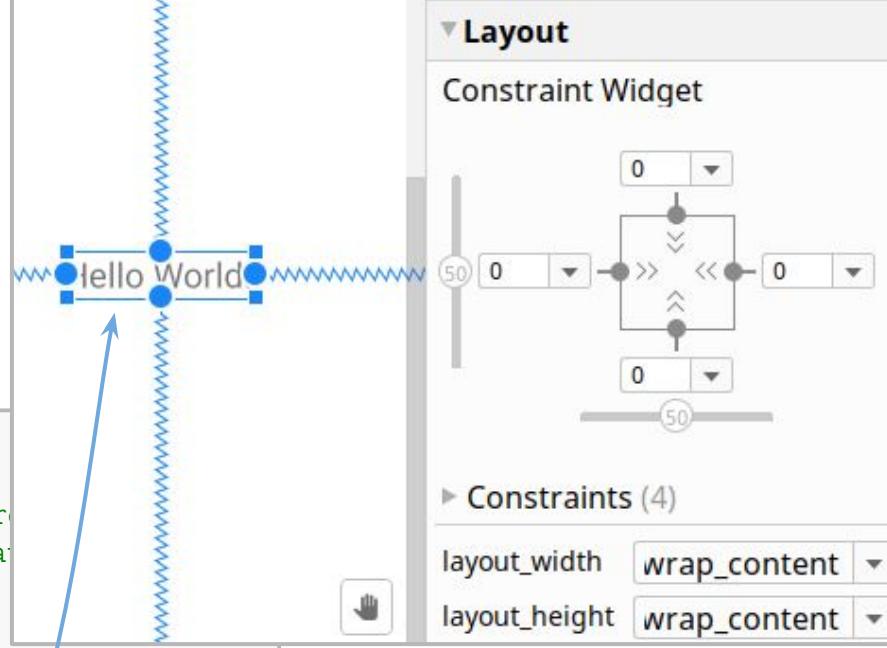
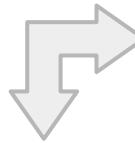


# Interface Gráfica

- A interface gráfica pode ser feita de três formas:
  - Usando o Editor do Android Studio (Design)
    - *O editor irá gerar um XML do Layout automaticamente*
  - Editando o XML (Text) manualmente
    - *O Android faz um parse do XML do Layout e gera internamente os objetos dos componentes (Views). Este processo é conhecido como “inflate” do XML.*
    - *Toda tag XML tem uma classe Java correspondente (subclasse da classe View)*
  - Gerando Códigos em Java manualmente
    - *Instanciando objetos de componentes gráficos (Views) manualmente e adicionando-os à interface (similar ao feito no Swing)*
    - *Não recomendado por quebrar o modelo MVC.*

# Design e Text

- Do design ao text
  - Na prática, alternamos entre os dois



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/r
    xmlns:app="http://schemas.android.com/apk/res-a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

# Layouts

- Layouts
  - Agrupam componentes UI (Views) relacionados
    - *Podem ter outros Layouts internamente (que agruparão outras Views)*
  - Definem como os componentes serão posicionados na tela
    - *Similar aos layouts do Swing*
  - Layouts são classes (e tags) que herdam a classe ViewGroup
- Todo componente (botões, textos, etc) deve estar dentro de um Layout
- Principais Layouts:
  - ConstraintLayout (recomendado)
  - RelativeLayout
  - LinearLayout

# Constraint Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textHello"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/textIntroducao"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="42dp"
        android:layout_marginTop="32dp"
        android:text="Introdução ao Android"
        app:layout_constraintLeft_toLeftOf="@+id/textHello"
        app:layout_constraintTop_toBottomOf="@+id/textHello" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



Attributes

Ab textIntroducao TextView

id textIntroducao

Declared Attributes + -

Layout

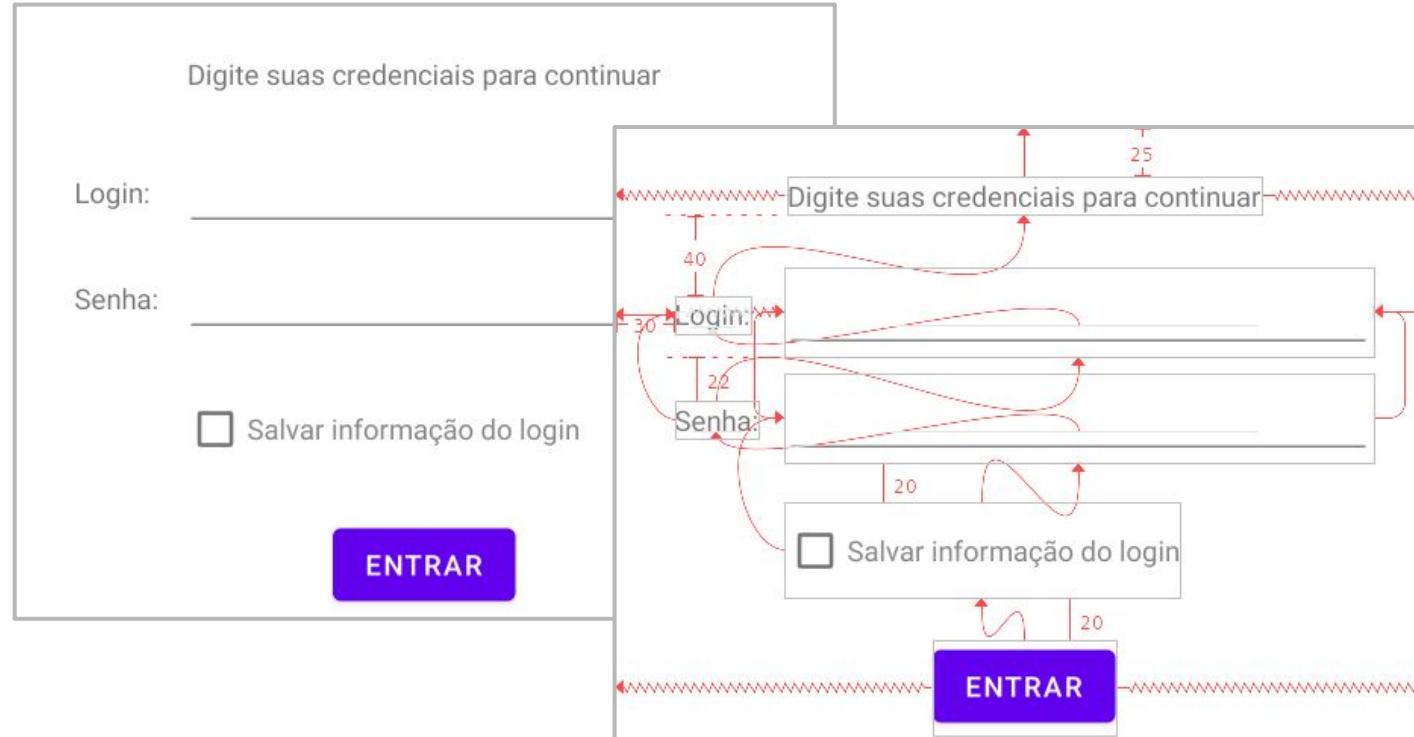
Constraint Widget

Constraints

- Left → LeftOf textHello (0dp)
- Top → BottomOf textHello (3...

# ConstraintLayout

- Posiciona elementos com base em outros, de forma relativa



# Recursos (*Resources*)

---

- ▣ Recursos são arquivos externos ao seu código-fonte (java)
- ▣ São “externalizados” do seu código para serem mantidos independentemente e mais facilmente
- ▣ Ficam na pasta “res”
- ▣ Exemplos:
  - Imagens
  - Layouts
  - Menus
  - Outros valores
    - *Strings, dimensões, estilos, etc*

# Recursos (Resources)

- Exemplo de recursos de Strings (res/values/strings.xml)

```
<resources>
    <string name="app_name">PlainText</string>
    <string name="introducao">Introdução ao Android</string>
</resources>
```

- São acessados pelo nome usando o “@”:

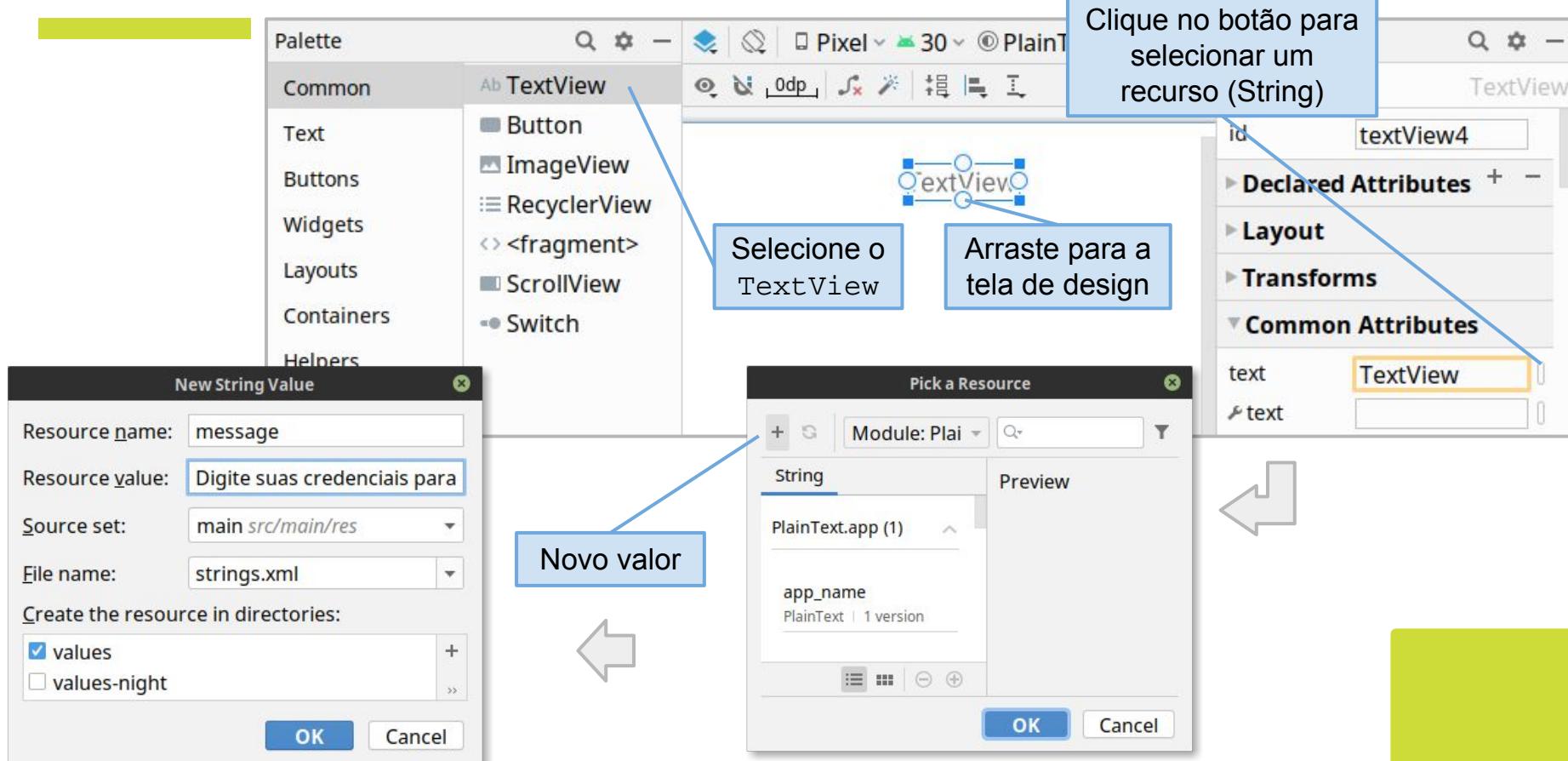
```
<TextView
    android:id="@+id/textIntroducao"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="42dp"
    android:layout_marginTop="32dp"
    android:text="@string/introducao"
    app:layout_constraintStart_toStartOf="@+id/textHello"
    app:layout_constraintTop_toBottomOf="@+id/textHello" />
```

# Componentes

- São as partes da tela que o usuário vê e interage
  - Assim como no Swing, são objetos de uma determinada classe
  - Possuem atributos e métodos
- Os componentes são também chamados de *views*, pois todas as classes que as implementam herdam a classe `View`
- Exemplos de componentes (*views*):
  - `TextView`
  - `EditText`
  - `CheckBox`
  - `Button`
  - `ImageView`

# TextView

■ Representa um texto (label)



# TextView

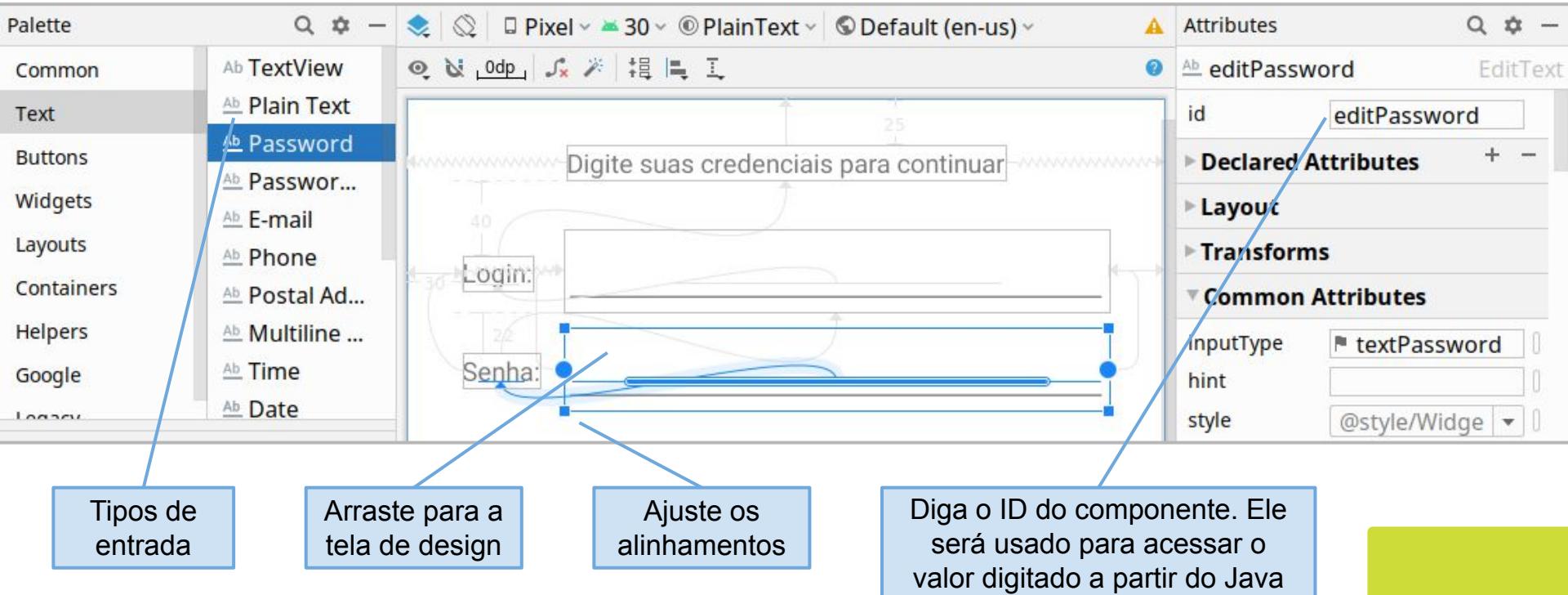
- Representa um texto (label)

- XML resultante do slide anterior (campo de senha apenas)

```
<TextView  
    android:id="@+id/textMessage"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="25dp"  
    android:text="@string/message"  
    android:textSize="14dp"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

# EditText

- Texto de entrada do usuário
- Vários tipos: text, textPassword, number, etc



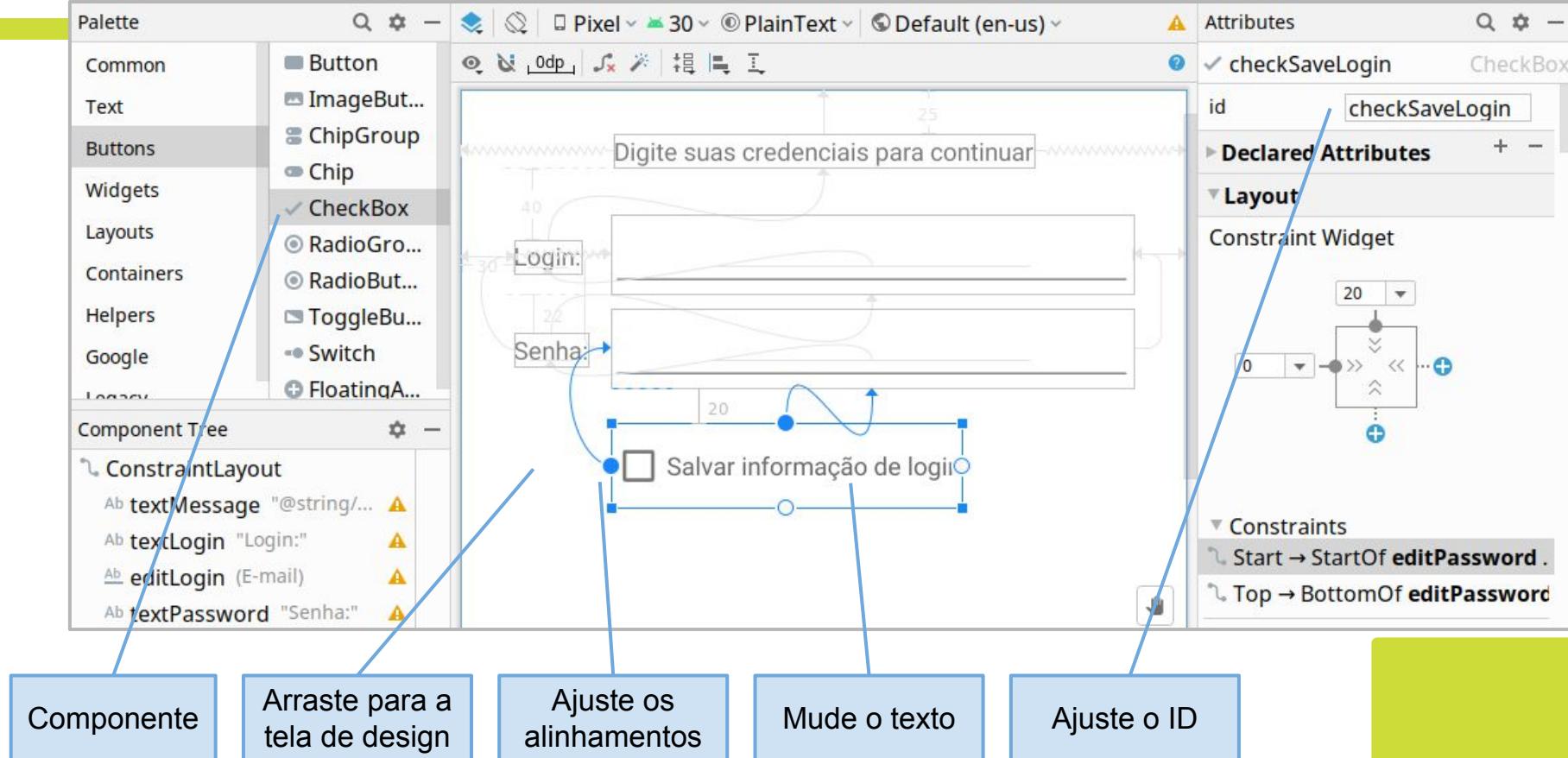
# EditText

- Texto de entrada do usuário
- Vários tipos: text, textPassword, number, etc
- XML resultante do slide anterior (campo de senha apenas)

```
<EditText  
    android:id="@+id/editPassword"  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:inputType="textPassword"  
    app:layout_constraintStart_toStartOf="@id/editLogin"  
    app:layout_constraintEnd_toEndOf="@id/editLogin"  
    app:layout_constraintBaseline_toBaselineOf="@id/textPassword" />
```

# CheckBox

## Caixa de seleção



# CheckBox

- Caixa de seleção

- XML resultante do slide anterior

```
<CheckBox  
    android:id="@+id/checkSaveLogin"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="20dp"  
    android:text="@string/saveLogin"  
    android:textColor="#777777"  
    app:layout_constraintStart_toStartOf="@+id/editPassword"  
    app:layout_constraintTop_toBottomOf="@+id/editPassword" />
```

# Button

## ■ Botão clicável

The screenshot shows the Android Studio Layout Editor with a login interface. A purple button labeled "ENTRAR" is selected. The "Buttons" category is highlighted in the Palette on the left. The Attributes panel on the right shows the button's ID as "buttonEnter". The Layout tab displays constraints: top=20, bottom=0, left=0, right=0, width=50dp, height=20dp.

**Componente**

**Arraste para a tela de design**

**Ajuste os alinhamentos**

**Mude o texto**

**Ajuste o ID**

# Button

- Botão clicável

- XML resultante do slide anterior

```
<Button  
    android:id="@+id/buttonEnter"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="20dp"  
    android:text="@string/enter"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/checkSaveLogin" />
```

# Mudando as Cores

---

- No arquivo “res/values/themes/themes.xml” é possível mudar as cores básicas do tema do seu aplicativo
- Na verdade, é possível personalizar basicamente todos os componentes do aplicativo
  - Entretanto, os detalhes desse arquivo vão além do escopo do curso

# Mudando as Cores

```
<resources xmlns:tools="http://schemas.android.com/tools">

    <style name="Theme.PlainText" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">#565758</item>
        <item name="colorPrimaryVariant">#565758</item>
        <item name="colorOnPrimary">#a1c639</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">#565758</item>
        <item name="colorSecondaryVariant">#a1c639</item>
        <item name="colorOnSecondary">#a1c639</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">#565758</item>
        <!-- Customize your theme here. -->
        <item name="android:textColor">#39393a</item>
    </style>
</resources>
```

Fundo da barra do app

Texto dos botões

Fundo da barra do relógio

Textos das telas

# Mudando as Cores

- Ao modificar o XML, basta voltar para a tela de design

Digite suas credenciais para continuar

Login: \_\_\_\_\_

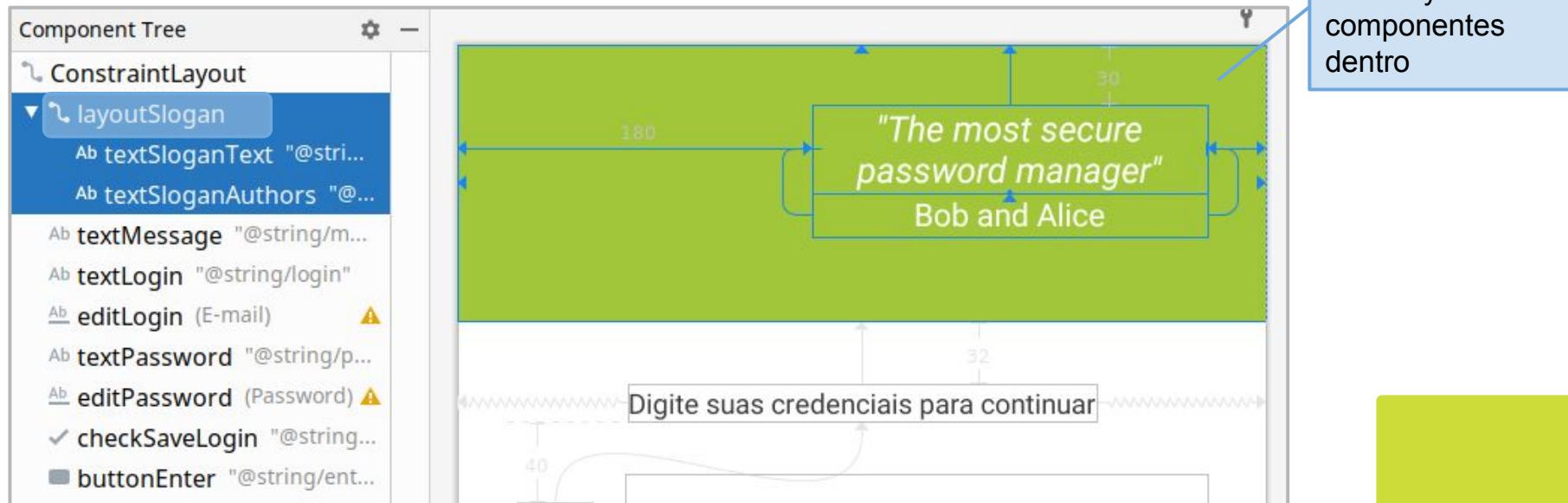
Senha: \_\_\_\_\_

Salvar informação de login

**ENTRAR**

# Layout dentro de Layout

- Um layout pode ser um sub-componente de um layout
- Combine layouts para obter os efeitos desejados

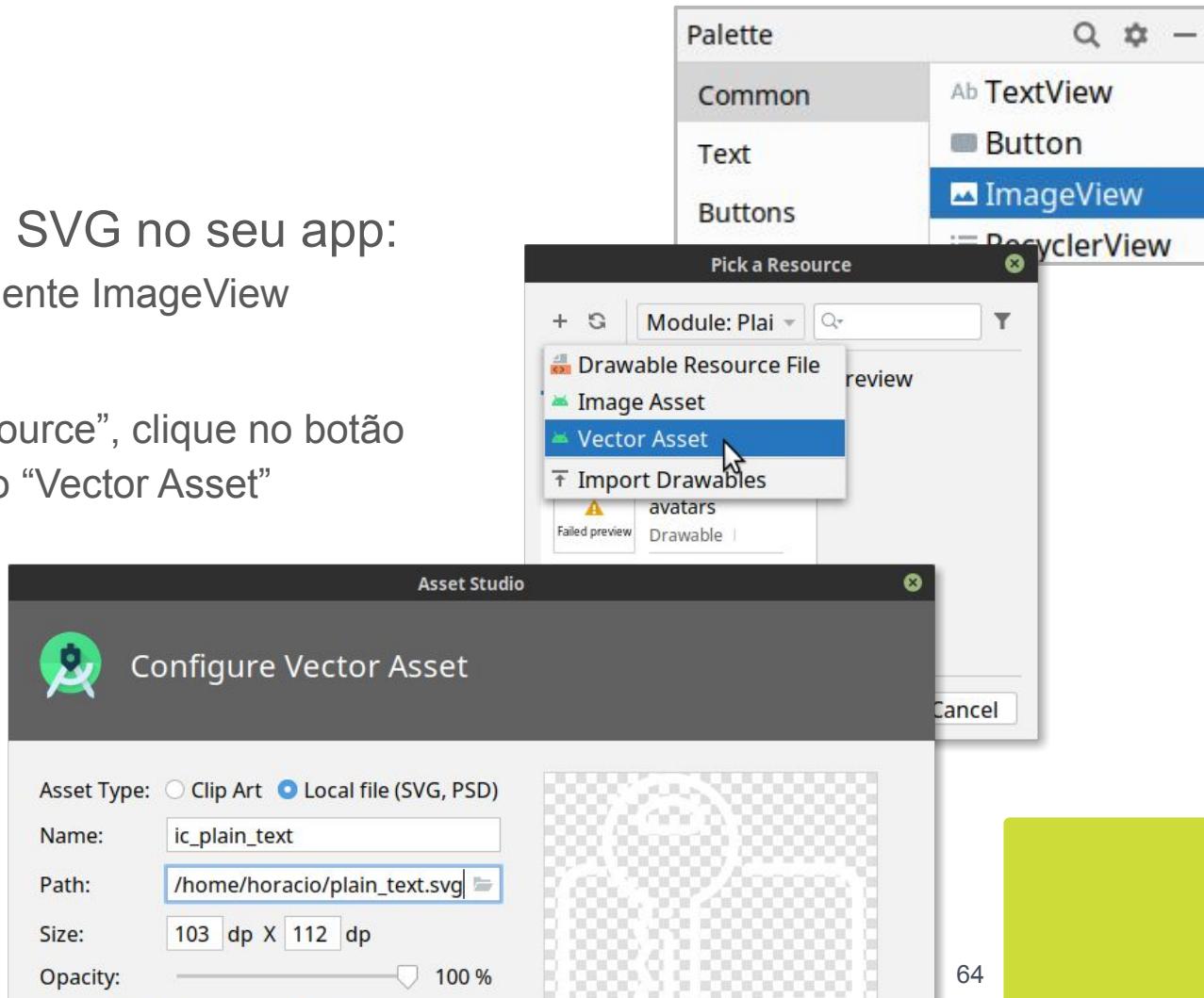


# ImageView

- O componente ImageView permite inserir imagens na interface
- Tipos de Imagens
  - Não-Vetoriais
    - *Imagen é composta por um conjunto de “pixels”*
    - *O tamanho/qualidade da imagem depende da densidade de pixels da tela*
    - *PNG, GIF, JPG*
  - Vetoriais
    - *Imagen é composta por um conjunto de linhas, quadrados, paths, etc*
    - *Qualidade é mantida independente do tamanho, zoom, densidade de pixes, etc*
    - *SVG, EPS*
- Como os apps executam em diferentes telas e dispositivos, recomenda-se o uso de imagens vetoriais
  - O Android tem suporte a imagens SVG

# ImageView

- Para adicionar um SVG no seu app:
  - Arraste o componente ImageView para a interface
  - Na janela “Pick a Resource”, clique no botão “+” e, depois na opção “Vector Asset”
- Selecione “Local File”
- Indique o Path
- Por fim, clique em “Next” → “Finish”, e selecione a nova imagem na janela “Pick a Resource”



# ImageView

Ajuste os alinhamentos

Ajuste o ID

The screenshot shows the Android Studio Layout Editor with a login screen. On the left, the Palette lists Common, Text, Buttons, Widgets, Layouts, and Containers. The Component Tree shows a ConstraintLayout containing a ImageView named 'imageLogo' and a TextView named 'textSloganText'. The main area displays a green card with a white key icon and the text "The most secure password manager" by Bob and Alice. Below the card, there's a message "Digite suas credenciais para continuar". The login form includes fields for 'Login:' and 'Senha', and a checkbox for 'Salvar informação de login'. The Attributes panel on the right shows the ImageView's id is set to 'imageLogo'. A blue callout box labeled 'Ajuste os alinhamentos' points to the constraint lines on the key icon. Another blue callout box labeled 'Ajuste o ID' points to the 'id' field in the Attributes panel.

Palette

Common Text Buttons Widgets Layouts Containers

Component Tree

ConstraintLayout layoutSlogan

imageLogo

Ab textView

Ab Button

Ab ImageView

Ab Recycler...

Ab <fragme...

Ab ScrollView

Ab Switch

Pixel 30 PlainText Default (en-us)

Attributes

imageLogo ImageView

id imageLogo

Declared Attributes

Layout

Constraint Widget

0 30 0

Constraints

Start → StartOf parent (30dp)

Top → TopOf parent (0dp)

Bottom → BottomOf parent (...)

layout\_width 151dp

layout\_height 110dp

visibility

Digite suas credenciais para continuar

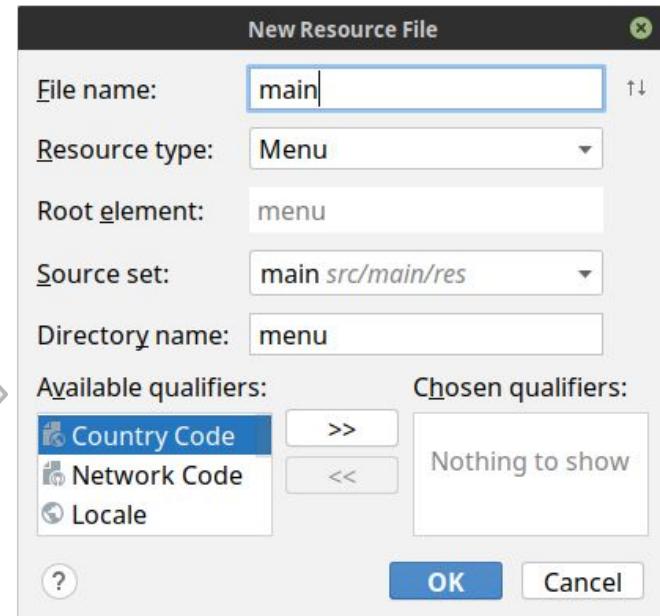
Login: \_\_\_\_\_

Senha \_\_\_\_\_

Salvar informação de login

# Criando um Menu

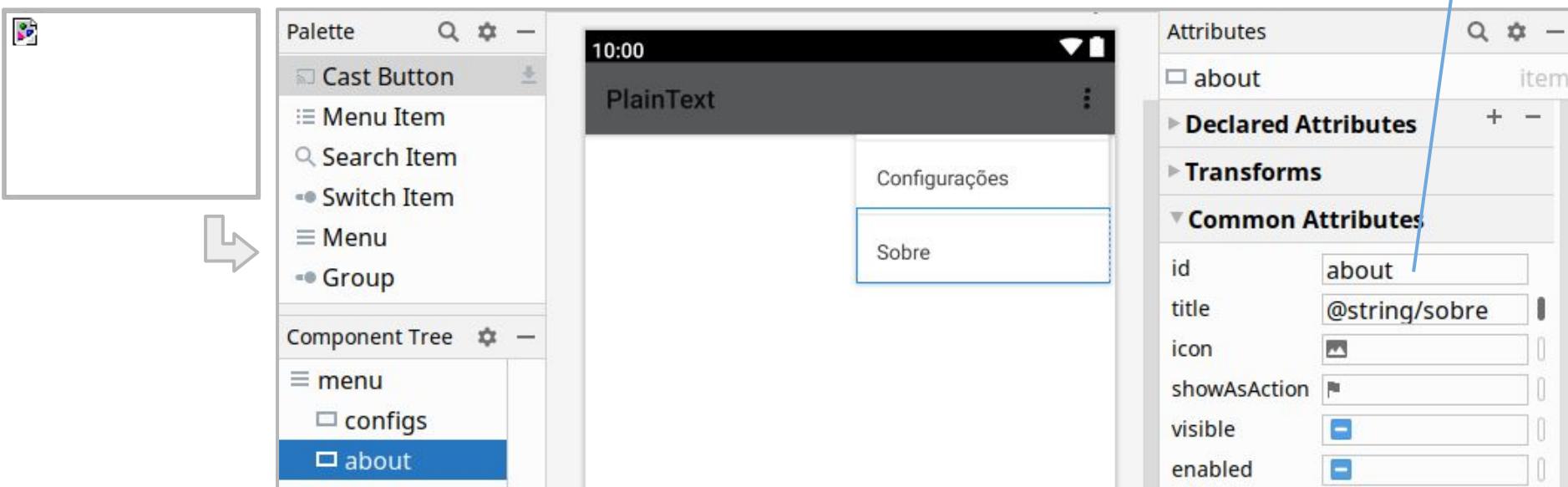
- Toda tela pode ter um menu
- Um menu é definido a partir de um arquivo XML
- Para criar um novo menu:
  - Botão direito no diretório “res”
  - New
  - Android Resource File



# Criando um Menu

- Será criado um arquivo XML - res/menu/main.xml
  - Este arquivo deve ser editado para adicionar os itens do menu
  - Você pode editar o arquivo na janela de design, arrastando os componentes do menu, ou editar o XML direto.

Lembre-se de  
ajustar os IDs



# Criando um Menu

- XML resultante do slide anterior

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:id="@+id/configs" android:title="@string/configs" />
    <item android:id="@+id/about" android:title="@string/sobre" />
</menu>
```

# Criando um Menu

- Por fim, modifique classe MainActivity, para usar o menu
  - Após o método onCreate, inclua o método abaixo:

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}
```

- Se você copiar e colar o código acima, provavelmente dará erro na classe Menu, pois o pacote dele não foi importado
- O Android Studio pode corrigir esse erro automaticamente, para isso, basta clicar no erro (texto vermelho sublinhado) e digitar Alt+Enter



# Interface Gráfica

## Resultado Final

PlainText



"The most secure password manager"  
Bob and Alice

Digite suas credenciais para continuar

Login: \_\_\_\_\_

Senha: \_\_\_\_\_

Salvar informação de login

**ENTRAR**

PlainText



Configurações  
Sobre  
"secure password manager"  
Bob and Alice

Digite suas credenciais para continuar

Login: \_\_\_\_\_

Senha: \_\_\_\_\_

Salvar informação de login

**ENTRAR**

PlainText



"The most secure password manager"  
Bob and Alice

Digite suas credenciais para continuar

Login: **dovahkiin**

Senha: **\*\*\*\*\***

Salvar informação de login

**ENTRAR**

# Interface Gráfica

- Caso não apareça o menu, verifique se o parent “noActionBar” está setado em res->values->themes->themes.xml
- Provavelmente estará algo como “\*.NoActionBar”. Basta remover.
- Se fizer deploy no celular, verifique se não está usando modo noturno

A screenshot of the Android Studio code editor showing the file "night\themes.xml". The code defines a style for the night theme:

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Teste2" parent="Theme.Material3.DayNight">
        <!-- Customize your dark theme here. -->
        <!-- <item name="colorPrimary">@color/my_dark_primary</item> -->
    </style>
```

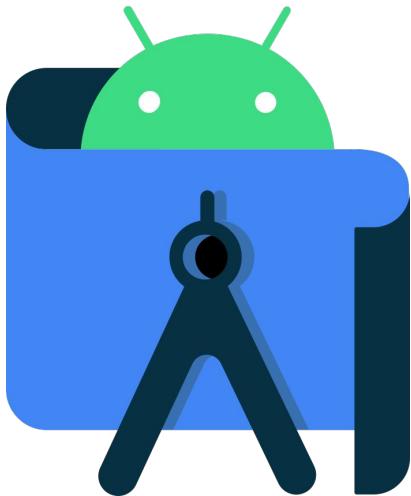
A red oval highlights the "parent" attribute in the style definition.

# Interface Gráfica

- Resultado Final
- MVC
  - Note como geramos a interface inteira sem (quase) necessidade de escrever códigos Java
- Vamos agora
  - Programar ações
    - *Controllers*
  - Acessar BD
    - *Models*
  - Gerar outras telas
  - Outros



Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



# Activities e Intents

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Activity

---

- Activity é uma parte da aplicação que permite:
  - acesso a uma instância da tela (interface) em que os componentes UI podem ser acessados e controlados
  - interação do usuário com o aplicativo e resposta a eventos
  - instanciar classes (criar objetos) dos modelos implementados

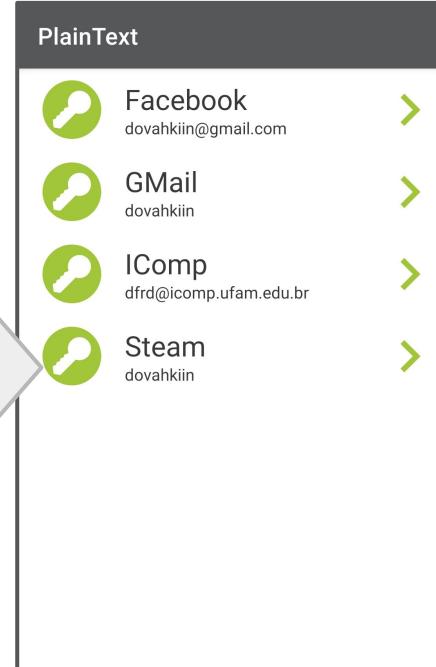
# Activity

- Normalmente, uma aplicação em Android é composta por diversas Activities
  - Cada nova tela da aplicação é uma activity
  - Uma das activities será a “main”, aberta quando a aplicação inicia
  - Uma activity chama a outra
    - *Ao fazer isso, a activity anterior para de executar, mas seu estado é guardado e, ao pressionar o botão de “voltar”, ela volta ao estado anterior*

Activity 1

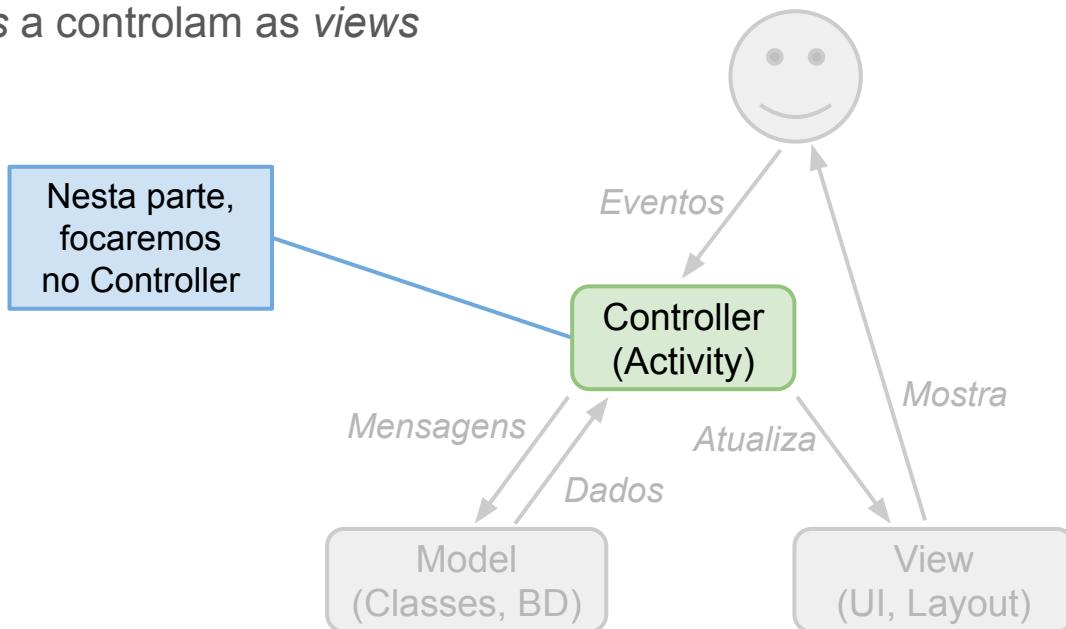


Activity 2



# Activity

- No modelo MVC, activity provê a parte do controller
  - Implementados em Java, instanciam os *modelos* e controlam as *views*



# Activity

- Toda activity extende a classe Activity (ou AppCompatActivity)
- Principais métodos:

- onCreate
- onStart
- onPause
- onResume
- onStop

Mostra a tela  
activity\_main, que  
criamos na parte anterior

Cria e mostra o menu,  
criado na parte anterior

```
package br.edu.ufam.icomp.plaintext;
// imports ...

public class MainActivity extends AppCompatActivity {

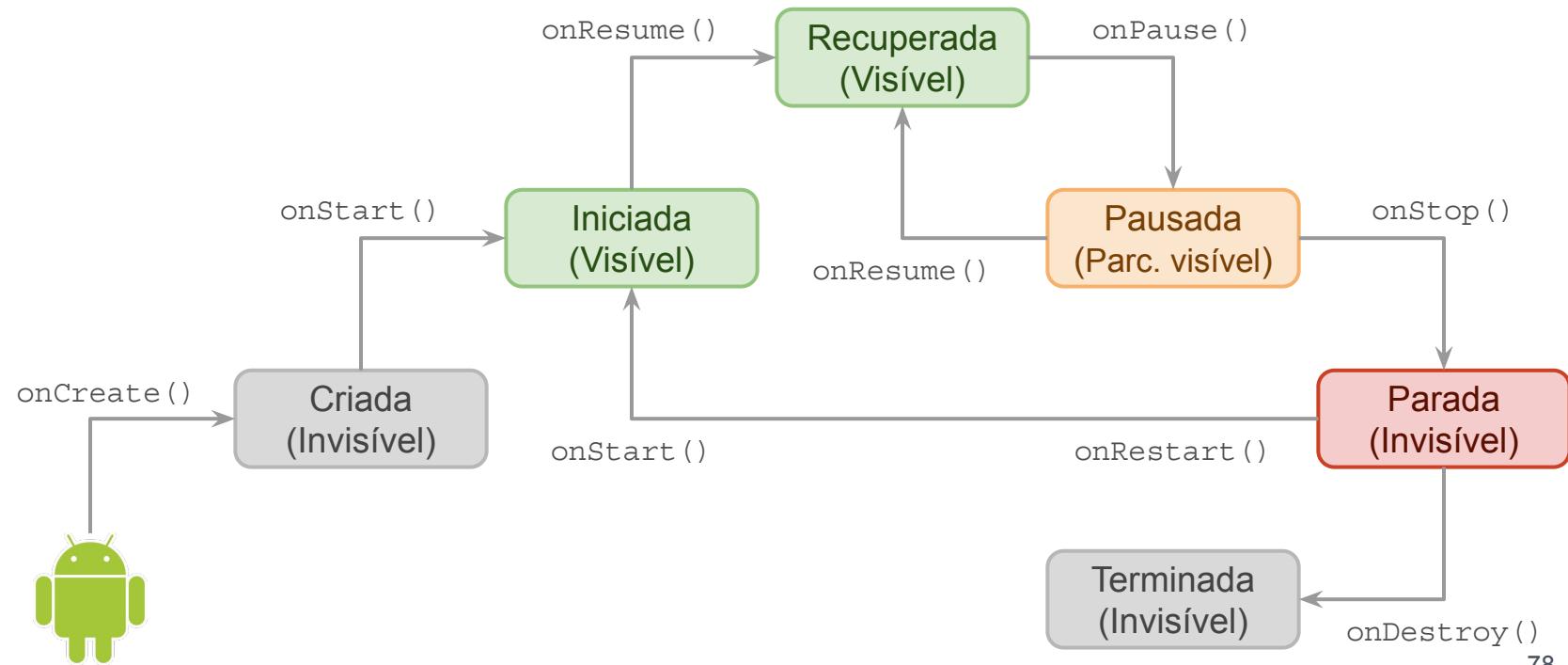
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

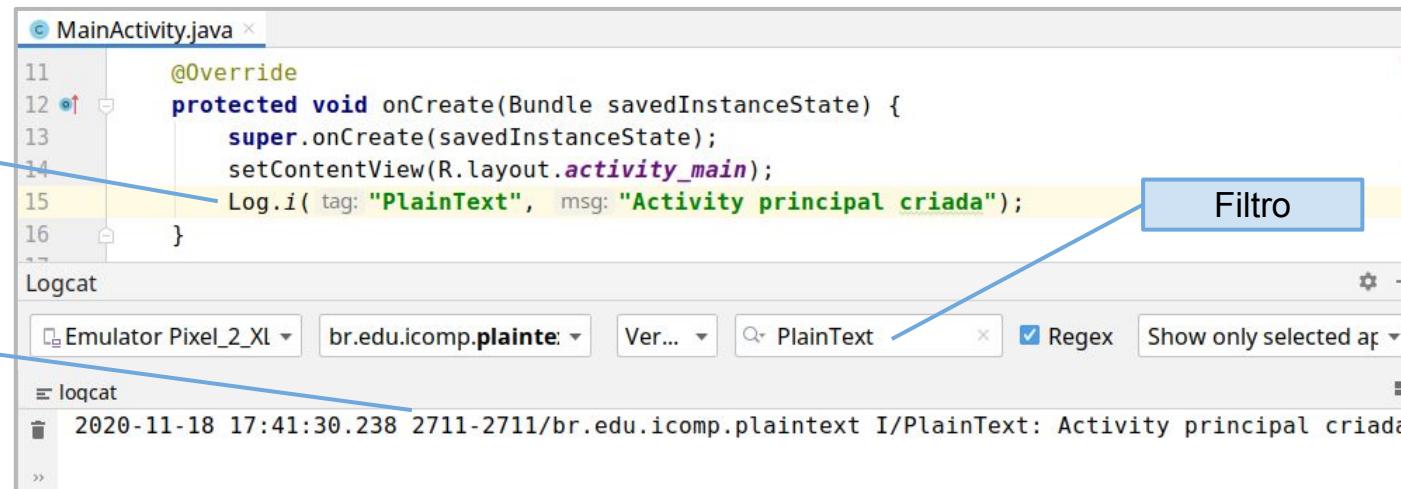
# Ciclo de Vida

- Toda activity segue um ciclo de vida



# Mensagens de Debug

- LogCat permite receber mensagens de debug da sua aplicação
  - Para escrever: `Log.i(String tag, String msg);`
  - Para ler: *View → Tool Windows → Android Monitor*
    - *Para facilitar a leitura, crie um filtro para mostrar apenas as suas Tags*
    - *Os erros do Java (incluindo as Exceções), são também mostradas no LogCat, mas é recomendável também criar um filtro para os erros*



# Debug e Ciclo de Vida

- Mensagens de debug nas fases do ciclo de vida

The screenshot shows the Android Studio interface with the code editor and Logcat window.

**MainActivity.java:**

```
24
25     @Override
26     protected void onStart() {
27         super.onStart();
28         Log.i( tag: "PlainText", msg: "Método onStart executado");
29
30
31     @Override
32     protected void onResume() {
33         super.onResume();
34         Log.i( tag: "PlainText", msg: "Método onResume executado");
```

**Annotations:**

- App Iniciada (onCreate)**: Points to the first two lines of the code, indicating the start of the application.
- Botão “Home” pressionado**: Points to the `onPause` method in the Logcat output, indicating the application has been paused.
- Retornando à App**: Points to the `onResume` method in the Logcat output, indicating the application has resumed.

**Logcat Output:**

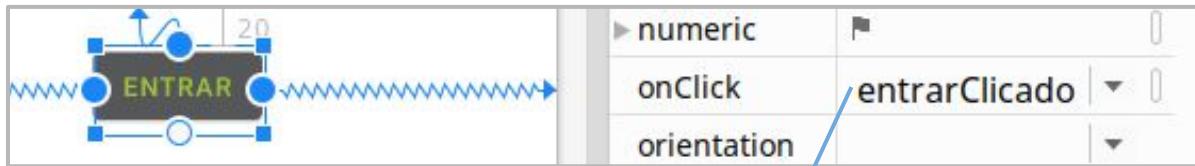
Timestamp	Process	Category	Message
2020-11-18 18:24:13.295	3449-3449/br.edu.icomp.plaintext	I/PlainText	Activity principal criada
2020-11-18 18:24:13.308	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onStart executado
2020-11-18 18:24:13.312	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onResume executado
2020-11-18 18:25:01.589	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onPause executado
2020-11-18 18:25:04.981	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onStop executado
2020-11-18 18:25:24.011	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onRestart executado
2020-11-18 18:25:24.345	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onStart executado
2020-11-18 18:25:24.771	3449-3449/br.edu.icomp.plaintext	I/PlainText	Método onResume executado

# Manipulando Eventos

- Cada um dos componentes UI possui uma série de eventos
  - Botões podem ser clicados, campos de texto podem ser modificados, etc
- Alguns eventos são especificados no próprio XML do componente e executam um determinado método implementado na Activity
  - Interface do método: `public void nomeDoMetodo (View view)`
    - *View view contém o objeto do componente UI que gerou o evento*

# Manipulando Eventos

- Exemplo: mostra “Olá” ao clicar no “Entrar”

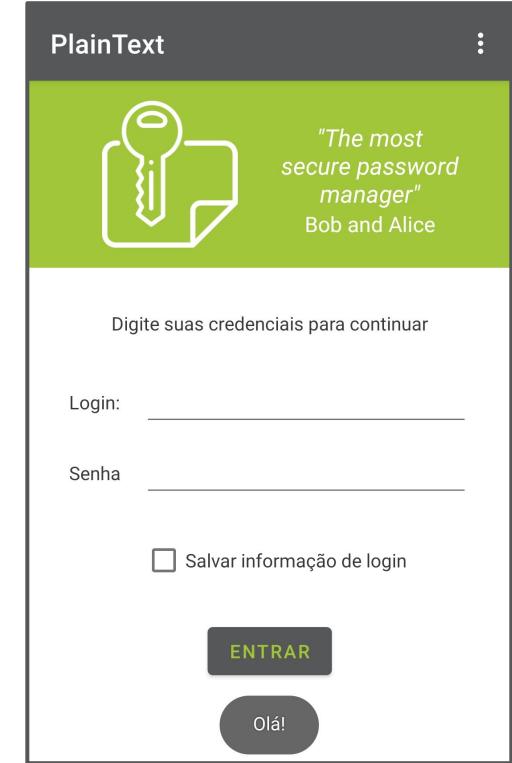


Método executado  
ao clicar

```
public class MainActivity extends AppCompatActivity {

    // onCreate, onCreateOptionsMenu ...

    public void entrarClicado(View view) {
        Toast.makeText(this, "Olá!", Toast.LENGTH_SHORT).show();
    }
}
```



# Eventos do Menu

- Quando uma opção do Menu é clicada, onOptionsItemSelected é executado passando, como argumento, o item que foi clicado

Qual o ID do item clicado?

Item do menu que foi clicado

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.about:  
            AlertDialog.Builder alert = new AlertDialog.Builder(this);  
            alert.setMessage("PlainText Password Manager v1.0")  
                .setNeutralButton("Ok", null)  
                .show();  
            return true;  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

Foi o item “sobre”?

Constrói e mostra uma Caixa de Alerta

# Eventos do Menu

PlainText



"The most secure password manager"  
Bob and Alice

Digite suas credenciais para continuar

Login: \_\_\_\_\_

Senha: \_\_\_\_\_

Salvar informação de login

**ENTRAR**

PlainText



Configurações

Sobre

"secure password manager"  
Bob and Alice

Digite suas credenciais para continuar

Login: \_\_\_\_\_

Senha: \_\_\_\_\_

Salvar informação de login

**ENTRAR**

PlainText



"The most secure password manager"  
Bob and Alice

Digite suas credenciais para continuar

PlainText Password Manager v1.0

OK

Salvar informação de login

**ENTRAR**

# Acessando os Componentes

---

- No XML dos Layouts, os componentes (botões, imagens, textos, etc) são acessados pelos seus IDs usando o “@”
  - Por isso, enfatizamos a necessidade de se atribuir os IDs aos componentes
- Mas para acessar tais componentes no Java, usa-se a classe “R”
- A classe R é gerada automaticamente e contém, dentre várias coisas, constantes para cada ID nos XMLs
  - Usaremos estas constantes para referenciar os componentes a partir do Java

# Acessando os Componentes

- Para acessar o componente, usamos o método `findViewById`, que tem como parâmetro o ID do componente (usando a classe R)

```
public void entrarClicado(View view) {  
    EditText editText = findViewById(R.id.editLogin);  
    String login = editText.getText().toString();  
    String msg = "Olá " + login + " !!";  
    Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();  
}
```

Acessa o componente

ID do componente

Objeto da classe  
EditText



# Abrindo outras Activities

- Um Intent provê uma ligação entre dois componentes que, em geral, são duas activities
  - Indica uma intenção de fazer alguma coisa

Intenção

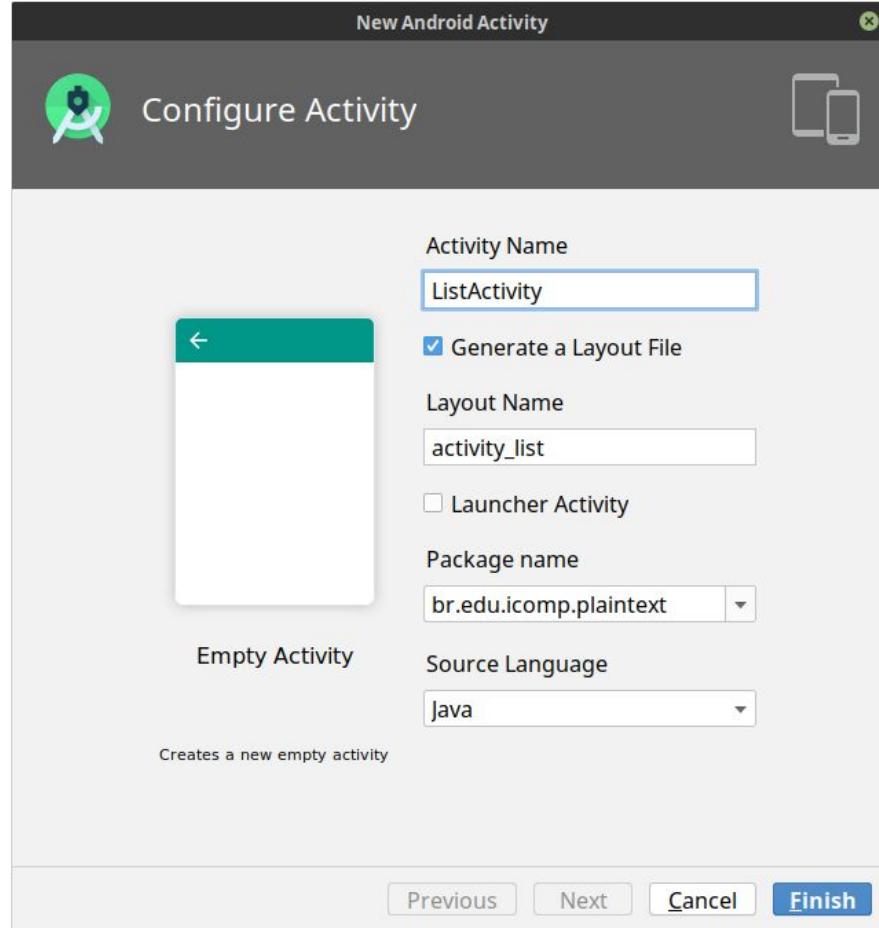
```
public void entrarClicado(View view) {  
    Intent intent = new Intent(this, ListActivity.class);  
    startActivity(intent);  
}
```

Execução

- A partir de agora, o botão “Entrar”, irá abrir uma nova activity
  - Entretanto, precisamos criar essa nova activity (ListActivity)*

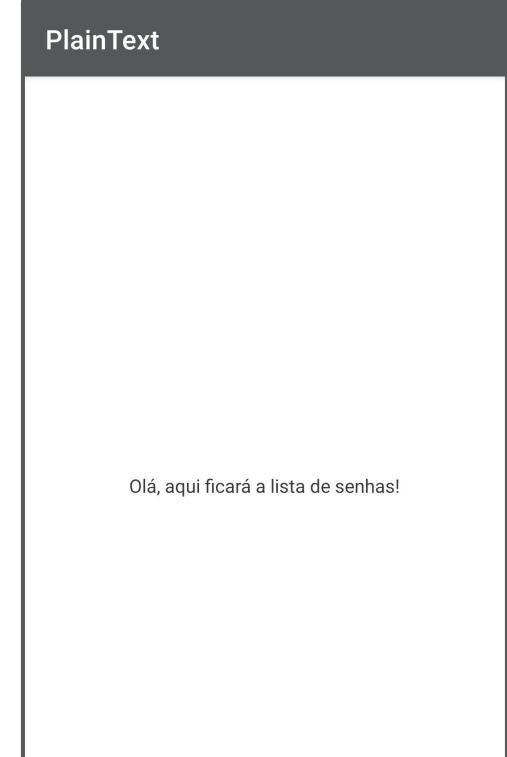
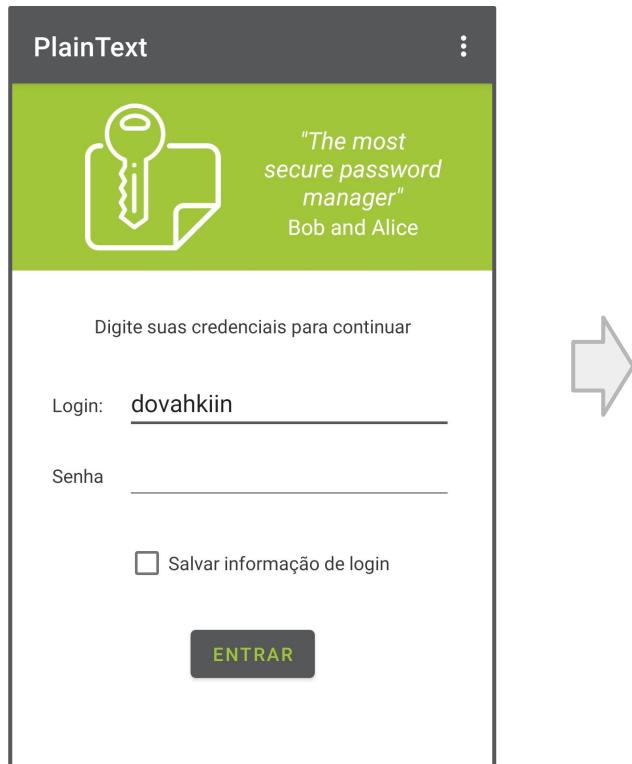
# Criando a Nova Activity

- ❑ No Android Studio
  - ❑ File
    - New
    - Activity
    - Empty Activity



# Testando a Nova Activity

- Ao clicar no botão “Entrar”, a nova activity é aberta



# Passando Dados entre Activities

- Para passar dados para a activity que será aberta, usa-se o método `putExtra` da classe Intent:

Acessa o login  
do usuário

Passando o  
login para a  
próxima  
activity

```
public void entrarClicado(View view) {  
    Intent intent = new Intent(this, ListActivity.class);  
  
    EditText inputLogin = findViewById(R.id.editLogin);  
    intent.putExtra("login", inputLogin.getText().toString());  
  
    startActivity(intent);  
}
```

Inicia a nova  
activity

# Passando Dados entre Activities

- Para acessar os dados na nova activity, usa-se o método `getExtra`:

onCreate da nova Activity

Acessa o texto de bem-vindo

Acessa o intent que abriu a activity

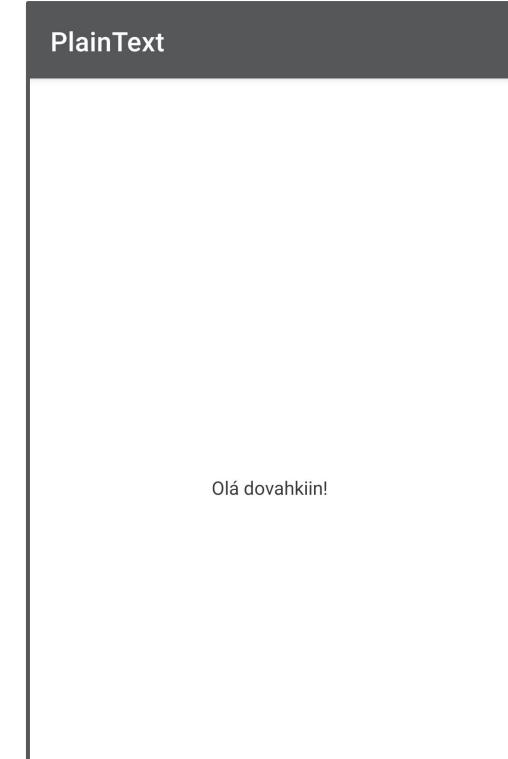
Acessa o valor do login passado

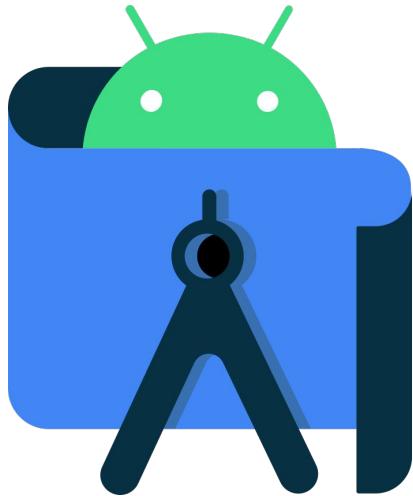
```
public class ListActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_list);  
  
        TextView textBemVindo = findViewById(R.id.textBemVindo);  
  
        Intent intent = getIntent();  
        String login = intent.getStringExtra("login");  
  
        textBemVindo.setText("Olá " + login + "!");  
    }  
}
```

Muda a mensagem de bem-vindo

# Passando Dados entre Activities

- Testando  
novamente o  
botão “Entrar”:





---

# Activity de Preferências

# Activity de Preferências

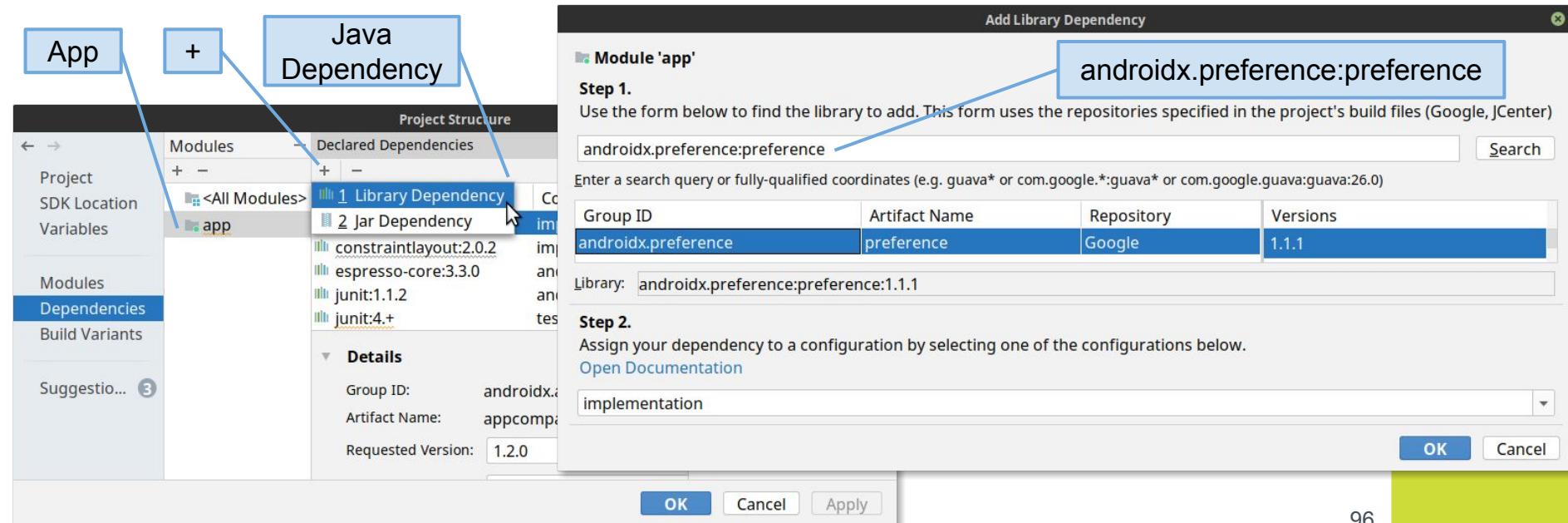
- As duas principais formas de armazenar dados em um app são:
  - Usando preferências
  - Usando banco de dados
- Por enquanto, iremos mostrar como criar preferências
  - Futuramente, veremos banco de dados
- Em nosso app de exemplo:
  - As informações de login para acessar o app serão salvas nas preferências
  - Os dados (itens) das senhas serão salvas no banco de dados

# Criando Preferências

- As preferências permitem armazenar dados de configurações da aplicação atual
- A forma mais prática de criar uma tela de configurações simples (com apenas uma tela), é
  - Criando um arquivo XML com as preferências
  - Criando uma activity que, ao ser criada, inicia um Fragmento de Preferências (classe PreferenceFragmentCompat), que provê todo o código necessário para:
    - *Mostrar as preferências de forma padronizada entre os diversos dispositivos*
    - *Salvar as configurações*

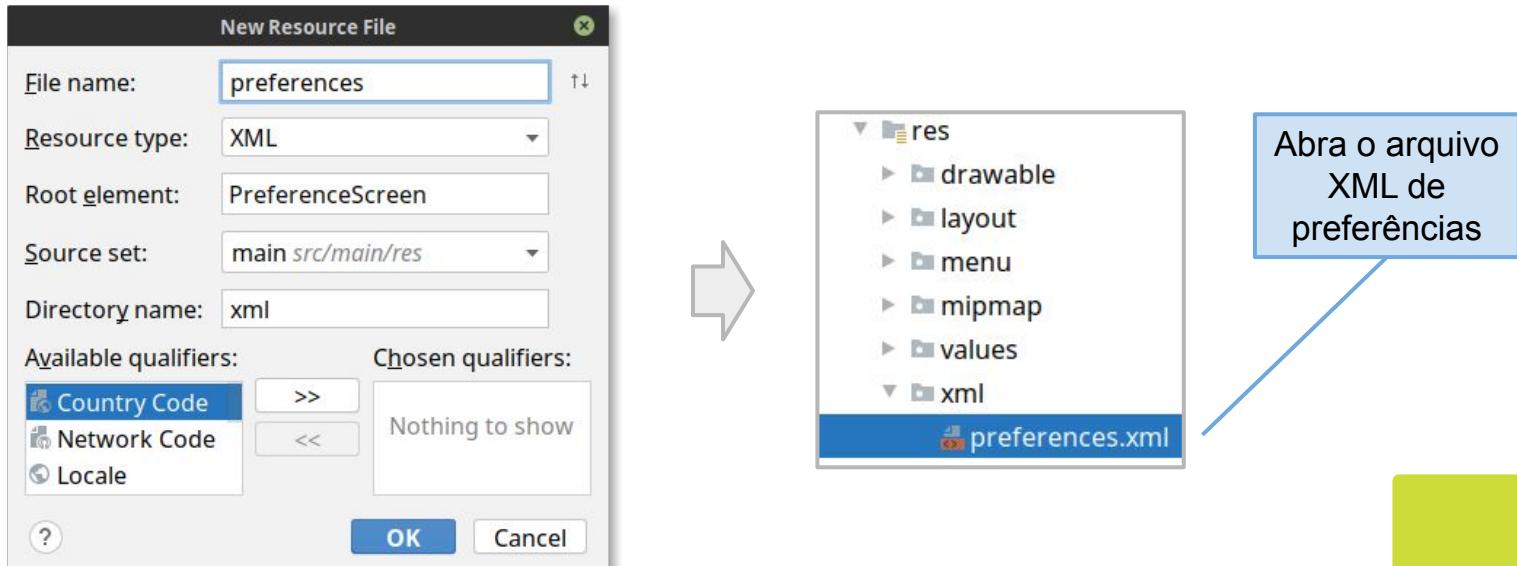
# Biblioteca de Preferências

- A parte de preferências do Android foi reimplementada
  - Devido a isso, é necessário incluir uma biblioteca no seu app:
    - Botão direito no App → Open Module Settings → Dependencies



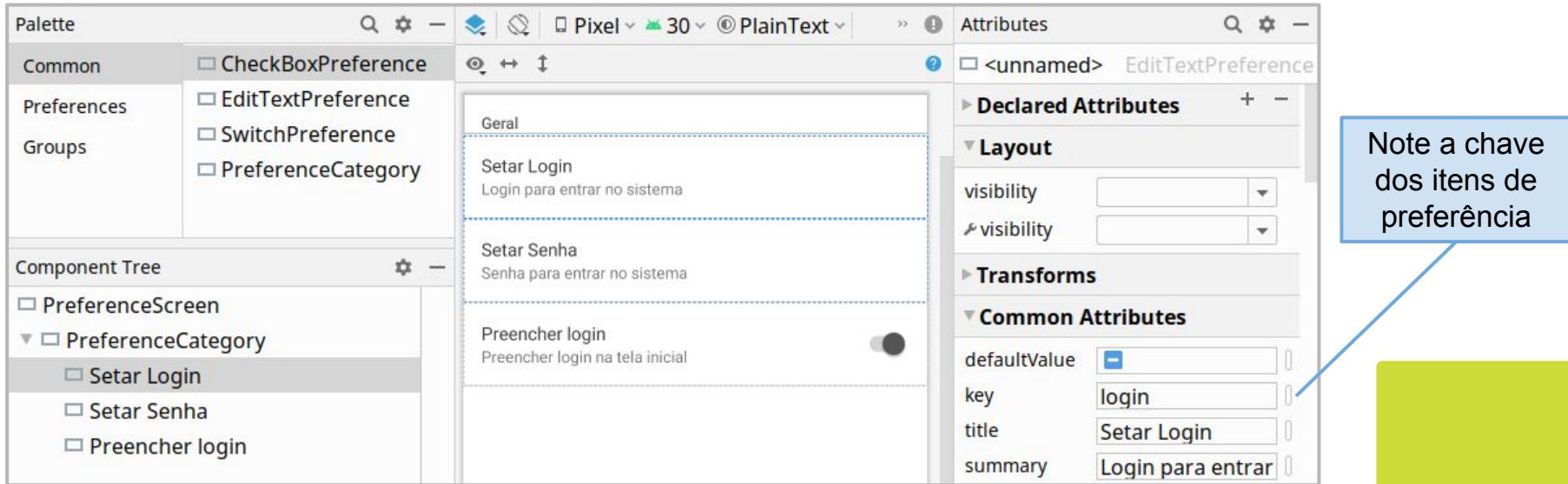
# XML de Preferências

- Para criar o arquivo XML de preferências:
  - File → New → Android Resource File



# XML de Preferências

- Assim como no Menu, você pode editar as preferências na tela de design ou editando o XML diretamente (próximo slide)
  - Editando na tela de design:



# XML de Preferências

## ■ XML resultante do slide anterior

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="Geral">
        <EditTextPreference
            android:dialogTitle="Setar Login"
            android:key="login"
            android:summary="Login para entrar no sistema"
            android:title="Setar Login" />
        <EditTextPreference
            android:dialogTitle="Setar Senha"
            android:key="password"
            android:dialogMessage="Digite a senha"
            android:inputType="textPassword"
            android:summary="Senha para entrar no sistema"
            android:title="Setar Senha" />
        <SwitchPreference
            android:key="showLogin"
            android:title="Preencher login"
            android:summary="Preencher login na tela inicial"
            android.defaultValue="true"/>
    </PreferenceCategory>
</PreferenceScreen>
```

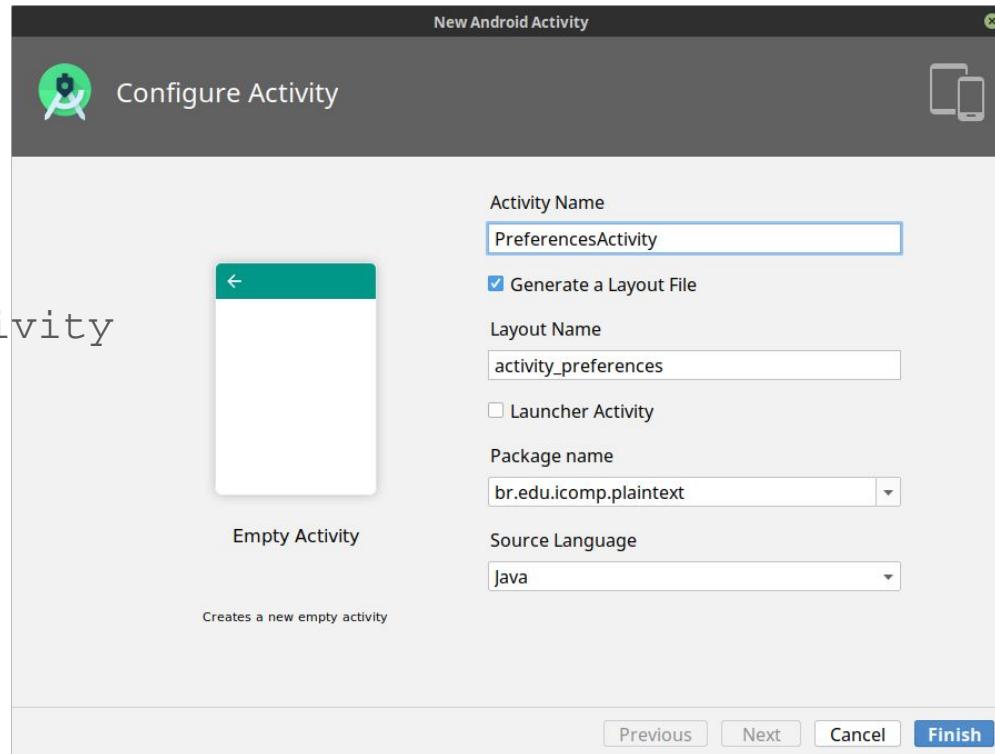
Note a chave  
dos itens de  
preferência

# Activity de Preferências

- Por fim, vamos criar a activity de preferências:

- File
    - New
    - Activity
    - Empty Activity

- Nome:  
PreferencesActivity



# Activity de Preferências

## PreferencesActivity.java

```
public class PreferencesActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        getSupportFragmentManager()
            .beginTransaction()
            .replace(android.R.id.content, new PreferencesFragment())
            .commit();
    }

    public static class PreferencesFragment extends PreferenceFragmentCompat {
        @Override
        public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
            setPreferencesFromResource(R.xml.preferences, rootKey);
        }
    }
}
```

# Iniciando a Activity de Preferências

- Para iniciar as configurações a partir da tela inicial, precisamos modificar o método `onOptionsItemSelected`:

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.about:  
            AlertDialog.Builder alert = new AlertDialog.Builder(this);  
            alert.setMessage("PlainText Password Manager v1.0")  
                .setNeutralButton("Ok", null).show();  
            return true;  
  
        case R.id.configs:  
            Intent intentConfig = new Intent(this, PreferencesActivity.class);  
            startActivity(intentConfig);  
            return true;  
  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

# Testando as Preferências

The image consists of five screenshots of a mobile application interface, likely demonstrating a penetration testing or security audit process.

- Screenshot 1:** The main screen shows a large green key icon and the title "PlainText". It has fields for "Login:" and "Senha:", a "Salvar informação de login" checkbox, and a "ENTRAR" button.
- Screenshot 2:** A modal titled "Configurações" is open, showing "Sobre" and a note about secure mail. It lists "PlainText", "Geral", "Setar Login", "Setar Senha", and "Preencher login".
- Screenshot 3:** A "Geral" settings screen with "Setar Login" and "Setar Senha" options.
- Screenshot 4:** A "Setar Login" screen with a text input containing "dovahkiin". A keyboard overlay is visible at the bottom.
- Screenshot 5:** A "Setar Senha" screen with a text input containing "Teste123". A keyboard overlay is visible at the bottom.

The application uses a light gray background with dark gray header bars. The "PlainText" logo is consistently present in the top left corner of each screen.

# Acessando as Preferências

```
public void entrarClicado(View view) {  
    SharedPreferences sharedpreferences = PreferenceManager.getDefaultSharedPreferences(this);  
    String prefLogin = sharedpreferences.getString("login", "");  
    String prefPass = sharedpreferences.getString("password", "");  
  
    String editLogin = ((EditText) findViewById(R.id.editLogin)).getText().toString();  
    String editPass = ((EditText) findViewById(R.id.editPassword)).getText().toString();  
  
    if (editLogin.equals(prefLogin) && editPass.equals(prefPass)) {  
        Intent intent = new Intent(this, LoginActivity.class);  
        EditText inputLogin = findViewById(R.id.editLogin);  
        intent.putExtra("login", inputLogin.getText().toString());  
        startActivity(intent);  
    }  
    else  
        Toast.makeText(this, "Login/senha inválidos!", Toast.LENGTH_SHORT).show();  
}
```

Acessa as preferências

Acessa o login salvo

Acessa a senha salva

Acessa os campos de texto de login e senha

Verifica o login/senha

# Acessando as Preferências

PlainText



"The most  
secure password  
manager"  
Bob and Alice

Digite suas credenciais para continuar

Login:

Senha

Salvar informação de login

**ENTRAR**

Login/senha inválidos!

PlainText



"The most  
secure password  
manager"  
Bob and Alice

Digite suas credenciais para continuar

Login:

Senha

Salvar informação de login

**ENTRAR**

PlainText

Olá dovahkiin!

Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



# Activities com Listas

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

 Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Listas

- Grande parte das aplicações para dispositivos móveis usam listas
  - GMail – Lista de e-mails
  - Calendar – Lista de eventos
  - Notes – Lista de notas de texto
  - ToDo – Lista de coisas a fazer
  - Twitter – Lista de tweets
  - Lemmy – Lista de conteúdo dos Lemmings
  - Facebook – Lista de posts dos amigos
- Por este motivo, o Android possui recursos avançados para a criação e gerenciamento de listas

# App de Exemplo

---

- Nesta parte do app PlainText, iremos implementar a lista de senhas cadastradas
- Inicialmente, os dados serão armazenados em um ArrayList
  - Futuramente, mudaremos a implementação para usar o banco de dados (SQLite)

## PlainText



# App de Exemplo

---

- Para isso, as seguintes classes serão criadas/modificadas na implementação:
  - Password
    - *Modelo de uma senha*
  - PasswordDAO
    - *Data Access Object das senhas*
  - ListActivity
    - *Será modificada para conter uma lista de senhas usando o componente RecyclerView*
  - EditActivity
    - *Controlará uma tela para criar uma senha ou ver e editar uma senha existente*

## PlainText



# Classe Password

- Uma senha será modelada através da classe Password, composta pelos seguintes atributos:
  - id: identificador (inteiro) da senha
  - name: um nome para a senha (e.g., nome do site)
  - login: login de acesso da senha
  - password: a senha a ser cadastrada
  - notes: observações gerais

C	Password
a	<i>id: int</i>
a	<i>name: String</i>
a	<i>login: String</i>
a	<i>password: String</i>
a	<i>notes: String</i>
M <sup>C</sup>	<i>Password(int id, String name, String login, String password, String notes)</i>
M	<i>Getters and Setters ...</i>

# Classe Password

- Código-fonte (parcial) da classe:

```
public class Password {  
    private int id;  
    private String name;  
    private String login;  
    private String password;  
    private String notes;  
  
    Password(int id, String name, String login, String password, String notes) {  
        this.id = id;  
        this.name = name;  
        this.login = login;  
        this.password = password;  
        this.notes = notes;  
    }  
  
    // Getters and Setters  
}
```

# Classe PasswordDAO

- Para implementar o armazenamento das senhas, criaremos a classe PasswordDAO
  - DAO - *Data Access Objects*, conforme feito nos slides de MySQL
- Inicialmente, esta classe irá salvar os dados em um ArrayList
  - Futuramente, mudaremos essa classe para usar banco de dados (SQLite)

# Classe PasswordDAO

- A classe PasswordDAO terá os seguintes métodos:
  - `ArrayList<Password> getList()`
    - *Retorna a lista de senhas cadastradas*
  - `boolean add(Password password)`
    - *Cadastra uma nova senha*
  - `boolean update(Password password)`
    - *Atualiza uma senha já existente*
  - `Password get(int id)`
    - *Retorna uma senha cadastrada através do seu id*
  - Para simplificar os slides, o método para remover um item não será implementado

# Classe PasswordDAO

## ■ Atributos e Construtor

```
public class PasswordDAO {  
  
    private Context context;  
    private static ArrayList<Password> passwordsList = new ArrayList<>();  
  
    public PasswordDAO(Context context) {  
        this.context = context;  
    }  
  
    // Classe continua com outros m...
```

Activity usando o DAO.  
Usado para toasts e BD.

Armazenamento das  
senhas

Construtor, recebe a activity  
(contexto) como parâmetro

# Classe PasswordDAO

## ■ Método getList

Se a lista estiver vazia, insere algumas senhas de teste

```
public ArrayList<Password> getList() {  
    if (passwordsList.size() == 0) {  
        passwordsList.add(new Password(0, "Facebook", "dovahkiin@gmail.com",  
                                         "FusRoDah123", ""));  
        passwordsList.add(new Password(1, "GMail", "dovahkiin",  
                                         "Teste123", ""));  
        passwordsList.add(new Password(2, "IComp", "dfrd@icomp.ufam.edu.br",  
                                         "Java4242", "Mudar a senha!"));  
        passwordsList.add(new Password(3, "Steam", "dovahkiin",  
                                         "FusRoDah123", "Conta do Brasil"));  
    }  
    return passwordsList;  
}
```

Retorna a lista de senhas

# Classe PasswordDAO

## ■ Método add

Seta o id da senha para ser um  
“autoincrement”

```
public boolean add>Password password) {  
    password.setId(passwordsList.size());  
    passwordsList.add(password);  
    Toast.makeText(context, "Senha salva!", Toast.LENGTH_SHORT).show();  
    return true;  
}
```

Salva a senha no ArrayList

# Classe PasswordDAO

- Métodos update e get

```
public boolean update>Password password) {  
    passwordsList.set(password.getId(), password);  
    Toast.makeText(context, "Senha atualizada!", Toast.LENGTH_SHORT).show();  
    return true;  
}  
  
public Password get(int id) {  
    return passwordsList.get(id);  
}
```

# Voltando para a Lista

---

- Agora que temos as classes auxiliares, vamos ver como criar uma lista na activity `ListActivity`, que criamos anteriormente
- A forma recomendada de se criar listas no Android é usando a biblioteca *RecyclerView*
  - Ele é mais complexo que as formas anteriores
  - Mas é muito mais poderoso e eficiente

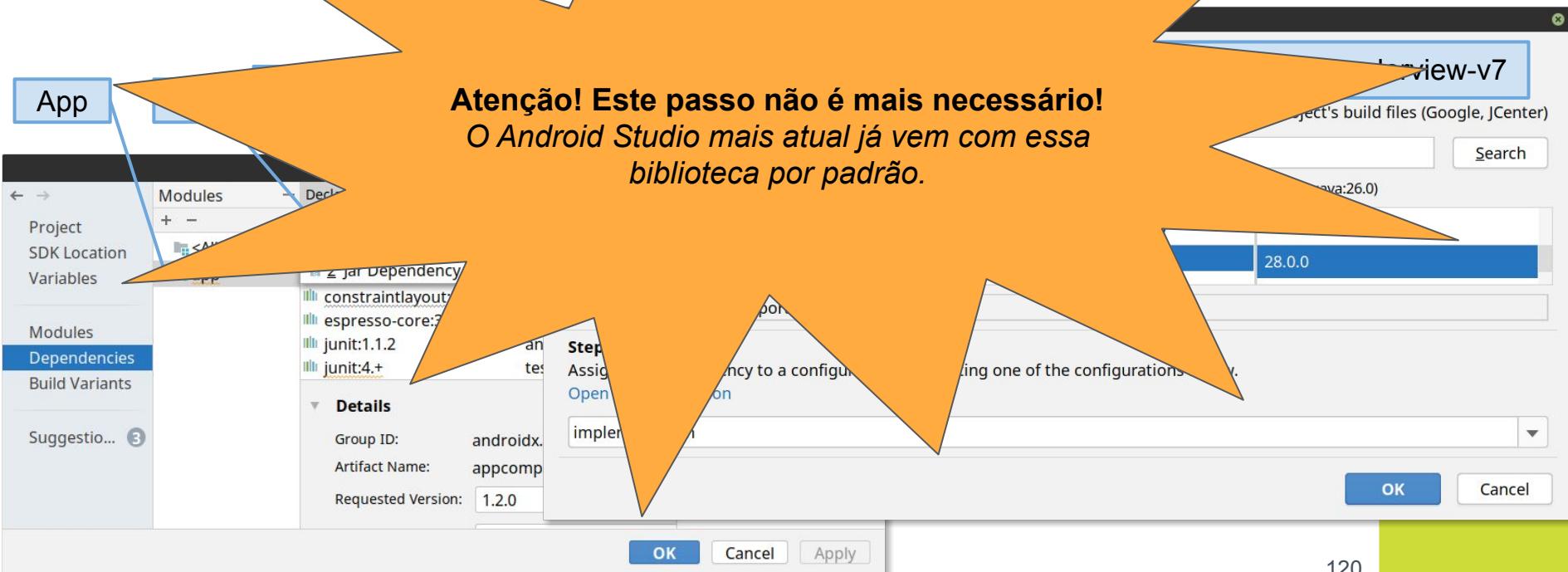
# RecyclerView

- Para se criar a lista usando o *RecyclerView*, deve-se
  1. Instalar a biblioteca
  2. Criar um componente *RecyclerView* no layout da activity (XML)
  3. Criar o layout de um item da lista (XML)
  4. Na activity, criar uma classe pro adaptador (*Adapter*)
    - O adaptador faz a ligação entre os dados, vindos do *PasswordDAO*, e a lista, que é o componente *RecyclerView*
  5. Na activity, criar uma classe pro *Holder*
    - O *Holder*, armazena as informações de um item da lista

# 1. Instalando o RecyclerView

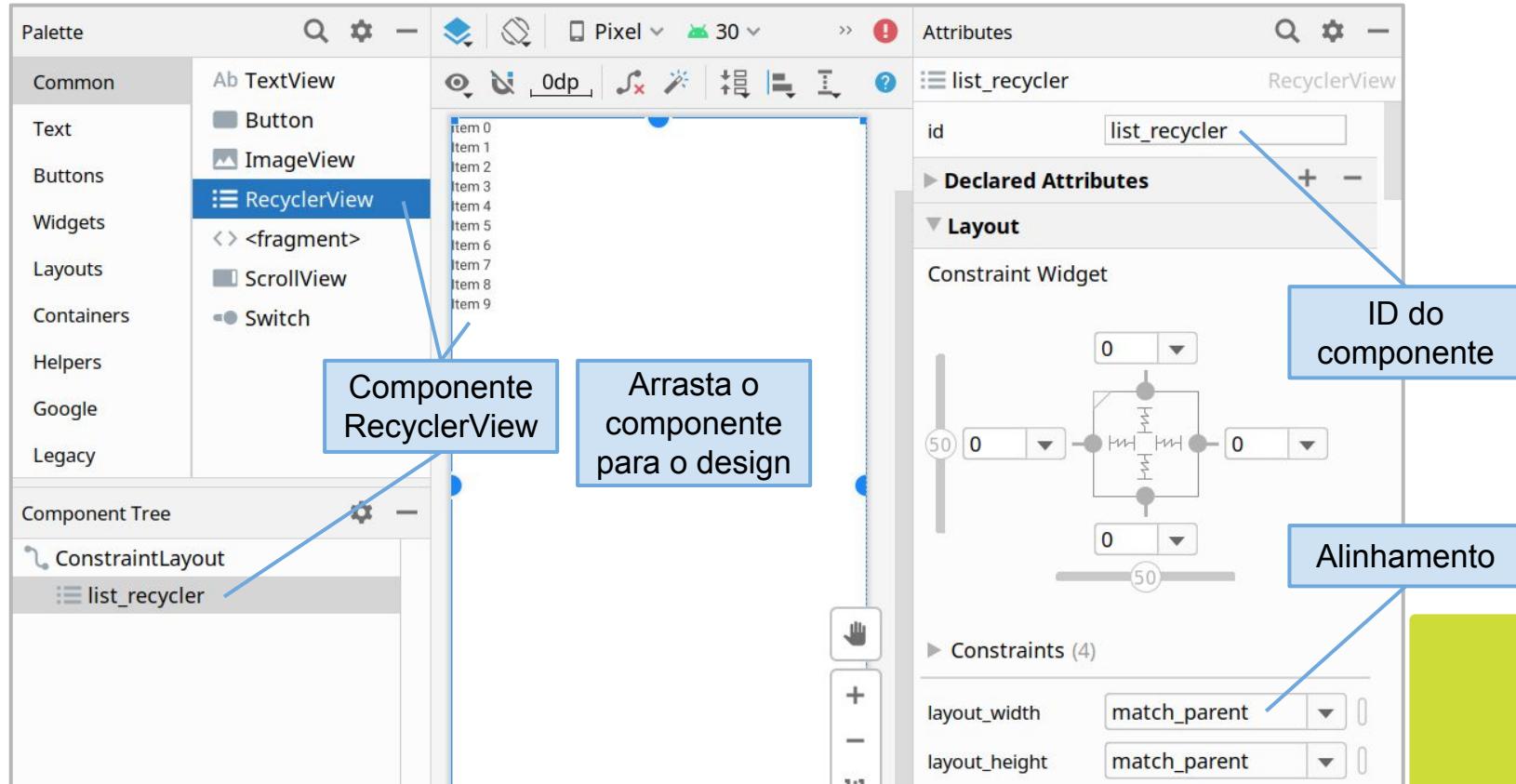
- Para incluir a biblioteca no seu app:
  - Botão Adicionar no App → Modules → Dependências

**Atenção! Este passo não é mais necessário!**  
O *Android Studio* mais atual já vem com essa  
biblioteca por padrão.



## 2. RecyclerView no Layout

■ Arquivo: res/layout/activity\_list.xml



## 2. RecyclerView no Layout

- XML do slide anterior

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    tools:context=".ListActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list_recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

### 3. Layout do Item da Lista

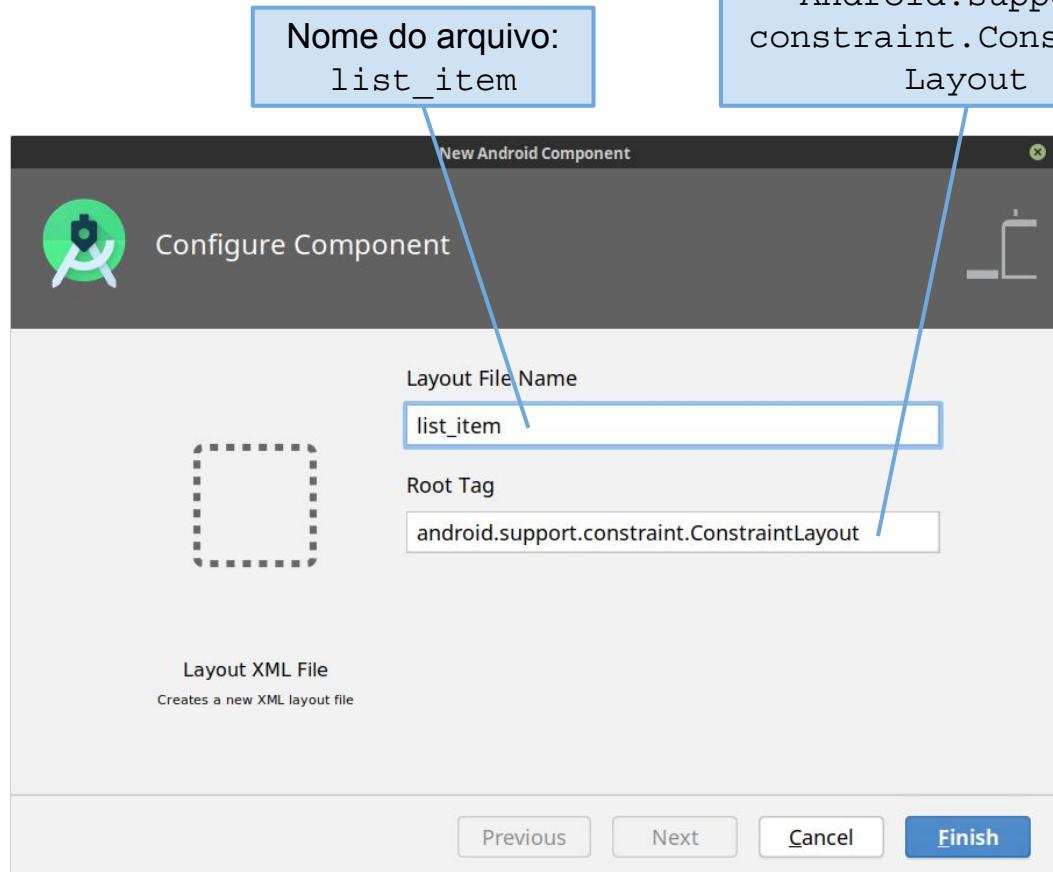
---

- Uma lista é composta por um conjunto de itens
- Todos os items possuem a mesma formatação (layout)
  - Mas com “valores” diferentes
- O próximo passo para criar a lista é criar uma formatação para os itens dessa lista
  - Essa formatação é feita através de um layout (XML)

# 3. Layout do Item da Lista

- Para criar o novo layout dos itens:

- File
    - New
    - XML
    - Layout
    - XML File

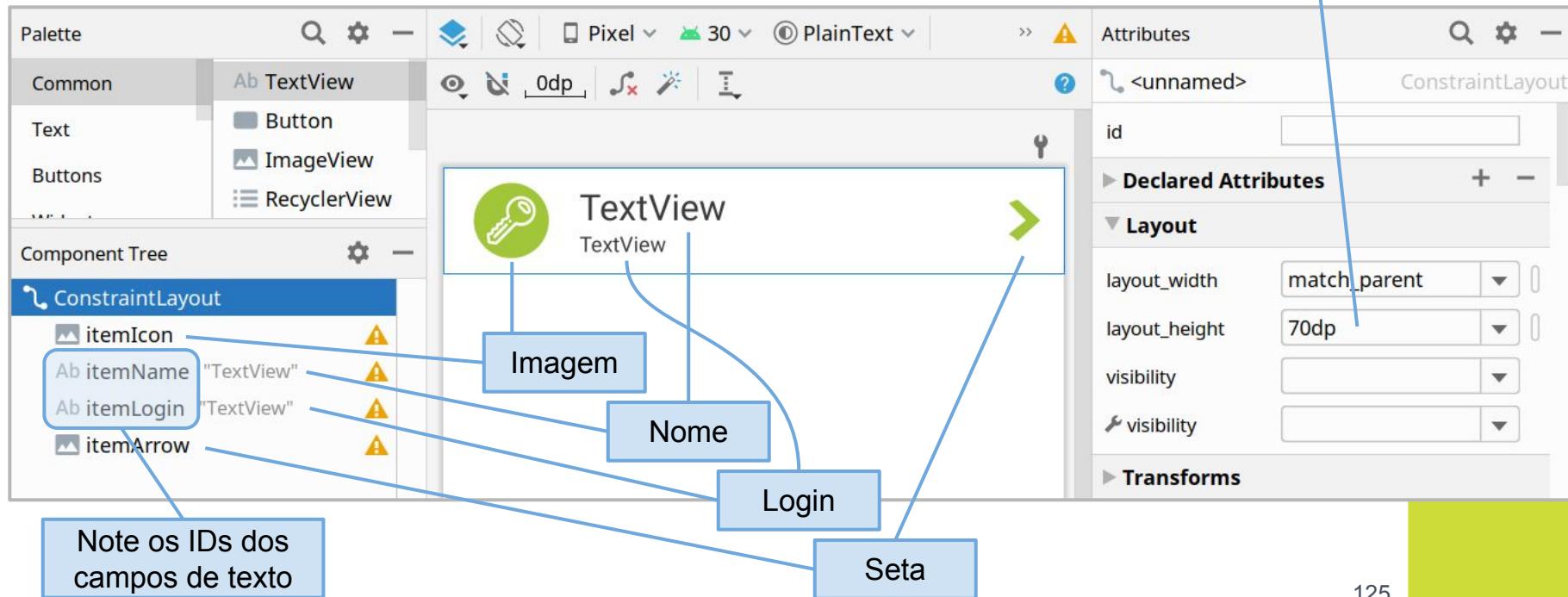


# 3. Layout do Item da Lista

## Design do layout:

- Arquivo res/layout/list\_item.xml

Altura de cada item



Note os IDs dos  
campos de texto

# 3. Layout do Item da Lista (XML)

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent" android:layout_height="70dp"
    android:background="?attr/selectableItemBackground">
    <ImageView
        android:id="@+id/itemIcon" android:layout_width="50dp" android:layout_height="50dp"
        android:layout_marginStart="20dp" app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_item_key" />
    <TextView
        android:id="@+id/itemName" android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_marginStart="20dp" android:text="TextView" android:textColor="#39393a" android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@+id/itemLogin" app:layout_constraintStart_toEndOf="@+id/itemIcon"
        app:layout_constraintTop_toTopOf="@+id/itemIcon" />
    <TextView
        android:id="@+id/itemLogin" android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="TextView" android:textColor="#39393a" android:textSize="14sp"
        app:layout_constraintBottom_toBottomOf="@+id/itemIcon" app:layout_constraintStart_toStartOf="@+id/itemName" />
    <ImageView
        android:id="@+id/itemArrow" android:layout_width="55dp" android:layout_height="55dp"
        app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" app:srcCompat="@drawable/ic_right_arrow" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

# 4. Adaptador

- Um adaptador **conecta dados a um componente**
  - Dados: no nosso exemplo, os dados vêm do método `getList` da classe `PasswordDAO`, explicado nos slides anteriores
  - Componente: É o *RecyclerView* adicionado no layout da activity
    - Componente com ID `list_recycler` do arquivo `res/layout/activity_list.xml`
- Vamos primeiro mostrar o uso do adaptador no `onCreate` da activity para, em seguida, implementá-lo

# 4. Uso do Adaptador

```
public class ListActivity extends AppCompatActivity {  
    private RecyclerView recyclerView;  
    private PasswordsAdapter adapter;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_list);  
  
        recyclerView = findViewById(R.id.list_recycler);  
        recyclerView.setLayoutManager(new LinearLayoutManager(this));  
  
        adapter = new PasswordsAdapter(this);  
        recyclerView.setAdapter(adapter);  
    }  
  
    @Override  
    protected void onRestart() {  
        super.onRestart();  
        adapter.update();  
        adapter.notifyDataSetChanged();  
    }  
}
```

Acessa o *RecyclerView* do Layout (XML)

Seta o Layout do *RecyclerView*

Seta o adaptador (criado nos slides seguintes)

Atualiza os dados da lista quando a activity for reiniciada

# 4. Criação do Adaptador

- No nosso app de exemplo, a classe do adaptador que criaremos se chamará `PasswordsAdapter`
  - Como essa classe só será usada pela `ListActivity`, criaremos essa classe no mesmo arquivo (`ListActivity.java`)
- Para implementar um adaptador para o `RecyclerView`, herda-se a classe `RecyclerView.Adapter`
  - Essa classe é uma classe genérica (assim como as coleções genéricas)
  - E tem como parâmetro a classe que será usada para representar os itens da lista.
    - Esta classe é conhecida como *Holder*
    - Será implementada futuramente

# 4. Criação do Adaptador

- A classe do adaptador implementará os seguintes métodos:
  - Construtor
  - update
    - *atualiza os dados do adaptador*
  - onCreateViewHolder
    - *chamada quando uma linha da lista for criada pela primeira vez*
  - onBindViewHolder
    - *chamada quando uma linha já existente for “reciclada”, para conter informações de outro elemento da lista*
  - getItemCount
    - *retorna a quantidade total de elementos da lista*

# 4. Criação do Adaptador

```
class PasswordsAdapter extends RecyclerView.Adapter<PasswordsViewHolder> {  
    private Context context;  
    private ArrayList<Password> passwords;  
    PasswordDAO passwordDAO;  
  
    public PasswordsAdapter(Context context) {  
        this.context = context;  
        passwordDAO = new PasswordDAO(context);  
        update();  
    }  
  
    public void update() { passwords = passwordDAO.getList(); }  
  
    public PasswordsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        ConstraintLayout v = (ConstraintLayout) LayoutInflater  
            .from(parent.getContext())  
            .inflate(R.layout.list_item, parent, false);  
        PasswordsViewHolder vh = new PasswordsViewHolder(v, context);  
        return vh;  
    }  
  
    public void onBindViewHolder(PasswordsViewHolder holder, int position) {  
        holder.name.setText(passwords.get(position).getName());  
        holder.login.setText(passwords.get(position).getLogin());  
        holder.id = passwords.get(position).getId();  
    }  
  
    public int getItemCount() { return passwords.size(); }  
}
```

Classe Holder, explicado no próximo slide

Activity, necessário para acessar o BD, Toasts, e abrir uma nova activity

Pega a lista de senhas cadastradas

Chamado quando um item é criado pela 1a vez

Infla o layout do item (list\_item)

Cria e retorna um objeto da classe  
PasswordsViewHolder

Atualiza os textos de um item (holder) de acordo com sua posição na lista

# 5. Criação do *Holder*

---

- Como mostrado no slide anterior, a classe *holder* se chama `PasswordsViewHolder`
  - Assim como no adaptador, como essa classe só será usada pela `ListActivity`, criaremos ela no mesmo arquivo (`ListActivity.java`)
- Para implementar um *holder*, herda-se a classe `RecyclerView.ViewHolder`
  - Como será possível “clicar” em um item, esta classe irá implementar a interface `View.OnClickListener`
  - Neste caso, deve-se implementar o método `onClick`

# 5. Criação do Holder

```
class PasswordsViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
    public Context context;  
    public TextView login, name;  
    public int id;  
  
    public PasswordsViewHolder(ConstraintLayout v, Context context) {  
        super(v);  
        this.context = context;  
        name = v.findViewById(R.id.itemName);  
        login = v.findViewById(R.id.itemLogin);  
        v.setOnClickListener(this);  
    }  
  
    public void onClick(View v) {  
        Toast.makeText(context, "Olá " + this.login.getText().toString(), Toast.LENGTH_LONG)  
            .show();  
    }  
}
```

Acessa os dados da view (list\_item.xml) do item atual

Ao clicar em um item, o método onClick dele será chamado

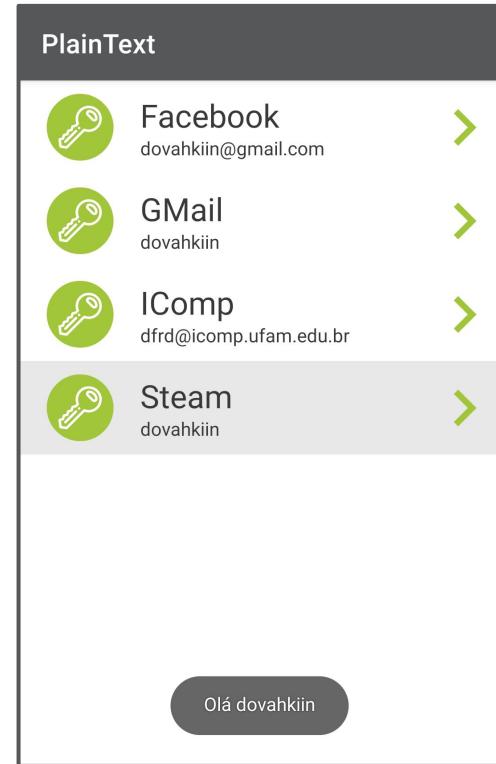
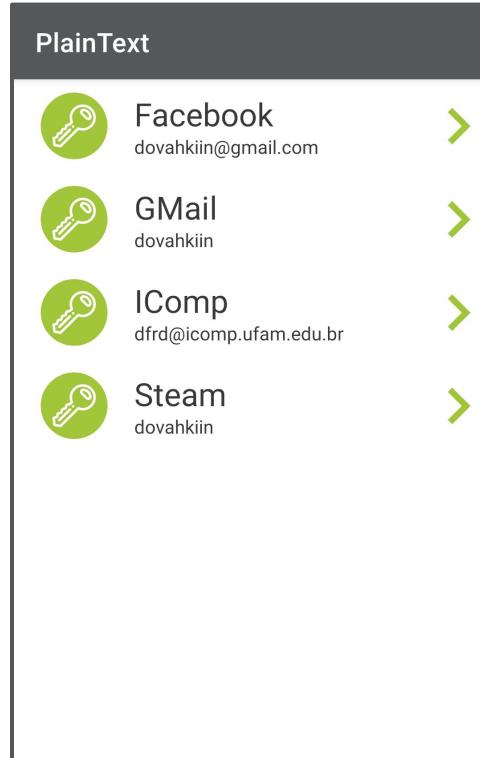
# Resultado

---

- Testando o *RecyclerView*

# Testando o RecyclerView

## Resultado



# Finalizando o App

- Para finalizar as funcionalidades do App, será implementada a parte para adicionar uma nova senha
  - Até o momento, as senhas foram adicionadas no código-fonte
- Para isso, iremos:
  1. Modificar a `ListActivity` para :
    - *Ter um botão de adicionar senha (irá abrir uma nova activity)*
    - *Abrir a nova activity para visualizar a senha ao clicar em um item da lista*
  2. Criar uma nova activity para adicionar, editar e visualizar as senhas
    - *A mesma activity será usada para adicionar, editar e visualizar uma senha*

# Adicionando o Botão de Adicionar

## ■ FloatingActionButton no ListActivity

The screenshot shows the Android Studio Layout Editor interface. On the left, the Palette lists various UI components under categories like Common, Text, Buttons, Widgets, etc. The 'Buttons' category is currently selected, and the 'FloatingActionButton' item is highlighted with a blue selection bar. In the center, the main workspace displays a 'Constraint Layout' containing a 'list\_recycler' (recycler view) and a 'buttonAdd' (FloatingActionButton). A blue callout box highlights the 'buttonAdd' component. To the right, the 'Attributes' panel is open for the 'buttonAdd' object. It shows the following configuration:

buttonAdd	FloatingActionButton
id	buttonAdd
Declared Attributes	
Layout	
Constraint Widget	
nextFocusUp	0
onClick	buttonAddClick
orientation	
Constraints (2)	
layout_width	wrap_content
layout_height	wrap_content
visibility	
✓ visibility	

A blue callout box also points to the 'onClick' attribute, which is set to 'buttonAddClick'. A larger blue callout box encloses the entire 'buttonAdd' component in the layout.

Método  
executado ao  
clicar no botão

# Adicionando o Botão de Adicionar

## ■ XML do slide anterior

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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" tools:context=".ListActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list_recycler" android:layout_width="match_parent" android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/buttonAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:onClick="buttonAddClick"
        app:fabCustomSize="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:srcCompat="@android:drawable/ic_input_add" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

# Adicionando o Botão de Adicionar

- Implementação do evento ao clicar no botão
  - Arquivo: ListActivity.java
  - Inicia a nova activity para adicionar/editar/visualizar (EditActivity)

```
public void buttonAddClick(View view) {  
    Intent intent = new Intent(this, EditActivity.class);  
    startActivity(intent);  
}
```

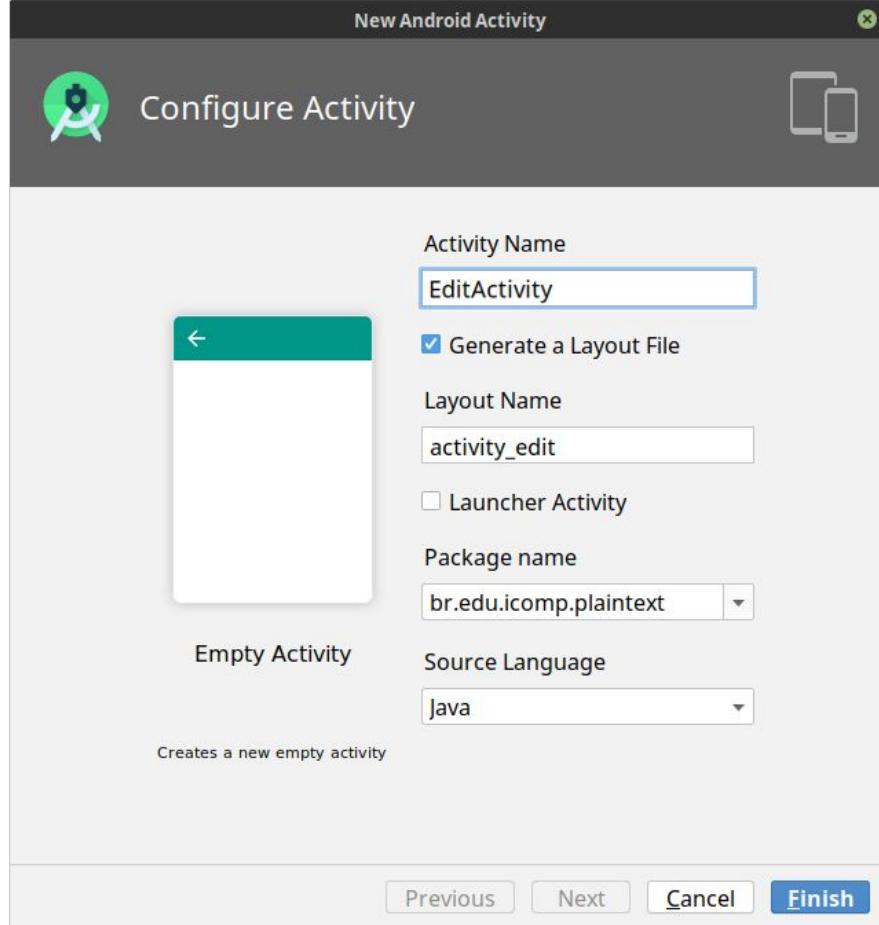
# Evento de Clique no Item da Lista

- Implementação do evento ao clicar em um item da lista
  - Arquivo: ListActivity.java
  - Dentro da classe PasswordsViewHolder
  - Inicia a nova activity para adicionar/editar/visualizar (EditActivity)

```
public void onClick(View v) {  
    Intent intent = new Intent(context, EditActivity.class);  
    intent.putExtra("passwordId", this.id);  
    context.startActivity(intent);  
}
```

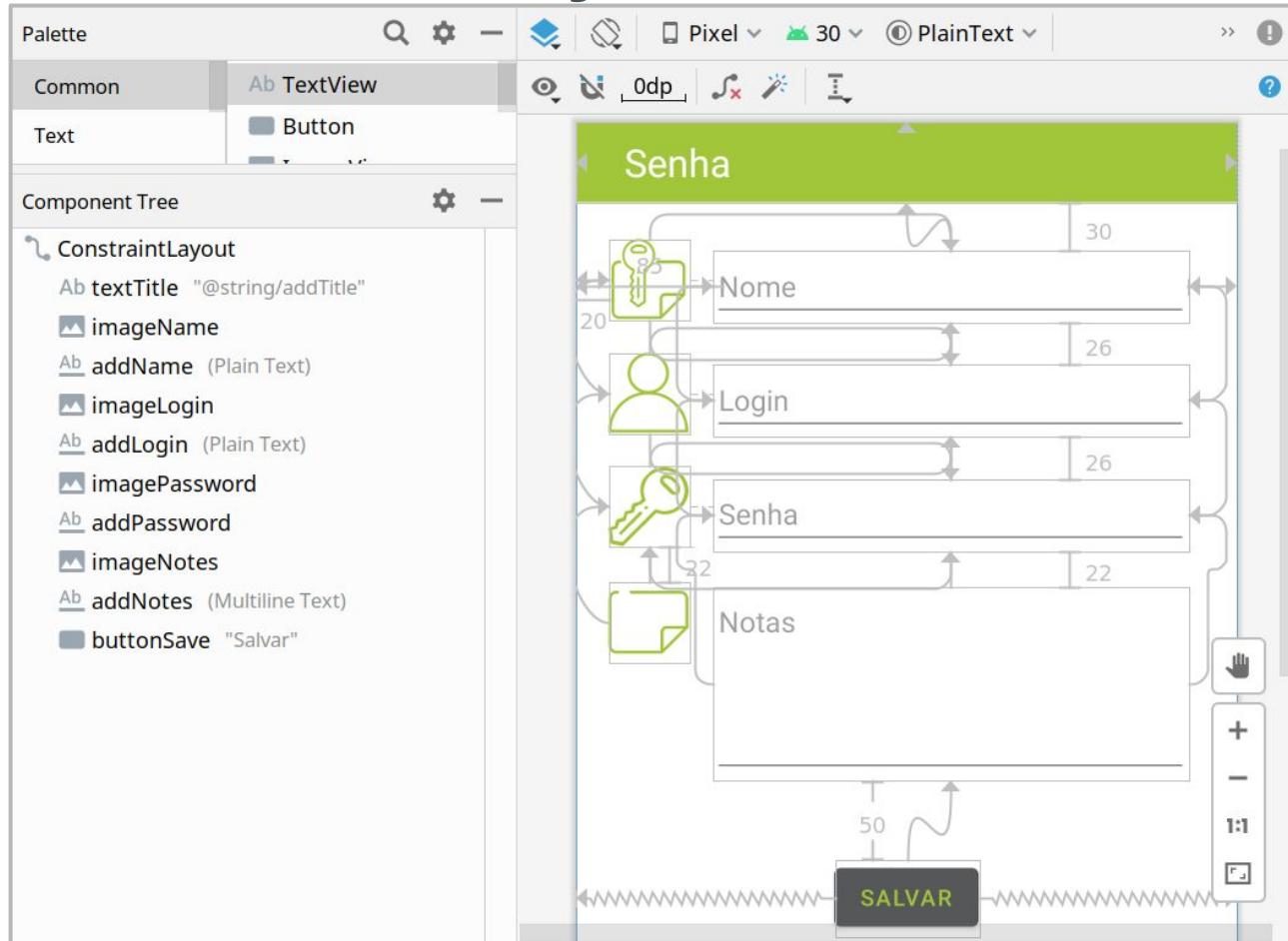
# Criando a Nova Activity

- ❑ No Android Studio
  - ❑ File
    - New
    - Activity
    - Empty Activity



# Criando a Nova Activity

## Layout



# Criando a Nova Activity

## XML do slide anterior

```
<?xml version="1.0" encoding="utf-8
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    tools:context=".EditActivity">
    <TextView
        android:id="@+id/textTitle"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:background="#a6c639"
        android:gravity="left|center"
        android:text="@string/addTitle"
        android:textAlignment="gravity"
        android:textColor="#ffff00"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
    <ImageView
        android:id="@+id/imageName"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginStart="10dp"
        android:layout_marginBottom="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_add_name"/>
    <EditText
        android:id="@+id/addName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="30dp"
        android:layout_marginBottom="30dp"
        android:ems="10"
        android:hint="@string/addName"
        android:inputType="textPersonName"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="parent"/>
    <EditText
        android:id="@+id/addLogin"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="260dp"
        android:layout_marginEnd="30dp"
        android:layout_marginBottom="10dp"
        android:ems="10"
        android:hint="Senha"
        android:inputType="textPassword"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="parent"/>
    <EditText
        android:id="@+id/addPassword"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="26dp"
        android:layout_marginEnd="30dp"
        android:layout_marginBottom="10dp"
        android:ems="10"
        android:hint="Senha"
        android:inputType="textPassword"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="parent"/>
    <Button
        android:id="@+id/buttonSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:onClick="salvarClicado"
        android:text="Salvar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="parent"/>

```

```
<EditText
    android:id="@+id/addName"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="30dp"
    android:layout_marginEnd="30dp"
    android:layout_marginBottom="30dp"
    android:ems="10"
    android:hint="@string/addName"
    android:inputType="textPersonName"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<EditText
    android:id="@+id/addLogin"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="260dp"
    android:layout_marginEnd="30dp"
    android:layout_marginBottom="10dp"
    android:ems="10"
    android:hint="Senha"
    android:inputType="textPassword"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<EditText
    android:id="@+id/addPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="26dp"
    android:layout_marginEnd="30dp"
    android:layout_marginBottom="10dp"
    android:ems="10"
    android:hint="Senha"
    android:inputType="textPassword"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<ImageViews
    android:id="@+id/imageLogin"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginBottom="10dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/ic_add_login"/>
<ImageViews
    android:id="@+id/imagePassword"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginBottom="10dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/ic_add_password"/>
<EditText
    android:id="@+id/addNotes"
    android:layout_width="0dp"
    android:layout_height="120dp"
    android:layout_marginTop="22dp"
    android:gravity="start|top"
    android:hint="Notas"
    android:inputType="textMultiLine"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<EditText
    android:id="@+id/addNotes"
    android:layout_width="0dp"
    android:layout_height="120dp"
    android:layout_marginTop="22dp"
    android:gravity="start|top"
    android:hint="Notas"
    android:inputType="textMultiLine"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<Button
    android:id="@+id/buttonSave"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:onClick="salvarClicado"
    android:text="Salvar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>

```

```
<ImageViews
    android:id="@+id/imagePassword"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginBottom="10dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@drawable/ic_add_password"/>
<EditText
    android:id="@+id/addPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="26dp"
    android:ems="10"
    android:hint="Senha"
    android:inputType="textPassword"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<ImageViews
    android:id="@+id/imageNotes"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_marginTop="22dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"
    app:srcCompat="@drawable/ic_add_notes"/>
<EditText
    android:id="@+id/addNotes"
    android:layout_width="0dp"
    android:layout_height="120dp"
    android:layout_marginTop="22dp"
    android:gravity="start|top"
    android:hint="Notas"
    android:inputType="textMultiLine"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>
<Button
    android:id="@+id/buttonSave"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:onClick="salvarClicado"
    android:text="Salvar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="parent"/>

```

# Criando a Nova Activity

## ■ EditActivity

```
public class EditActivity extends AppCompatActivity {  
    private PasswordDAO passwordDAO;  
    private int passwordId;  
    private TextView editName, editLogin, editPassword, editNotes;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_edit);  
  
        editName = findViewById(R.id.addName);  
        editLogin = findViewById(R.id.addLogin);  
        editPassword = findViewById(R.id.addPassword);  
        editNotes = findViewById(R.id.addNotes);  
        passwordDAO = new PasswordDAO(this);  
  
        Intent intent = getIntent();  
        passwordId = intent.getIntExtra("passwordId", -1);  
  
        // Verifica se uma senha foi passada como parâmetro  
        if (passwordId != -1) {  
            Password password = passwordDAO.get(passwordId);  
            editName.setText(password.getName());  
            editLogin.setText(password.getLogin());  
            editPassword.setText(password.getPassword());  
            editNotes.setText(password.getNotes());  
        }  
    }  
    // Método saveClicked (próximo slide)
```

Acessa os componentes da tela (e atribui para os atributos da classe)

DAO para gerenciar as senhas

Tenta acessar o passwordId, enviado ao abrir a activity

Se o passwordId foi enviado, pega os dados da senha e preenche os campos de texto da tela

# Clicando no Botão de Salvar

- Ao clicar no botão de salvar, o seguinte método será executado
  - Arquivo EditActivity.java, dentro da classe EditActivity

```
public void salvarClicado(View view) {  
    Password password = new Password(passwordId, editName.getText().toString(),  
        editLogin.getText().toString(), editPassword.getText().toString(),  
        editNotes.getText().toString());  
  
    boolean result;  
    if (passwordId == -1) result = passwordDAO.add(password);  
    else result = passwordDAO.update(password);  
  
    if (result) finish();  
}
```

Cria um novo Password (modelo)

É uma nova senha, adiciona

É uma senha já existente, atualiza

Se deu certo, volta para a activity anterior (lista)

# Resultado

---

- Testando a nova activity

# Testando a Nova Activity

## Resultado

PlainText

- Facebook dovahkiin@gmail.com >
- GMail dovahkiin >
- IComp dfrd@icomp.ufam.edu.br >
- Steam dovahkiin >

+

PlainText

Senha

	Coursera
	sovngarde
	HunKaalZoor
	Notas

SALVAR

PlainText

- Facebook dovahkiin@gmail.com >
- GMail dovahkiin >
- IComp dfrd@icomp.ufam.edu.br >
- Steam dovahkiin >
- Coursera sovngarde >

+

# Conclusão

---

- Ao reiniciar o aplicativo, os dados inseridos são perdidos, pois eles não foram armazenados, estavam apenas na memória
- A seguir, iremos modificar o PasswordDAO para salvar os dados no banco de dados (SQLite), de forma que os dados fiquem armazenados permanentemente

Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



---

# Banco de Dados SQLite

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Banco de Dados

- O Banco de Dados do Android é o SQLite
  - Suporta a comandos SQL, Leve
  - Dados são salvos em um único arquivo local
  - Já vem instalado
    - *Sem necessidade de usuário e senha. Entretanto, seus dados estão seguros, pois só serão acessíveis por sua app*
  - <http://www.sqlite.org/>

# Banco de Dados

---

- Para utilizar o banco de dados no app, iremos:
  - Criar uma nova classe auxiliar para usar o banco de dados
    - *Database.java*
  - Modificar a classe PasswordDAO para acessar os dados do banco de dados
    - *Ao invés de usar o ArrayList, como feito nos slides anteriores*

# Auxiliar de Banco de Dados

---

- ❑ Nome da classe: Database
- ❑ Esta classe estende a classe SQLiteOpenHelper
- ❑ Terá constantes e métodos para auxiliar o acesso ao BD
  - ❑ Explicados a seguir
- ❑ Ela será instanciada dentro de cada classe DAO
  - ❑ DAO: Data Access Objects

# Auxiliar de Banco de Dados

## Constantes:

- DATABASE\_VERSION
  - *Versão do banco de dados. Incremente esse valor sempre que fizer modificação na estrutura do banco de dados ou quando quiser limpar o banco de dados (nessa implementação).*
  - *Sempre que o android perceber que a versão dessa constante é maior que a última versão do BD no seu celular, ele irá executar o método onUpdate, mostrado no próximo slide*
- DATABASE\_NAME
  - *Nome do arquivo do banco de dados*
- SQL\_CREATE\_PASS, SQL\_POPULATE\_PASS, SQL\_DELETE\_PASS
  - *Comandos SQL para criar, popular e remover a tabela de senhas*
  - *Crie constantes diferentes para cada uma das tabelas do seu app*

# Auxiliar de Banco de Dados

## ■ Métodos:

- Database
  - *Construtor, recebe o contexto (Activity) como parâmetro*
  - *Executa o construtor da classe pai (SQLiteOpenHelper) passando o contexto, e o nome e a versão do banco de dados*
- onCreate
  - *Método executado quando o aplicativo é iniciado e nenhum banco de dados existe ainda*
  - *Provavelmente o aplicativo acabou de ser instalado*
- onUpgrade
  - *Executado quando a versão do banco de dados do app atual é diferente da versão do banco de dados no arquivo do celular*
  - *Provavelmente o aplicativo foi atualizado, será necessário executar comandos SQL para “converter” o BD do celular para a versão mais nova*
  - *No exemplo a seguir, apagamos a tabela e criamos novamente*

# Auxiliar de Banco de Dados

```
public class Database extends SQLiteOpenHelper {  
    public static final int DATABASE_VERSION = 4; Versão do banco de dados  
    public static final String DATABASE_NAME = "PlainText.db";  
    private static final String SQL_CREATE_PASS = "CREATE TABLE passwords (" +  
        "id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, login TEXT," +  
        "password TEXT, notes TEXT)";  
    private static final String SQL_POPULATE_PASS = "INSERT INTO passwords VALUES " +  
        "(NULL, 'GMail', 'dovahkiin', 'Teste123', 'Nota de Teste')";  
    private static final String SQL_DELETE_PASS = "DROP TABLE IF EXISTS passwords";  
  
    public Database(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL(SQL_CREATE_PASS);  
        db.execSQL(SQL_POPULATE_PASS);  
    }  
  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        db.execSQL(SQL_DELETE_PASS);  
        onCreate(db);  
    }  
}
```

Comandos SQL para criar, popular e remover a tabela passwords

Executado quando o BD for criado pela primeira vez

Executado quando o BD for atualizado

# Modificando a Classe PasswordDAO

- A classe PasswordDAO deixará de salvar os dados em um ArrayList e passará a usar o banco de dados
- Atributos e Construtor:

```
public class PasswordDAO {  
    private Context context;  
    private SQLiteDatabase database;  
  
    public PasswordDAO(Context context) {  
        this.context = context;  
        this.database = (new Database(context)).getWritableDatabase();  
    }  
}
```

Acessa o SQLite usando a  
nossa classe auxiliar

# Modificando a Classe PasswordDAO

## ■ Método getList

```
public ArrayList<Password> getList() {  
    ArrayList<Password> result = new ArrayList<Password>();  
    String sql = "SELECT * FROM passwords ORDER BY name";  
    Cursor cursor = database.rawQuery(sql, null);  
  
    while (cursor.moveToNext()) {  
        int id = cursor.getInt(0);  
        String name = cursor.getString(1);  
        String login = cursor.getString(2);  
        String password = cursor.getString(3);  
        String notes = cursor.getString(4);  
        result.add(new Password(id, name, login, password, notes));  
    }  
  
    return result;  
}
```

Retorna um ArrayList,  
mas os dados virão do BD

Executa um comando SQL  
no SQLite do Android

Para cada linha retornada  
da consulta anterior ...

Acessa a primeira coluna  
da linha como um inteiro

Cria um objeto Password  
e insere na lista de retorno

# Modificando a Classe PasswordDAO

## ■ Método add

```
public boolean add>Password password) {  
    String sql = "INSERT INTO passwords VALUES (NULL, "  
        + "'" + password.getName() + "', "  
        + "'" + password.getLogin() + "', "  
        + "'" + password.getPassword() + "', "  
        + "'" + password.getNotes() + "')";  
  
    try {  
        database.execSQL(sql);  
        Toast.makeText(context, "Senha salva!", Toast.LENGTH_SHORT).show();  
        return true;  
    }  
    catch (SQLException e) {  
        Toast.makeText(context, "Erro! " + e.getMessage(), Toast.LENGTH_SHORT).show();  
        return false;  
    }  
}
```

Adiciona um Password no banco de dados

Comando SQL para inserir os dados no BD

Executa o comando SQL

Erro ao executar o comando SQL

# Modificando a Classe PasswordDAO

## ■ Método update

```
public boolean update>Password(password) {  
    String sql = "UPDATE passwords SET "  
        + "name=' " + password.getName() + "' , "  
        + "login=' " + password.getLogin() + "' , "  
        + "password=' " + password.getPassword() + "' , "  
        + "notes=' " + password.getNotes() + "' "  
        + "WHERE id=" + password.getId();  
  
    try {  
        database.execSQL(sql);  
        Toast.makeText(context, "Senha atualizada!", Toast.LENGTH_SHORT).show();  
        return true;  
    }  
    catch (SQLException e) {  
        Toast.makeText(context, "Erro! " + e.getMessage(), Toast.LENGTH_SHORT).show();  
        return false;  
    }  
}
```

Atualiza um Password no banco de dados

Comando SQL para atualizar os dados no BD

Executa o comando SQL

Erro ao executar o comando SQL

# Modificando a Classe PasswordDAO

## ■ Método get

```
public Password get(int id) {  
    String sql = "SELECT * FROM passwords WHERE id=" + id;  
    Cursor cursor = database.rawQuery(sql, null);  
  
    if (cursor.moveToNext()) {  
        String name = cursor.getString(1);  
        String login = cursor.getString(2);  
        String password = cursor.getString(3);  
        String notes = cursor.getString(4);  
        return new Password(id, name, login, password, notes);  
    }  
  
    return null;  
}
```

Se a senha com o ID  
passado não for  
encontrada, retorna nulo

Dado um ID, retorna o  
Password, com este ID

Executa um comando SQL  
no SQLite do Android

Acessa a primeira linha  
retornada da consulta

Acessa a segunda coluna  
da linha como uma string

Cria um objeto Password  
e o retorna

# Resultado

---

- Testando o app com acesso ao banco de dados

# Testando o Banco de Dados

## Resultado

PlainText

GMail  
dovahkiin

+

PlainText

Senha

	Coursera
	sovngarde
	HunKaalZoor
	Notas

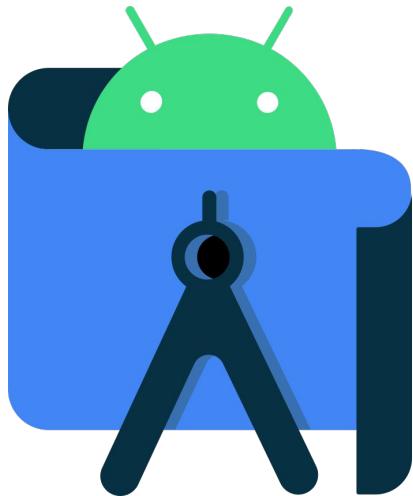
SALVAR

PlainText

	Coursera sovngarde	>
	GMail dovahkiin	>

+

Universidade Federal do Amazonas  
Instituto de Computação  
Projeto de Programas  
Técnicas Avançadas de Programação



# Conclusões

ColabWeb: [bit.ly/pp-colabweb](https://bit.ly/pp-colabweb)

 Horácio Fernandes  
[horacio@icomp.ufam.edu.br](mailto:horacio@icomp.ufam.edu.br)

# Tópicos Avançados

- Web Services
  - Acessar dados de servidores na Internet
- Fragmentos
  - Criar aplicativos em uma só tela com “fragmentos” dos lados
- Serviços Baseados em Localização
  - GPS, Mapas
- Sensores
  - Acelerômetro, Giroscópio, Luminosidade, Magnetômetro
- Multimídia
  - Câmera, Vídeo, Voz
- Telefonia
  - Contatos, SMS

# Referências

---

- ❑ Training for Android Developers
  - ❑ <http://developer.android.com/training/>
- ❑ Android API Guide
  - ❑ <http://developer.android.com/guide/components/>
- ❑ Android API Reference
  - ❑ <http://developer.android.com/reference/>