



Desenvolvimento Mobile

Introdução Android

Professor MSc. Antônio Catani
antonio.catani@faculdadeimpacta.com.br

Sumário

- Competências
- Habilidades
- Bibliografia
- Conteúdo Programático
- Critérios de Avaliação

Android Studio

Android Studio

- Android Studio é o ambiente de desenvolvimento de aplicativos para Android do Google.
 - Utiliza linguagem Java ou Kotlin.
 - Contém a IDE e o Android SDK.
- Download
 - <https://developer.android.com/studio/index.html>

LMS App

LMSApp

- Durante o curso vamos exercitar a programação para Android desenvolvendo partes do aplicativo para o LMS.
- Os códigos de exemplo feitos em aulas estão no GitHub:
 - <https://github.com/fesousa/aula-android-kotlin-2022.git>

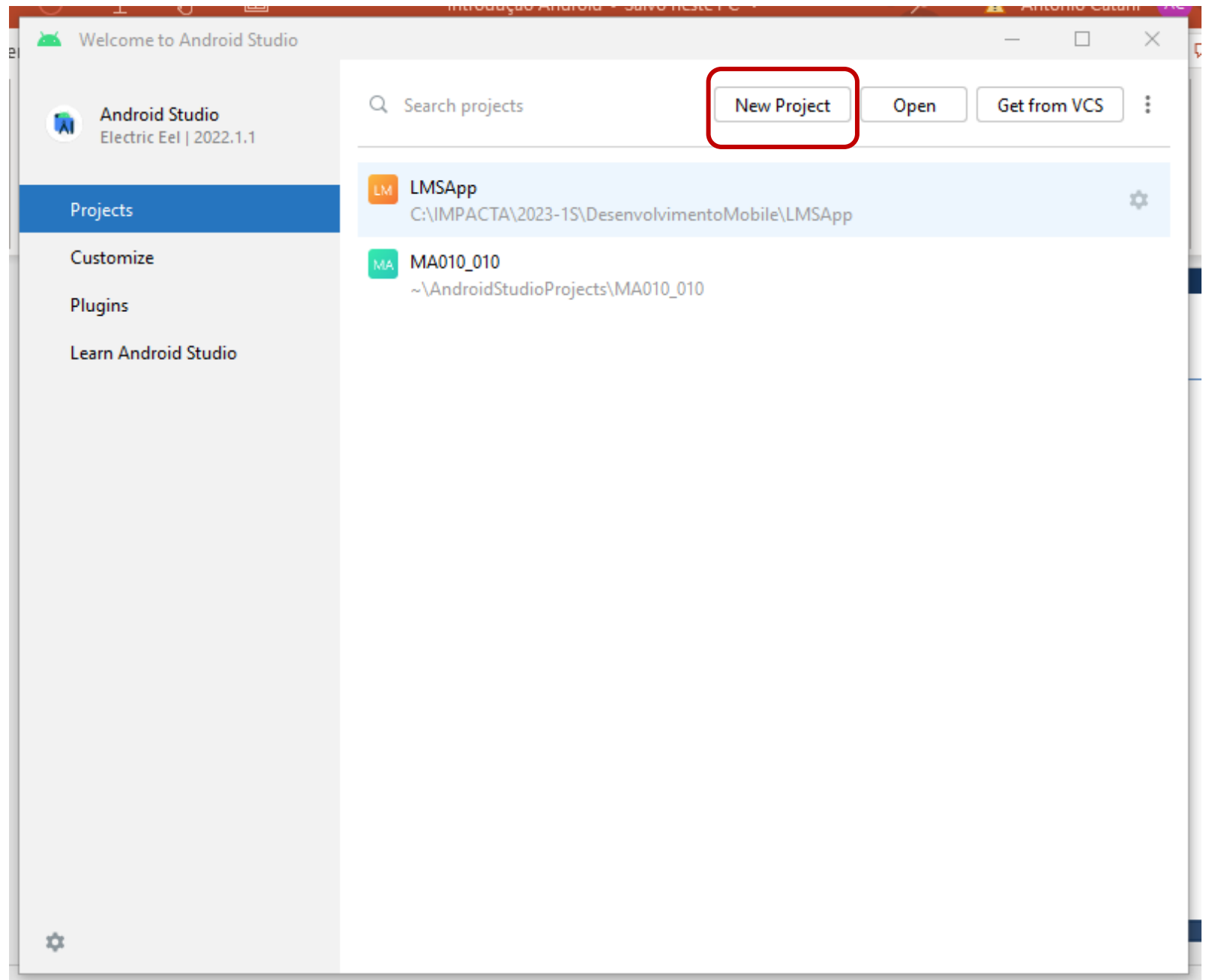
LMSApp

- Cada aula em uma branch diferente.
 - Final desta aula: LMSApp_01_Intro.
- Caso tenha perdido o conteúdo da aula anterior, baixe o código produzido no final da aula que está na Branch.
 - O Link estará no final de cada aula e no começo da aula seguinte.

Criando o Aplicativo

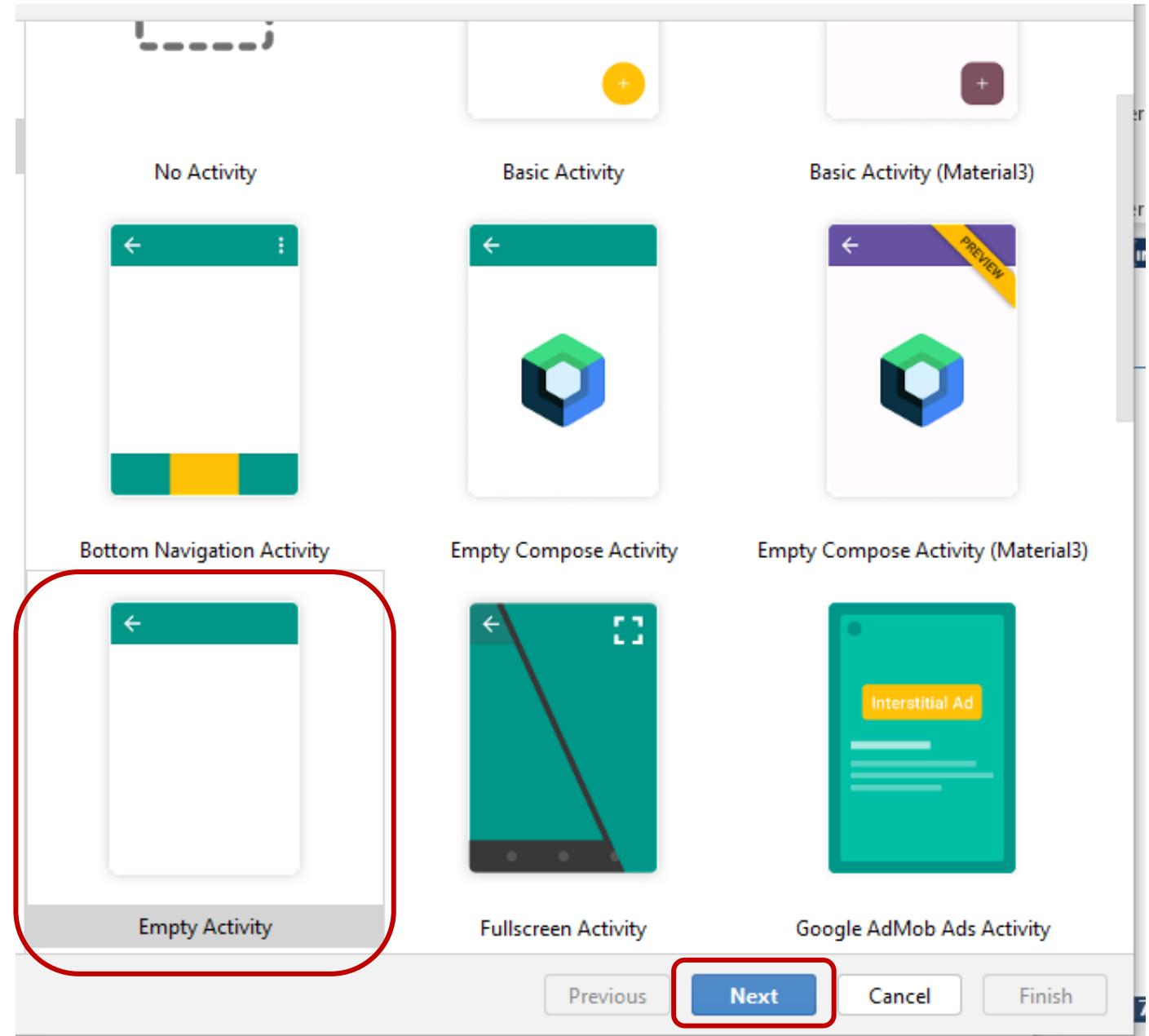
Criando o Aplicativo

- Abra Android Studio
- Selecionar a opção **New Project**.



Criando o Aplicativo

- Na próxima tela,
- escolha **Empty Activity**.
- Clicar em **Next**.



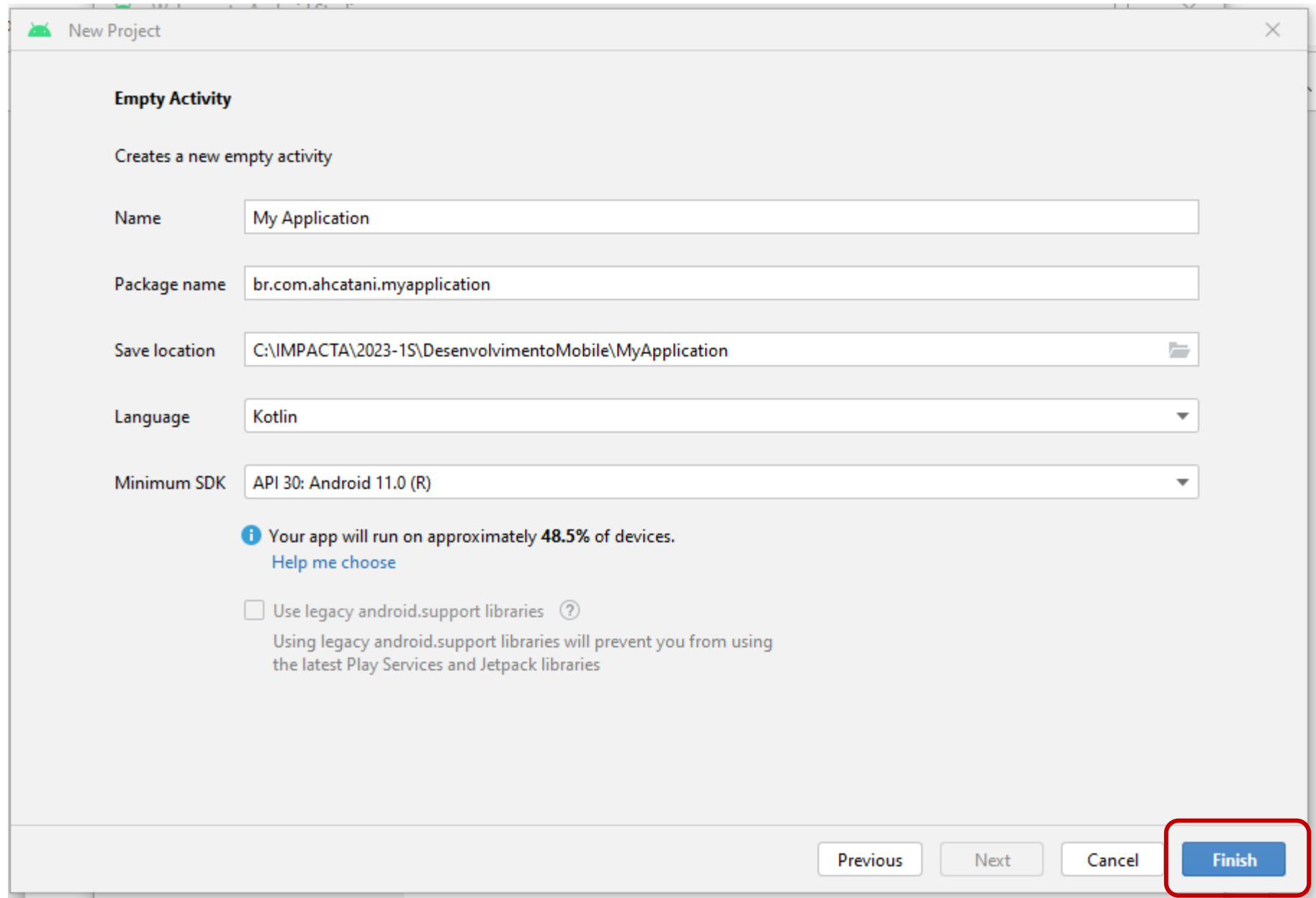
Criando o Aplicativo

- Na próxima tela, definir:
 - Nome da aplicação (**Name**) - **LMSAppN** (**N** identificação da turma).
 - Nome do pacote (**Package Name**).
 - O nome do pacote precisa ser único na Google Play, por exemplo o domínio da empresa, ao contrário, é uma boa alternativa.
 - Local onde salvar o projeto (**Save Location**).

Criando o Aplicativo

- Na próxima tela, definir (...):
 - Linguagem (**Language**): Kotlin.
 - Nível da API (**Minimum SDK**): 30.
 - Quanto menor a versão, em mais dispositivos o aplicativo funcionará, mas menos recursos estarão disponíveis.
- Clicar em **Finish**.

Criando o Aplicativo



New Project

Empty Activity

Creates a new empty activity

Name: My Application

Package name: br.com.ahcatani.myapplication

Save location: C:\IMPACTA\2023-1S\DesenvolvimentoMobile\MyApplication

Language: Kotlin

Minimum SDK: API 30: Android 11.0 (R)

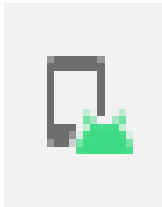
i Your app will run on approximately **48.5%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries **?**
 Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

Previous Next Cancel **Finish**

Criando um Emulador

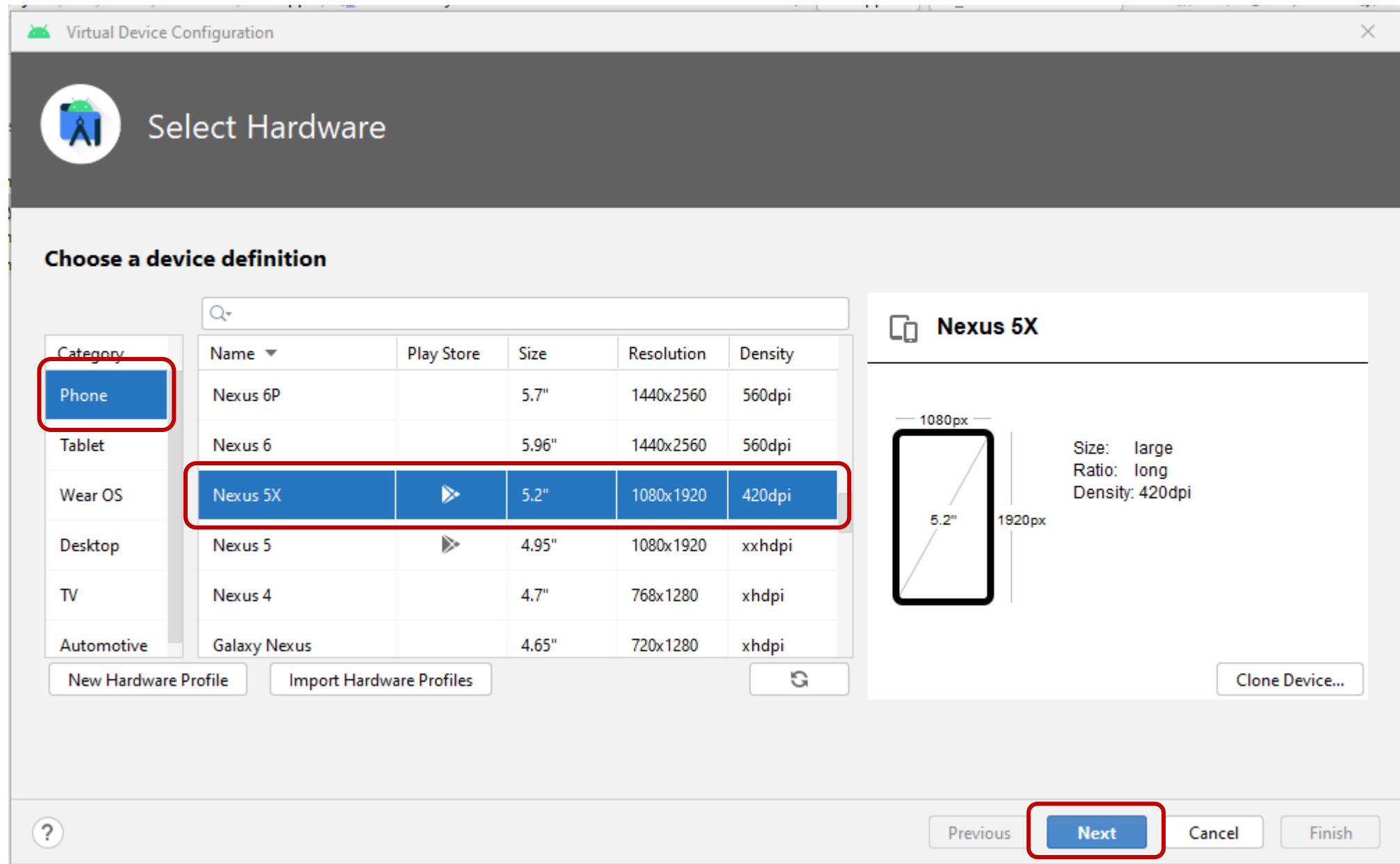
Criando um Emulador

- O Emulador Android é um dispositivo virtual que simula a configuração real de um celular.
 - É possível criar emuladores com diversas configurações diferentes, incluindo versão da API Android.
- Clicar no ícone do Device Manager () para criar um novo emulador.
- Na tela do Device Manager, selecionar a opção **Virtual** e clicar em **Create device**.

Criando um Emulador

- A primeira tela mostra as opções de hardware,
- Existem opções pré-configuradas para diversos tipos de smartphones, tablets, wearables e TV.
- Para a aula, será utilizada a opção **Nexus 5X**, dentro da categoria **Phone**.
- Clicar em **Next**.

Selecionar o Hardware




Criando um Emulador

- Na próxima tela selecionar a imagem do emulador, (**System Image**) com a versão do Android.
 - As opções mostradas serão aquelas já instaladas na máquina de desenvolvimento.
 - Lembrar sempre de escolher uma versão igual ou superior ao da API escolhida para o aplicativo.
- Clicar em **Next**.

Imagem do Sistema


Virtual Device Configuration

 System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API Level ▼	ABI	Target
<i>UpsideDownCake</i> ⬇	<i>UpsideDownCake</i>	x86_64	Android API UpsideDow
<i>TiramisuPrivac...</i> ⬇	<i>TiramisuPrivacySan</i>	x86_64	Android API TiramisuPn
<i>Tiramisu</i> ⬇	33	x86_64	Android 13.0 (Google Pl
<i>Sv2</i> ⬇	32	x86_64	Android 12L (Google Pla
<i>S</i> ⬇	31	x86_64	Android 12.0 (Google Pl
R	30	x86	Android 11.0 (Google P
<i>Q</i> ⬇	29	x86	Android 10.0 (Google Pl



R

API Level
30

Android
11.0

Google Inc.

System Image
x86


We recommend these Google Play images because this device is compatible with Google Play.


Previous **Next** Cancel Finish

Criando um Emulador

- Depois, basta verificar e alterar alguma configuração necessária.
 - Para o hardware escolhido, pode-se alterar o nome do AVD (Android Virtual Device).
- Clicar em **Finish**.



Verificação da Configuração

 Virtual Device Configuration


 Android Virtual Device (AVD)

Verify Configuration


AVD Name

	Nexus 5X	5.2 1080x1920 420dpi	<button>Change...</button>
	R	Android 11.0 x86	<button>Change...</button>

Startup orientation



Portrait



Landscape

Show Advanced Settings

AVD Name

The name of this AVD.

?

Previous

Next

Cancel

Finish

Executando o Aplicativo

Criando um Emulador

- Podem ser criados quantos emuladores forem necessários
 - Testar o aplicativo em diferentes versões de API, configurações de hardware e tamanhos de tela.
- Na tela principal da IDE executar o aplicativo:
 - Selecionar o emulador.
 - Clicar no **Play** da barra superior.

Executando o Aplicativo no SmartPhone

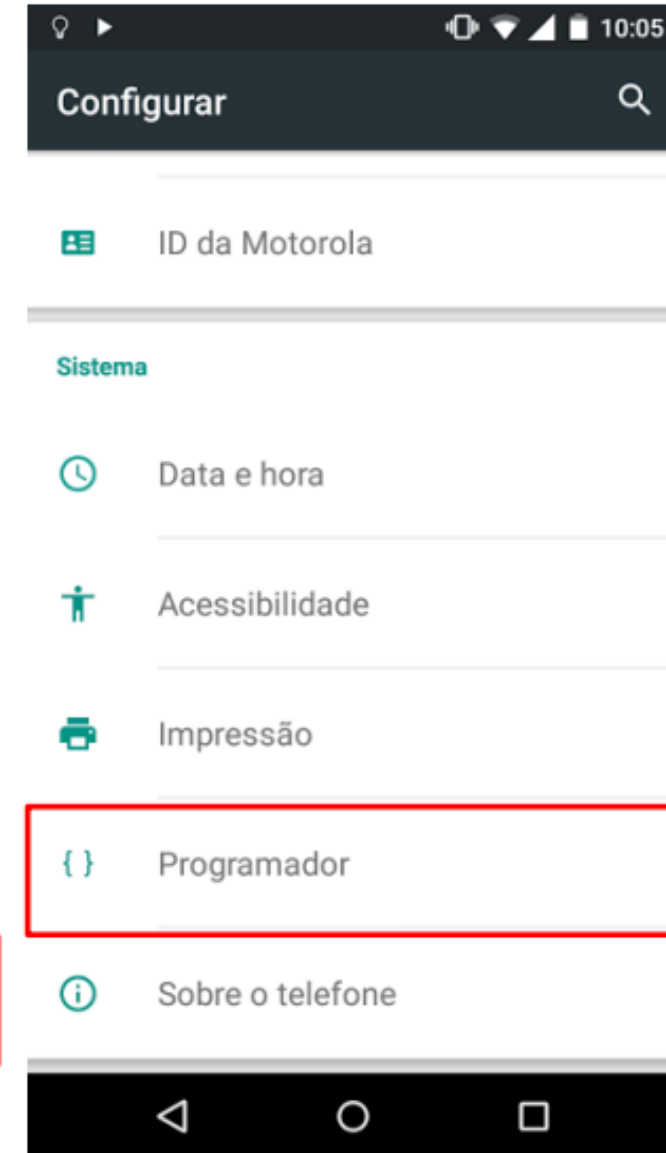
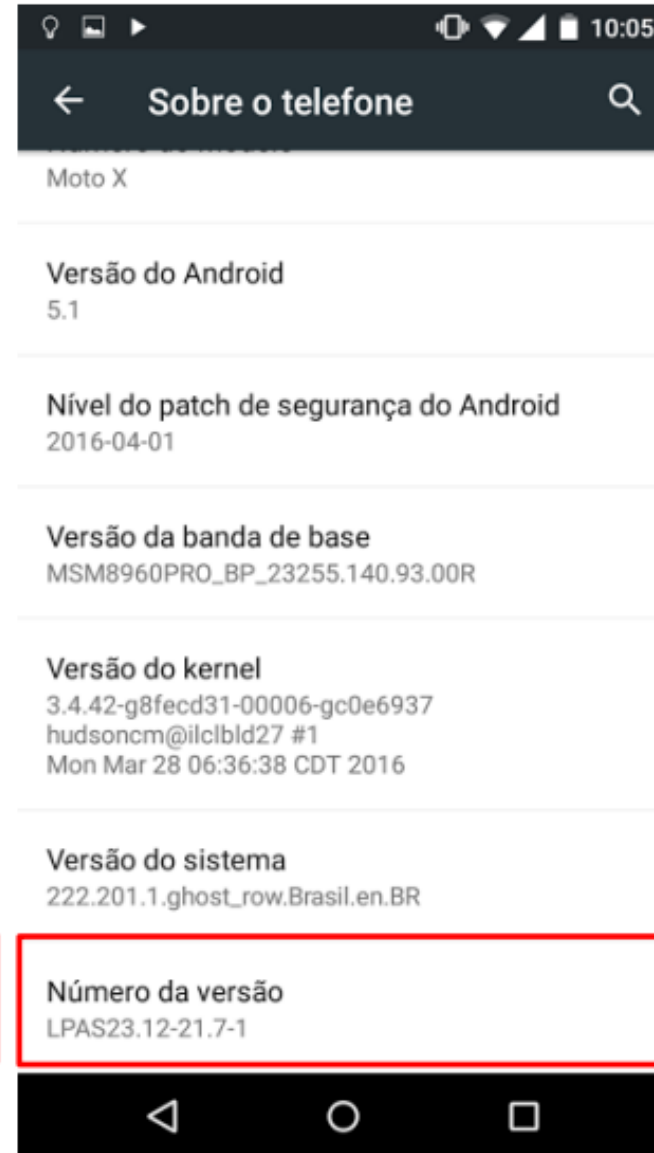
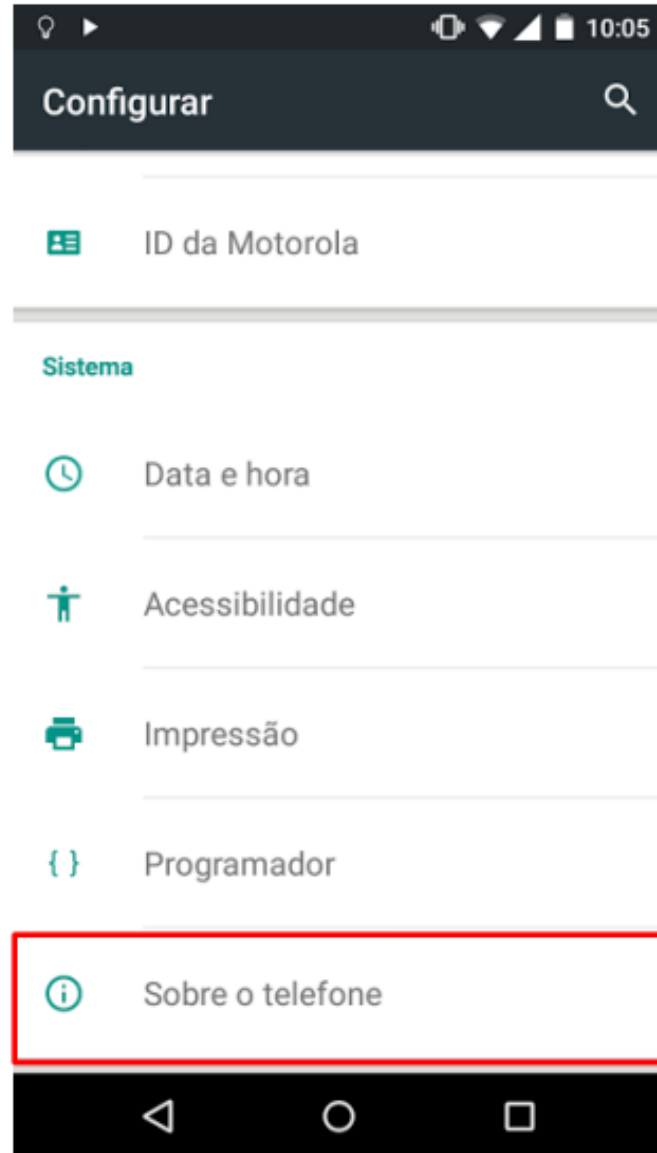
Executando o Aplicativo no SmartPhone

- Pode-se também executar o aplicativo diretamente de um smartphone Android.
- A configuração depende do fabricante do SmartPhone.
- Para executar diretamente no SmartPhone, é necessário habilitá-lo seu para desenvolvimento.

Executando o Aplicativo no SmartPhone

- Abrir as configuração do telefone.
- Vá até a opção **Sobre o dispositivo** (último item).
- Clique 7 (ou 5) vezes no item **Número da versão**.
- Volte às configurações, e haverá uma nova opção, **Programador**.
- Abrir esta opção.

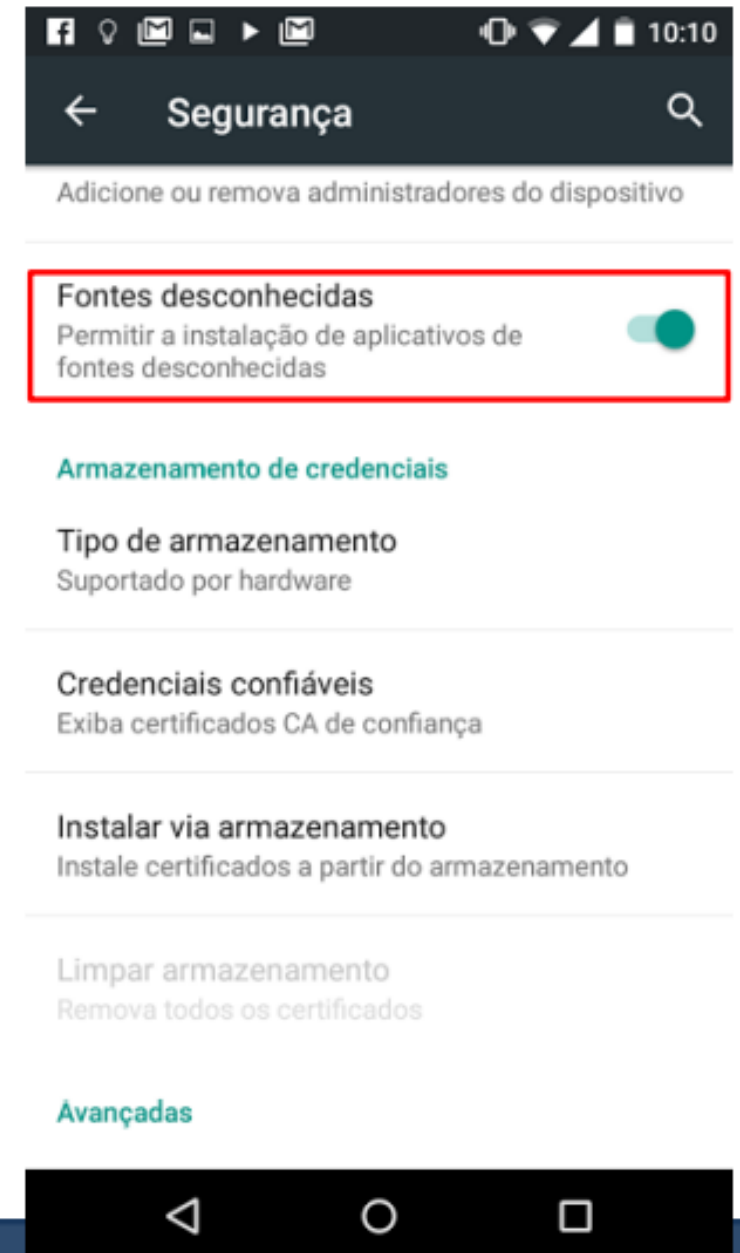
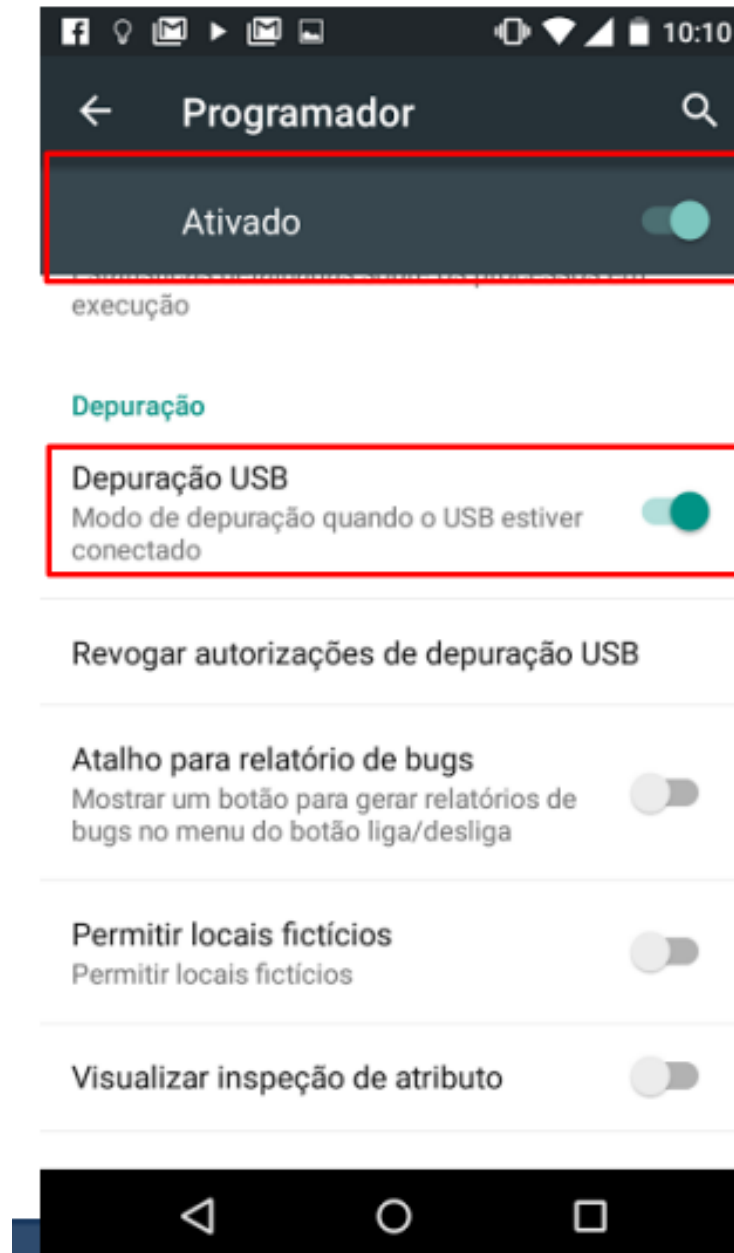
Executando o Aplicativo no SmartPhone



Executando o Aplicativo no SmartPhone

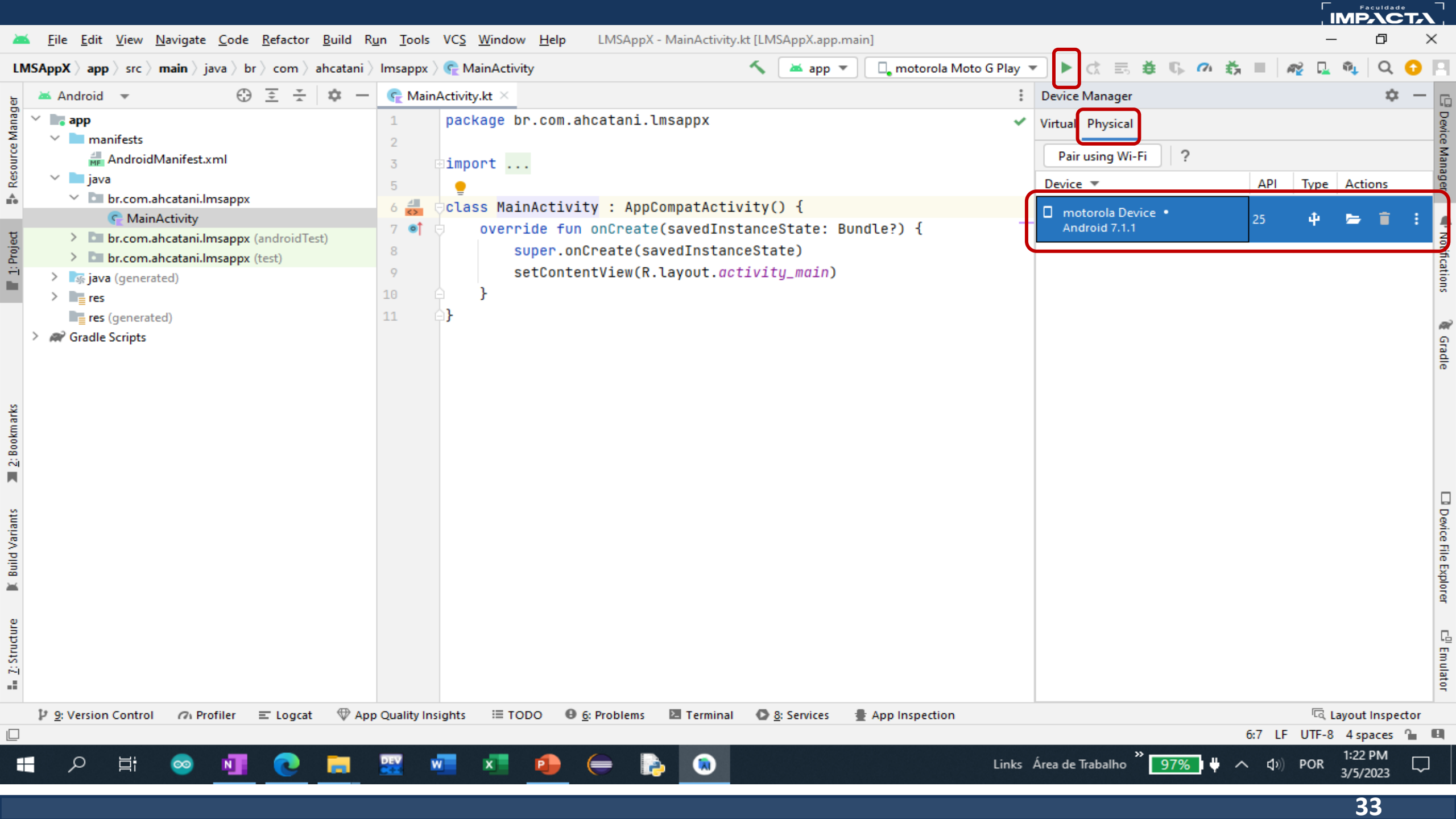
- Na opção programador:
 - Ativar opção **depuração USB**.
- Volte às configurações e selecione a opção **Segurança**.
 - Habilitar a opção **Fontes desconhecidas**.

Executando o Aplicativo no SmartPhone



Executando o Aplicativo no SmartPhone

- Agora é preciso conectar o smartPhone no computador via USB com um cabo de dados.
- Depois disso, o SmartPhone vai aparecer na lista de dispositivos quando for executar o aplicativo pelo Android Studio.
- Na tela principal da IDE executar o aplicativo:
 - Selecionar o dispositivo
 - Clicar no **Play** da barra superior.



Estrutura do Projeto

Estrutura do Projeto

- Ao criar o projeto no Android Studio, existe uma região chamada **Project**, com a aba **Android** habilitada, do lado esquerdo da IDE.
 - Esta aba mostra os arquivos essenciais do seu projeto.
 - Aqueles que podem ser alterados dentro do seu projeto.
- Existem basicamente 3 pastas dentro da pasta **app**, contendo o código fonte:
 - **manifest**
 - **java**
 - **res**

Pasta manifest

Pasta manifest

- Contém apenas um arquivo:
 - **AndroidManifest.xml**.
- **AndroidManifest.xml** é o arquivo principal do projeto que centraliza as configurações do aplicativo.
 - Indica a **tela** e a **Activity** inicial do projeto.
 - Indicar quais as **Activities** existentes.
 - **Nome** e **imagem** do aplicativo.
 - **Nome do pacote** do projeto.

Pasta manifest

- Observações:
 - Todas as Activities do projeto devem ser declaradas no **AndroidManifest.xml**.
 - Apenas uma Activity deve ser configurada como principal (**MAIN**) e com o ícone do aplicativo (**LAUNCHER**).
 - As outras Activities podem ter outros tipos de filtros, mas inicialmente elas devem ser declaradas apenas com o nome.

<activity android:name=".NomeActivity" />

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSAppX - AndroidManifest.xml [LMSAppX.app.main]

LMSAppX > app > src > main > MF AndroidManifest.xml

AndroidManifest.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools">
4
5   <application
6     android:allowBackup="true"
7     android:dataExtractionRules="@xml/data_extraction_rules"
8     android:fullBackupContent="@xml/backup_rules"
9     android:icon="@mipmap/ic_launcher"
10    android:label="LMSAppX"
11    android:supportsRtl="true"
12    android:theme="@style/Theme.LMSAppX"
13    tools:targetApi="31">
14     <activity
15       android:name=".MainActivity"
16       android:exported="true">
17       <intent-filter>
18         <action android:name="android.intent.action.MAIN" />
19
20         <category android:name="android.intent.category.LAUNCHER" />
21       </intent-filter>
22     </activity>
23   </application>
24
25 </manifest>

```

manifest > application

Text Merged Manifest

9: Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

11:35 CRLF UTF-8 4 spaces 100% POR 2:31 PM 3/5/2023

Pasta java

Pasta java

- Na pasta java estarão todos os módulos Kotlin ou Java do seu projeto.
- Pode-se criar outros pacotes para organizar os arquivos, mas é aconselhável manter todos dentro do pacote principal (configurado na criação do app e declarado no **AndroidManifest.xml**).
- Quando o projeto é criado uma Activity é criada.
 - O padrão é **MainActivity**, mas pode ter qualquer nome.

Pasta java - MainActivity

Pasta java - MainActivity

- O importante é que esta classe, e qualquer outra Activity do projeto, seja filha da classe **android.app.Activity**.
 - No exemplo, está herdando **AppCompatActivity**.
 - **AppCompatActivity** é filha de **android.app.Activity**.
- **MainActivity** está configurada para ser a Activity inicial e ser iniciada quando clicar no ícone do aplicativo.
 - Lembre-se da configuração em **AndroidManifest.xml**.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSAppX - MainActivity.kt [LMSAppX.app.main]

LMSAppX > app

Android

app

manifests

AndroidManifest.xml

java

br.com.ahcatani.lmsappx

MainActivity

br.com.ahcatani.lmsappx (androidTest)

br.com.ahcatani.lmsappx (test)

java (generated)

res

res (generated)

Gradle Scripts

MainActivity.kt

AndroidManifest.xml

```

1 package br.com.ahcatani.lmsappx
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }

```

Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

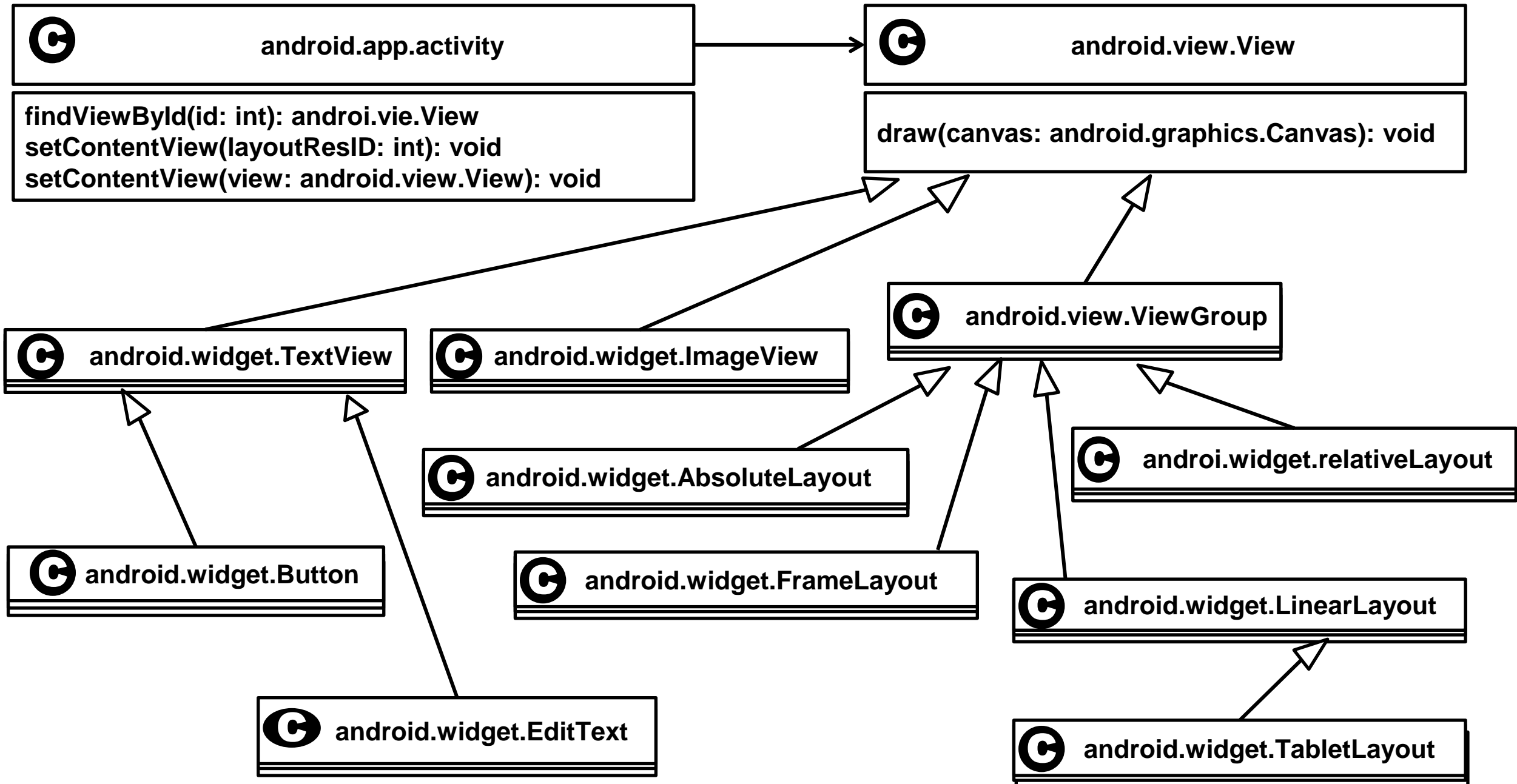
11:2 LF UTF-8 4 spaces 100% 4:25 PM 3/5/2023

Pasta java - MainActivity

- Uma Activity representa uma tela do aplicativo:
 - Controla o estado.
 - Controla os eventos de tela.
 - Cada tela do aplicativo terá uma Activity relacionada.
- Para cada Activity criada, é obrigatório implementar o método **onCreate(bundle)**.
 - Este método é chamado automaticamente quando a tela é criada.
 - **android.app.Activity** contém outros métodos que podem ser sobrecarregados, mas não são obrigatórios.

Pasta java - MainActivity

- Apesar de representar a tela, uma Activity não sabe “desenhar” os elementos da tela.
 - Precisa de uma classe do tipo **android.view.View**.
 - Uma **View** é responsável por desenhar os elementos da tela.
 - Textos, campos, botões, imagens, layouts... .
 - Cada elemento da tela é uma subclasse de **android.view.View**.



Pasta java - MainActivity

- Para adicionar um elemento na tela relacionada à Activity, utilize o método **setContentView**.
 - Método já implementado na classe mãe **Activity**.
 - Responsável por fazer a ligação entre a **Activity** e a **View** que desenha a interface gráfica.
 - Deve ser chamado sempre no método **onCreate** da **Activity**.

Pasta java - MainActivity

- O **setContentView** pode receber como argumento.
 - Uma instância de **View**: para construir telas diretamente pelo código Kotlin ou Java (mais dinâmico).
 - Um número inteiro que representa um recurso ou um arquivo **xml** de **layout** (menos dinâmico).

Pasta java - MainActivity

- No exemplo, o método **setContentView** recebe como argumento **R.layout.activity_main**.
 - Constante inteira.
 - Representa um recurso do aplicativo.
 - Todos os recursos estarão dentro da pasta **res** do projeto.
- Neste caso, **R.layout.activity_main** representa o arquivo de **layout activity_main.xml**.

Pasta res

Pasta res

- A pasta **res** contém os recursos do aplicativo.
- Recurso é qualquer arquivo utilizado no aplicativo que não seja o arquivo de configuração (AndroidManifest.xml) e classes kotlin ou java:
 - **Imagens** - subpasta **drawable**.
 - **Layouts de telas** - subpasta **layout**.
 - **Strings/textos** - subpasta values, arquivo **string.xml**.
 - **Estilos** – subpasta values, arquivo **styles.xml**.
- Todos os recursos são associados a uma constante inteira presente na classe **R**.

Utilizando os Recursos da Classe R

Utilizando os Recursos da Classe R

- Qualquer recurso definido na classe **R** pode ser utilizado, tanto em outros recursos como nas classes Kotlin / Java.
- Dentro de arquivos XML, o padrão de uso dos recursos é sempre o mesmo.

@tipoRecurso/nome_recurso

Utilizando os Recursos da Classe R

- Para imagens e arquivos de layout, o nome do recurso é o nome do arquivo sem a extensão.
- O tipo do recurso é o nome da pasta (**layout**, **drawable**).
- Para os outros, o nome do recurso está dentro do arquivo correspondente (**strings.xml**, **styles.xml**).
- O nome do arquivo representa o tipo de recurso.

Utilizando os Recursos da Classe R

- Exemplos:

@layout/activity_main

- Tipo de recurso: **layout**.
- Nome do recurso: **activity_main**.
- Arquivo de **layout** com nome **activity_main.xml** dentro da pasta **res/layout**.

Utilizando os Recursos da Classe R

- Exemplos:

@string/mensagem

- Tipo de recurso: **string**.
- Nome do recurso: **mensagem**.
- **Texto** com o identificador **mensagem** dentro do arquivo **strings.xml** na pasta **res/**.

Utilizando os Recursos da Classe R

- Nas classes Kotlin/Java, também existe um padrão de uso:
R.tipoRecurso.nome_recurso
 - Para imagens e arquivos de layout, o nome do recurso é o nome do arquivo sem a extensão
 - O tipo do recurso é o nome da pasta (**layout**, **drawable**).
 - Para os outros, o nome do recurso está dentro do arquivo correspondente (**strings.xml**, **styles.xml**).
 - O nome do arquivo representa o tipo de recurso.

Utilizando os Recursos da Classe R

- Exemplos:

R.layout.activity_main

- Tipo de recurso: **layout**.
- Nome do recurso: **activity_main**.
- Arquivo de **layout** com nome **activity_main.xml** dentro da pasta **res/layout**.

Utilizando os Recursos da Classe R

- Exemplos:

R.string.mensagem

- Tipo de recurso: **string**.
- Nome do recurso: **mensagem**.
- **Texto** com o identificador **mensagem** dentro do arquivo **strings.xml** na pasta **res/**.

Pasta res – layout activity_main.xml

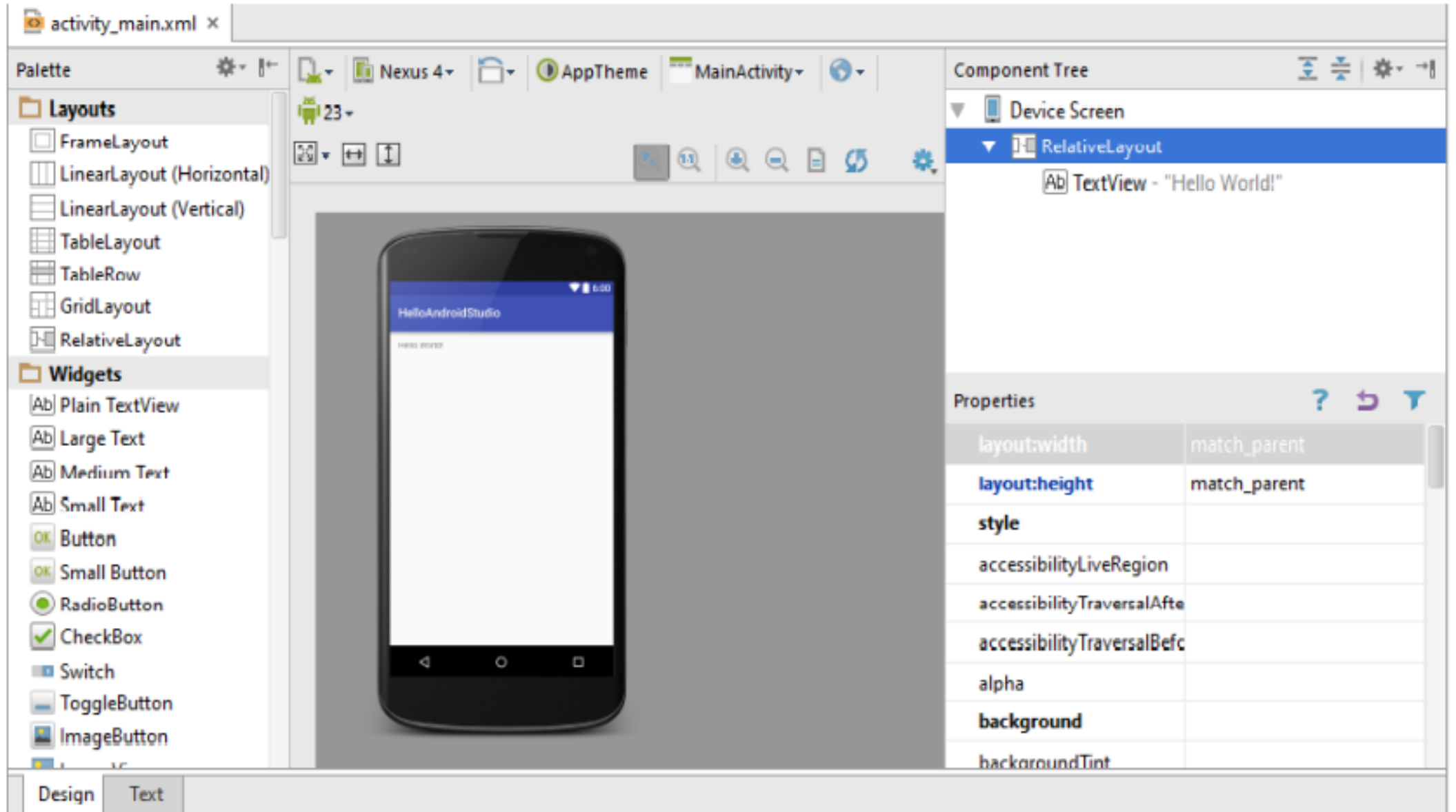
Pasta res – layout activity_main.xml

- Layouts de tela podem ser criados utilizando arquivos **XML** ou utilizar **Kotlin/Java**.
 - Recomendado: **XML**.
 - Separar lógica de negócio da apresentação.
 - Activity: **Controller**; Arquivo XML de layout: **View**.
- Todos os layouts devem ficar em **res/layout**.
- Quando o projeto é criado o arquivo **activity_main.xml** é criado, para comportar a tela inicial do aplicativo.

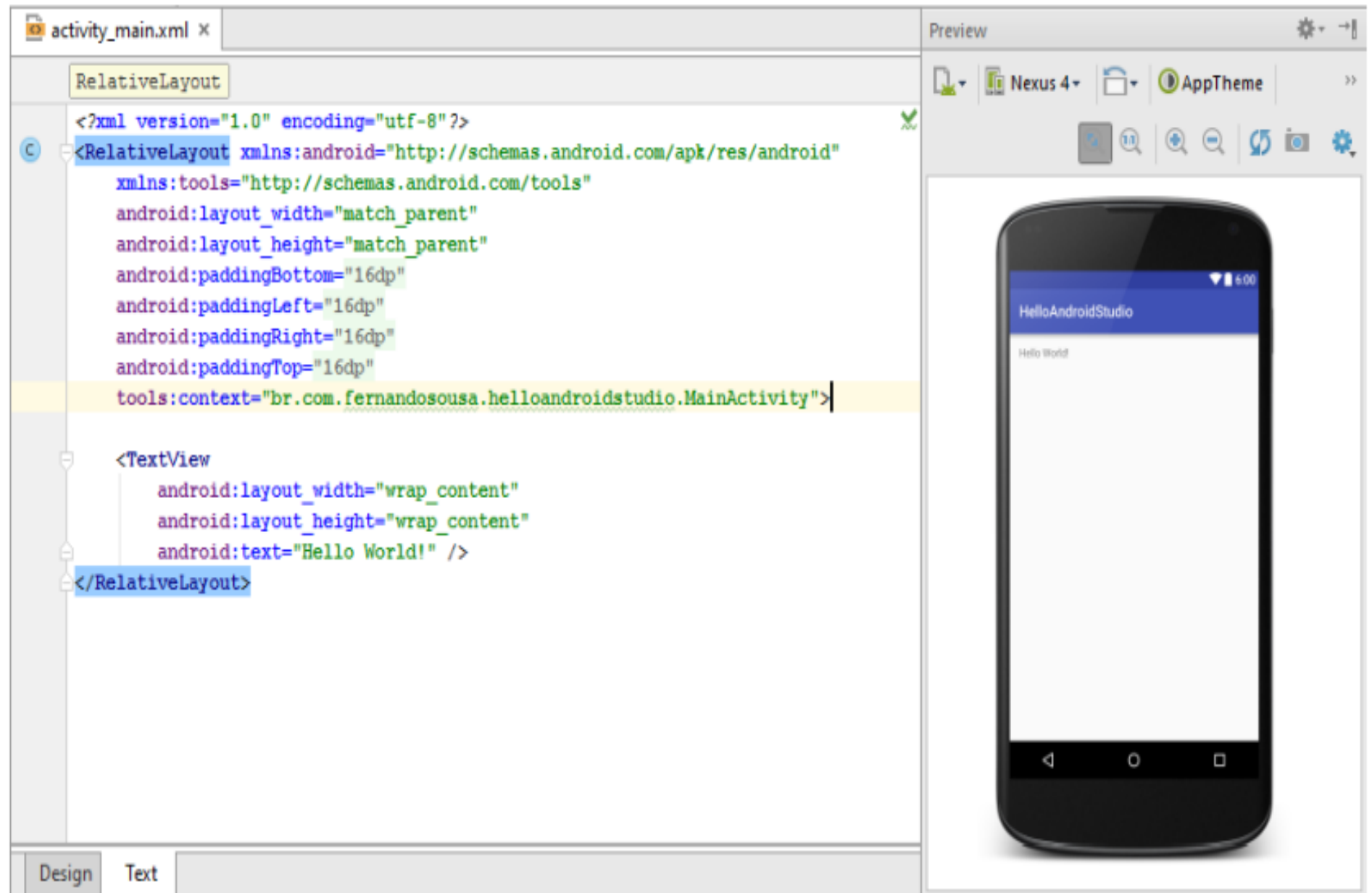
Pasta res – layout activity_main.xml

- Todos os elementos de tela são filhos de **android.view.View**.
- No arquivo XML de layout, os marcadores representam a subclasse de View, e os atributos dos marcadores são os atributos da subclasse.
- A construção da tela pode ser feita escrevendo os marcadores diretamente do código fonte ou utilizando o editor visual.

Editor Visual



Editor Visual



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSAppX - activity_main.xml [LMSAppX.app.main]

LMSAppX > app > src > main > res > layout > activity_main.xml

Android MainActivity.kt activity_main.xml AndroidManifest.xml

app

- manifests
 - AndroidManifest.xml
- java
 - br.com.ahcatani.lmsappx
 - MainActivity
 - br.com.ahcatani.lmsappx (androidTest)
 - br.com.ahcatani.lmsappx (test)
 - java (generated)
- res
 - drawable
 - layout
 - activity_main.xml
 - mipmap
 - values
 - xml
 - res (generated)
- Gradle Scripts

Palette

- Common
 - View
 - ImageView
 - WebView
 - VideoView
- Text
- Buttons
- Widgets
 - Calendar...
- Layouts
 - Progress...
 - Progress...
- Containers
 - SeekBar
 - SeekBar (...)
 - RatingBar
 - SearchView
 - TextureVi...
- Helpers
- Google
- Legacy

Component Tree

- ConstraintLayout
 - Ab TextView "Hello Worl...

activity_main.xml

Pixel 33 LMSAppX

0dp

View

Attributes

ConstraintLayout <unnamed>

id

Declared Attributes

Layout

layout_width itch_parent

layout_hei... itch_parent

visibility

visibility

Transforms

View

Rotation

x 0

y 0

z 0

Version Control Profiler Logcat App Quality Insights TODO 6 Problems Terminal 8 Services App Inspection Layout Inspector

Editor
Visual

Pasta res – layout activity_main.xml

- O arquivo de layout criado será utilizado para definir a View representada pela Activity.
- O acesso na Activity é feito por uma constante inteira, como argumento do método **setContentView()**.

R.layout.activity_main

- Para acessar qualquer arquivo de layout dentro de uma classe kotlin/java, utiliza-se o padrão:

R.layout.nome_arquivo

Pasta res – novo layout

Pasta res – novo layout

- Exercício:
 - Crie um novo arquivo de Layout chamado **activity_login.xml**.
 - Utilize o menu de contexto: botão direito na pasta **res/layout → new → Layout Resource File**.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - AndroidManifest.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout

Android activity_main.xml MainActivity.kt AndroidManifest.xml

Device Manager

Virtual Physical

Create device

Device	API	Size on disk	Actions
Nexus 5X Android 11.0 Google Play x86	30	10 GB	▶ 📁 ✎ ⋮

1: Project 2: Bookmarks 3: Structure 4: Build Variants 5: Version Control 6: Run and Debug 7: Logcat 8: Services 9: App Inspection 10: Layout Inspector

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     tools="http://schemas.android.com/tools">
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

New

- Add C++ to Module
- Cut Ctrl+X
- Copy Ctrl+C
- Copy Path/Reference...
- Paste Ctrl+V
- Find Usages Alt+F7
- Find in Files... Ctrl+Shift+F
- Replace in Files... Ctrl+Shift+R
- Analyze
- Refactor
- Bookmarks
- Show In Resource Manager Ctrl+Shift+T
- Reformat Code Ctrl+Alt+L
- Optimize Imports Ctrl+Alt+O
- Delete... Exclur
- Open In
- Local History
- Repair IDE
- Reload from Disk
- Compare With... Ctrl+D
- Mark Directory as
- Convert Java File to Kotlin File Ctrl+Alt+Shift+K

- Kotlin Class/File
- Layout Resource File
- Sample Data Directory
- File
- Scratch File Ctrl+Alt+Shift+Insert
- Directory
- Image Asset
- Vector Asset
- Kotlin Script
- Kotlin Worksheet
- CMakeLists.txt
- Activity
- Fragment
- Folder
- Service
- UiComponent
- Automotive
- XML
- Wear
- AIDL
- Widget
- Google
- Compose
- Other
- Resource Bundle
- EditorConfig File

terminal 8: Services App Inspection

16:37 CRLF UTF-8 4 spaces 100% 3:24 PM 3/6/2023

Pasta res – layout activity_main.xml

- Exercício(...):
 - Dê um nome para o arquivo.
 - Em **Root** elemento digitar **LinearLayout**.
 - Clicar e **OK**.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - AndroidManifest.xml [LMSApp.app.main]

LMSApp > app > src > main > AndroidManifest.xml

activity_main.xml x MainActivity.kt x AndroidManifest.xml x

Android > manifests > java > java (generated) > res > drawable > layout > activity_main.xml > mipmap > values > xml > res (generated) > Gradle Scripts

1: Project 2: Bookmarks Build Variants 3: Structure

Device Manager Gradle Assistant Notifications Device File Explorer Emulator

1 <?xml version="1.0" encoding="utf-8"?>

2 <manifest xmlns:android="http://schemas.android.com/apk/res/android">

24

25 </manifest>

manifest > application > activity

Text Merged Manifest

9: Version Control 10: Profiler 11: Logcat 12: App Quality Insights 13: TODO 14: Problems 15: Terminal 16: Services 17: App Inspection

16:37 CRLF UTF-8 4 spaces 100% 4:11 PM 3/6/2023

Links Área de Trabalho

76

New Resource File

File name: activity_login

Root element: LinearLayout

Source set: main src/main/res

Directory name: layout

Available qualifiers:

- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation
- UI Mode
- Night Mode

Chosen qualifiers:

Nothing to show

OK Cancel

Pasta res – layout activity_main.xml

- Exercício(...):
 - Coloque neste arquivo um **TextView**, utilizando a paleta de opções do lado esquerdo.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - activity_login.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout > activity_login.xml

Android activity_main.xml MainActivity.kt AndroidManifest.xml activity_login.xml

Code Split Design

Palette

- Common
- Text**
 - Ab TextView**
 - Ab Plain Text
 - Ab Password
 - Ab Password (N...
 - Ab E-mail
 - Ab Phone
 - Ab Postal Address
 - Ab Multiline Text
 - Ab Time
 - Ab Date
 - Ab Number
 - Ab Number (Sig...
- Buttons
- Widgets
- Layouts
- Containers
- Helpers
- Google
- Legacy

Component Tree

- LinearLayout (vertical)
 - Ab textView "TextView"

LinearLayout

13:00

LMSApp

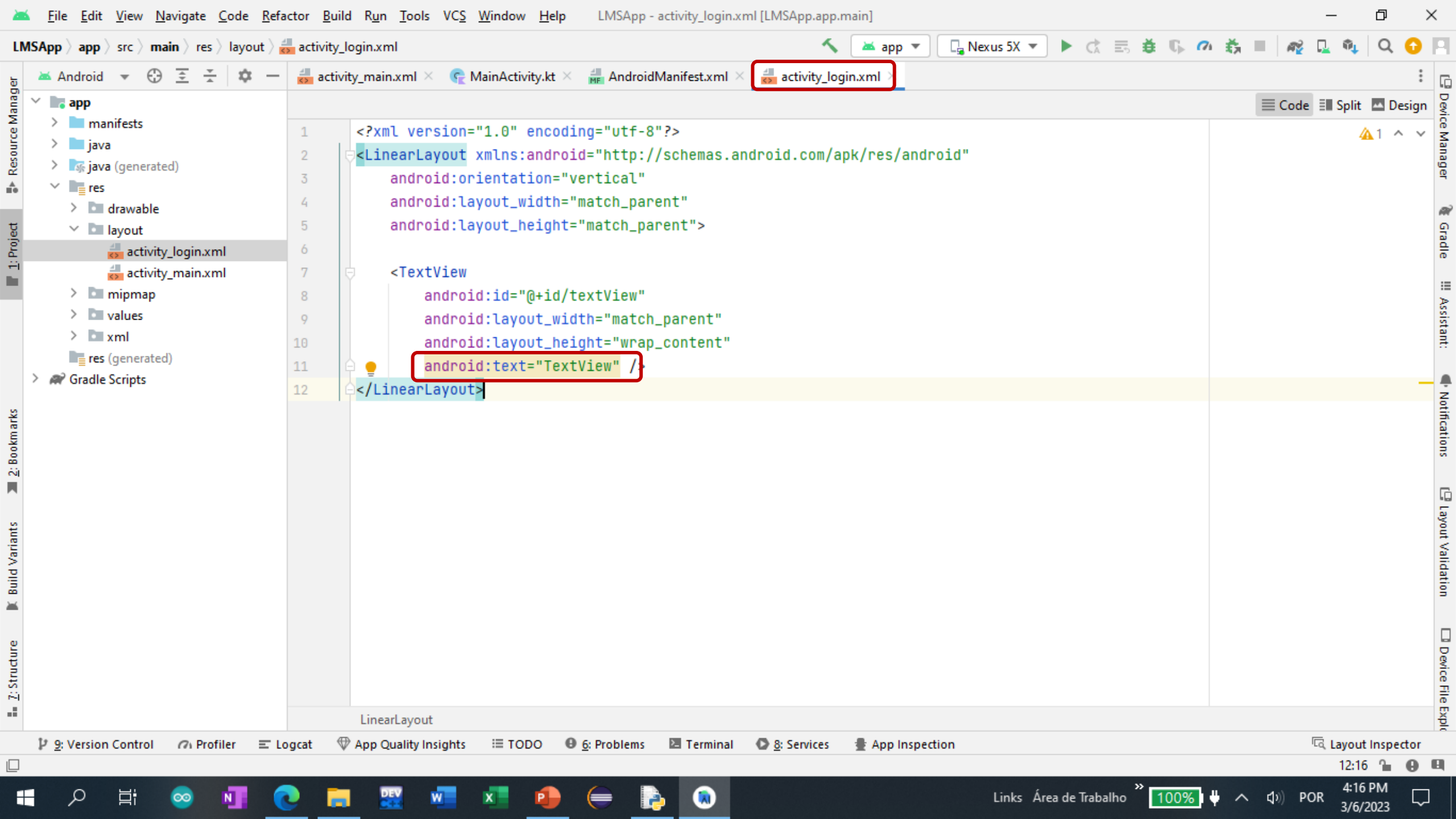
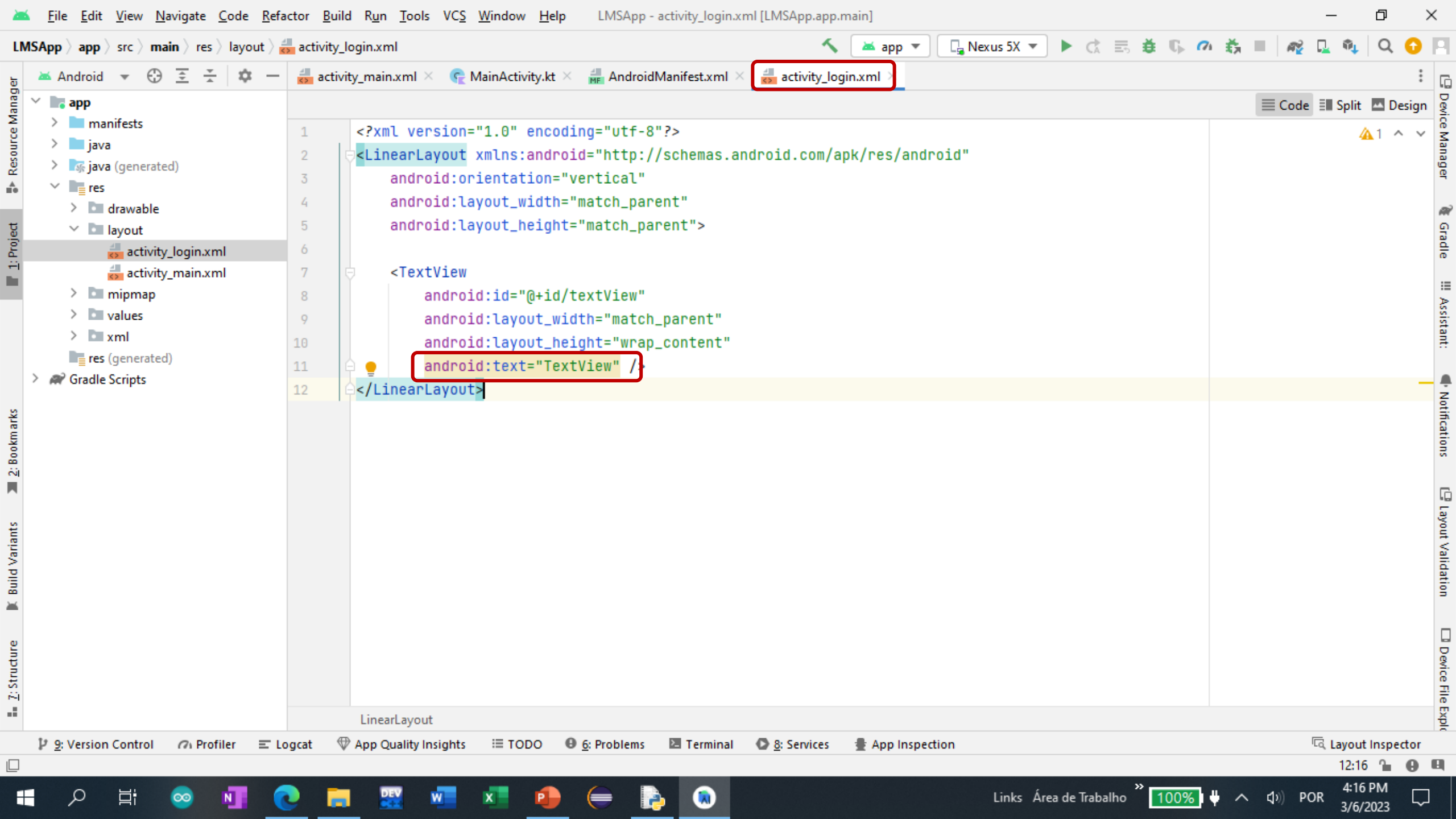
TextView

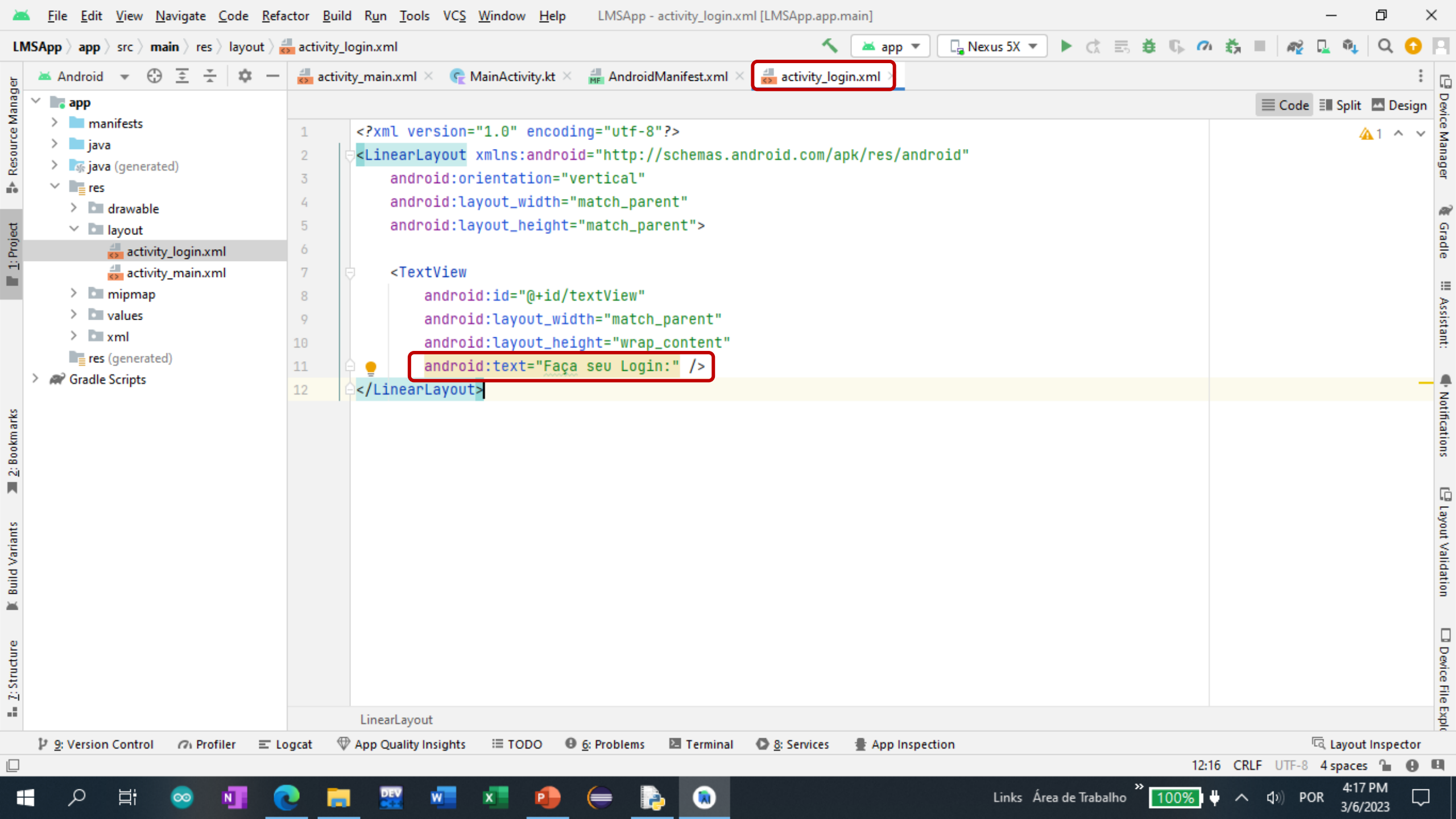
Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

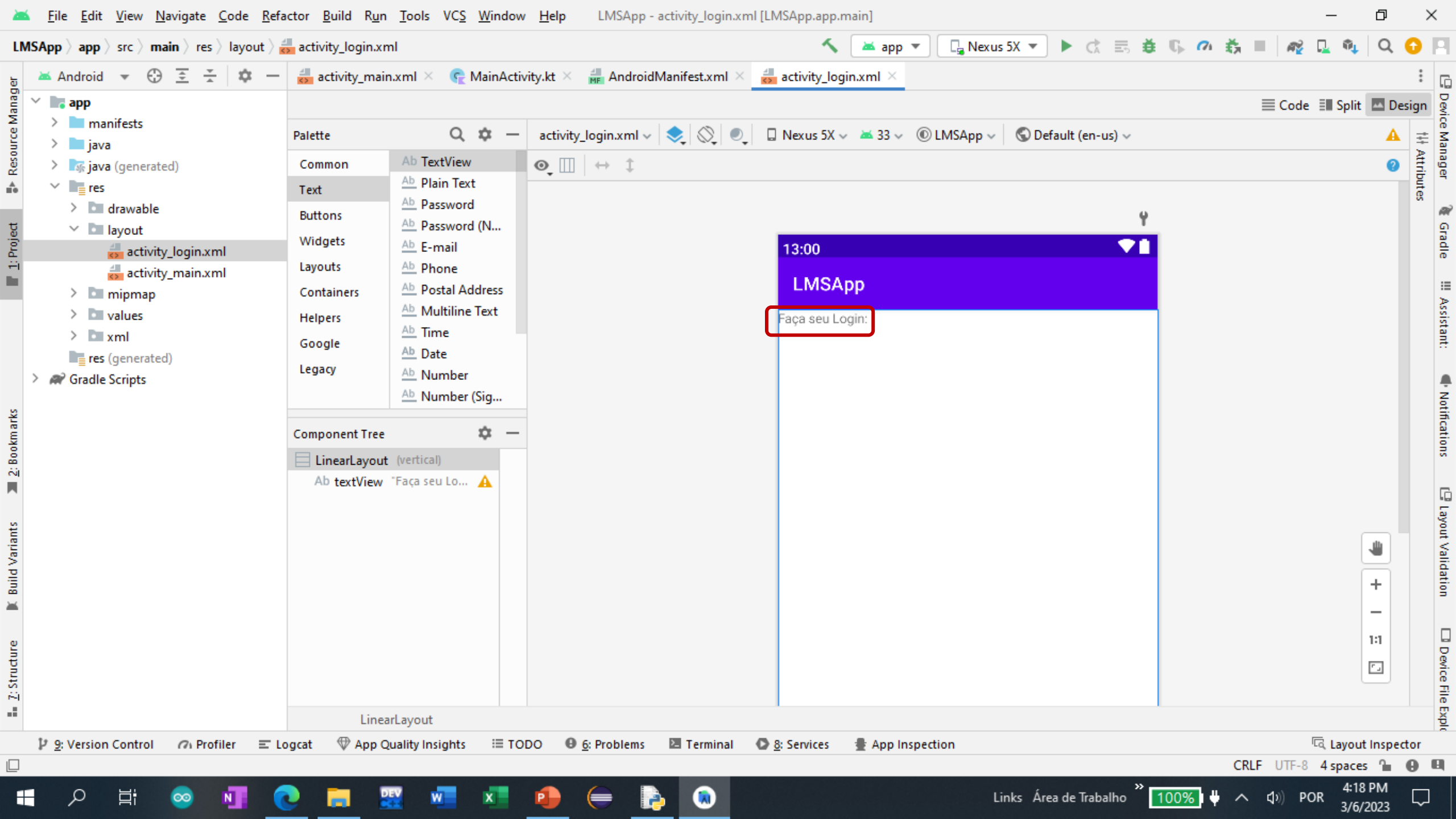
Links Área de Trabalho 100% 4:13 PM 3/6/2023

Pasta res – layout activity_main.xml

- Exercício(...):
 - Troque o texto do atributo android:text pelo texto **Faça seu Login:.**







Pasta res – layout activity_main.xml

- Exercício(...):
 - Referencie este layout na classe **MainActivity.kt**.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - MainActivity.kt [LMSApp.app.main]

LMSApp > app > src > main > java > br > com > ahcatani > lmsapp > MainActivity

activity_main.xml x MainActivity.kt x AndroidManifest.xml x activity_login.xml x

```

1 package br.com.ahcatani1.Lmsapp
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10        setContentView(R.layout.activity_login)
11    }
12 }

```

1: Project 2: Bookmarks Build Variants 2: Structure

Device Manager Gradle Assistant: Notifications Device File Explorer Emulator

9: Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection

Layout Inspector

12:2 LF UTF-8 4 spaces 100% POR 4:22 PM 3/6/2023

Links Área de Trabalho

Pasta res – string.xml

Pasta res – string.xml

- **strings.xml** é um arquivo de recursos com as mensagens e textos do aplicativo.
 - Organizar os textos em um arquivo.
 - Suporte a internacionalização (vários idiomas).
- Quando o aplicativo é criado, este arquivo contém o nome do aplicativo.

<string name="app_name">LMSApp</string>

- Cada recurso é definido pelo atributo name.
 - O valor está entre a abertura e o fechamento do marcador.

Pasta res – string.xml

- Estes recursos podem ser utilizados em qualquer arquivo xml do projeto.
 - **AndroidManifest.**
 - Outros recursos, como **layout.**
- Para acessar qualquer recurso de string de um arquivo XML (como layout), basta utilizar o seguinte padrão.
 - **@string/nome_recurso.**
 - Por exemplo: **@string/app_name.**

Pasta res – string.xml

- Exercício:
 - Crie um novo recurso de string chamado **mensagem_login** no arquivo **strings.xml**.
 - Coloque neste recurso a mensagem de sua tela de login.
 - Troque também o texto do recurso **app_name** e veja o que acontece no cabeçalho da tela de layout **activity_login.xml**.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - strings.xml [LMSApp.app.main]

LMSApp > app > src > main > res > values > strings.xml

Android > activity_main.xml x MainActivity.kt x strings.xml x AndroidManifest.xml x activity_login.xml x

1 <resources>
2 <string name="app_name">LMSAppNew</string>
3 <string name="mensagem_login">Faça seu Login:</string>
4 </resources>

resources > string

2:38 LF UTF-8 4 spaces 100% 4:44 PM 3/6/2023

Pasta res – string.xml

- Exercício:
 - Referencie este recurso no arquivo **activity_login.xml** para definir a mensagem na tela.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - activity_login.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout > activity_login.xml

Android app Nexus 5X

activity_main.xml MainActivity.kt strings.xml AndroidManifest.xml activity_login.xml

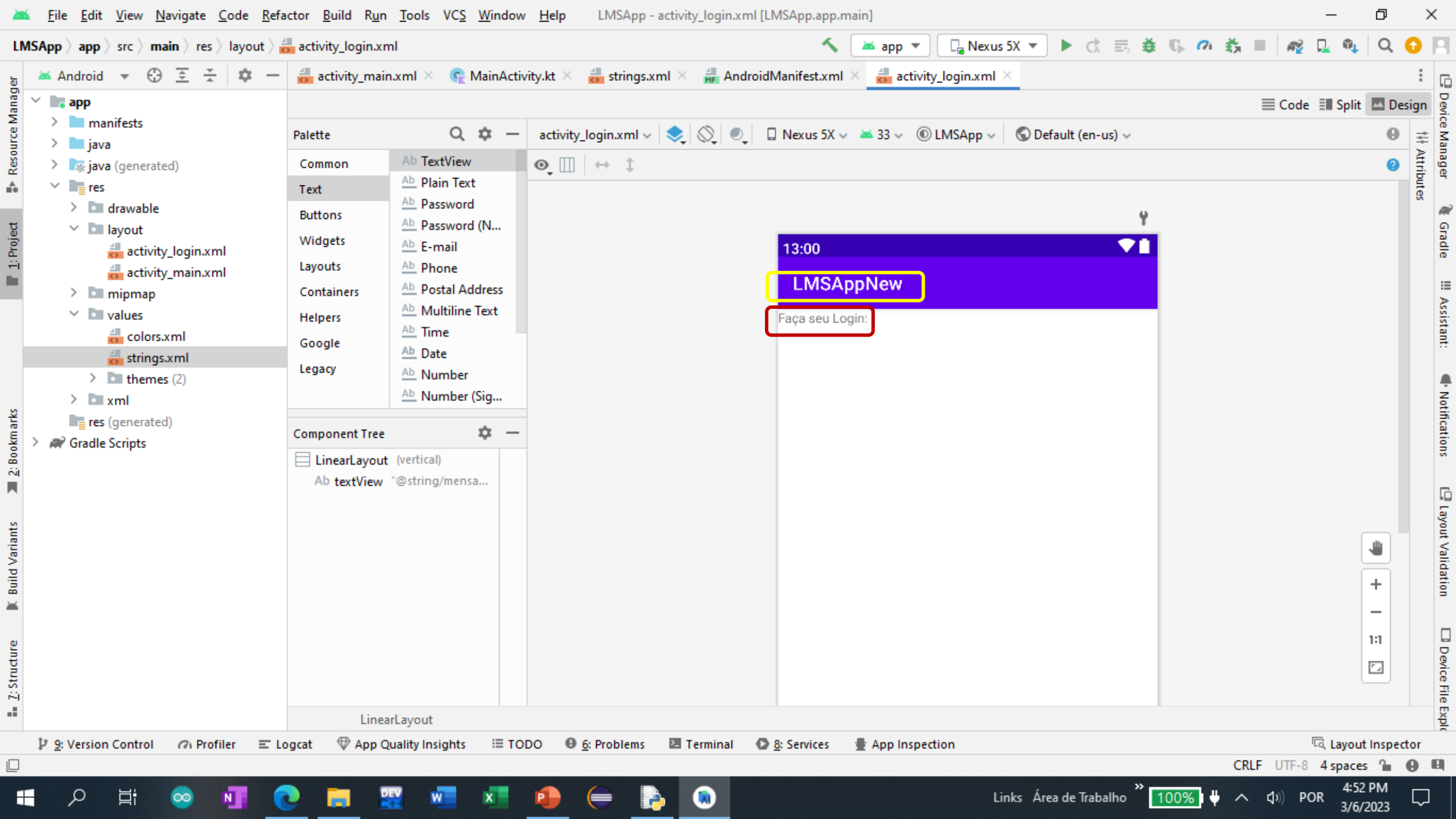
Code Split Design

1 <?xml version="1.0" encoding="utf-8"?>
 2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3 android:orientation="vertical"
 4 android:layout_width="match_parent"
 5 android:layout_height="match_parent">
 6
 7 <TextView
 8 android:id="@+id/textView"
 9 android:layout_width="match_parent"
 10 android:layout_height="wrap_content"
 11 android:text="@string/mensagem_login" />
 12 </LinearLayout>

LinearLayout

Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

12:16 CRLF UTF-8 4 spaces 100% 4:51 PM 3/6/2023



Recursos de imagens

Recursos de imagens

- As imagens utilizadas no aplicativo devem ficar em res/drawable.
 - No projeto do Android Studio, o ícone do aplicativo fica dentro de **res/mipmap**.
 - Para acessar os recursos de drawable.
@drawable/nome_arquivo
 - Para acessar os recursos de mipmap.
@mipmap/nome_arquivo

Recursos de imagens

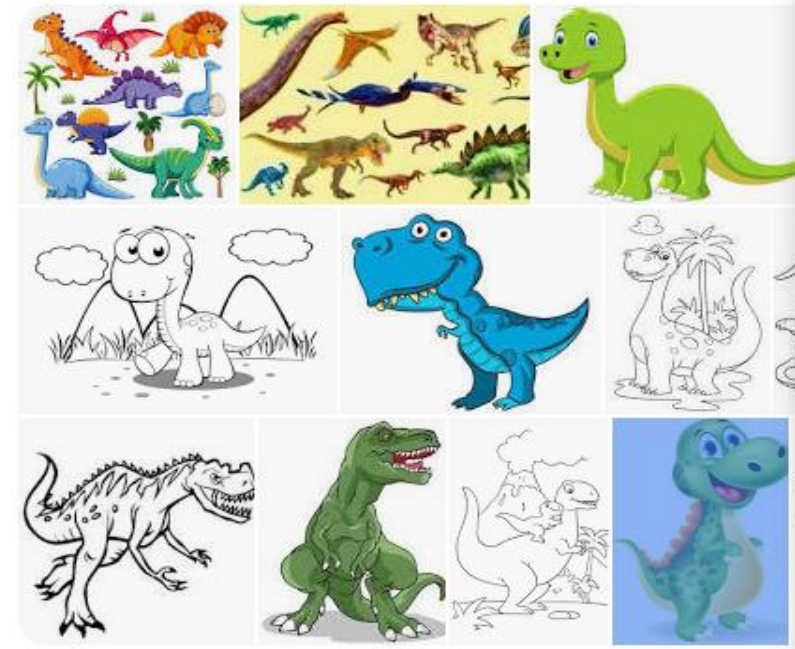
- Os recursos de imagem pode ser utilizados nos arquivos de layout.
- Faça o seguinte exercício:
 - Procure uma imagem na internet.
 - Coloque dentro de **res/drawable**.
 - Abra o arquivo de **layout activity_login.xml**.

Todas Imagens Shopping Videos Notícias Mais Ferramentas

Aproximadamente 204.000 resultados (0,51 segundos)

Imagens de dinossauros para imprimir

quarto infantil festa dinossauros kit festa



Ver tudo →

- Abrir link na nova guia
- Abrir link em Nova guia de modo do Internet Explorer
- Abrir link em uma nova janela
- Abrir link em uma janela InPrivate
- Gerar código QR para esta imagem
- Salvar link como
- Copiar link
- Abrir imagem em uma nova guia
- Salvar imagem como
- Copiar imagem**
- Copiar link de imagem
- Editar imagem
- Pesquisar a imagem na Web
- Pesquisa Visual
- Adicionar a Coleções
- Compartilhar
- Seleção da web Ctrl+Shift+X
- Captura da web Ctrl+Shift+S
- Inspecionar



785 x 1.024

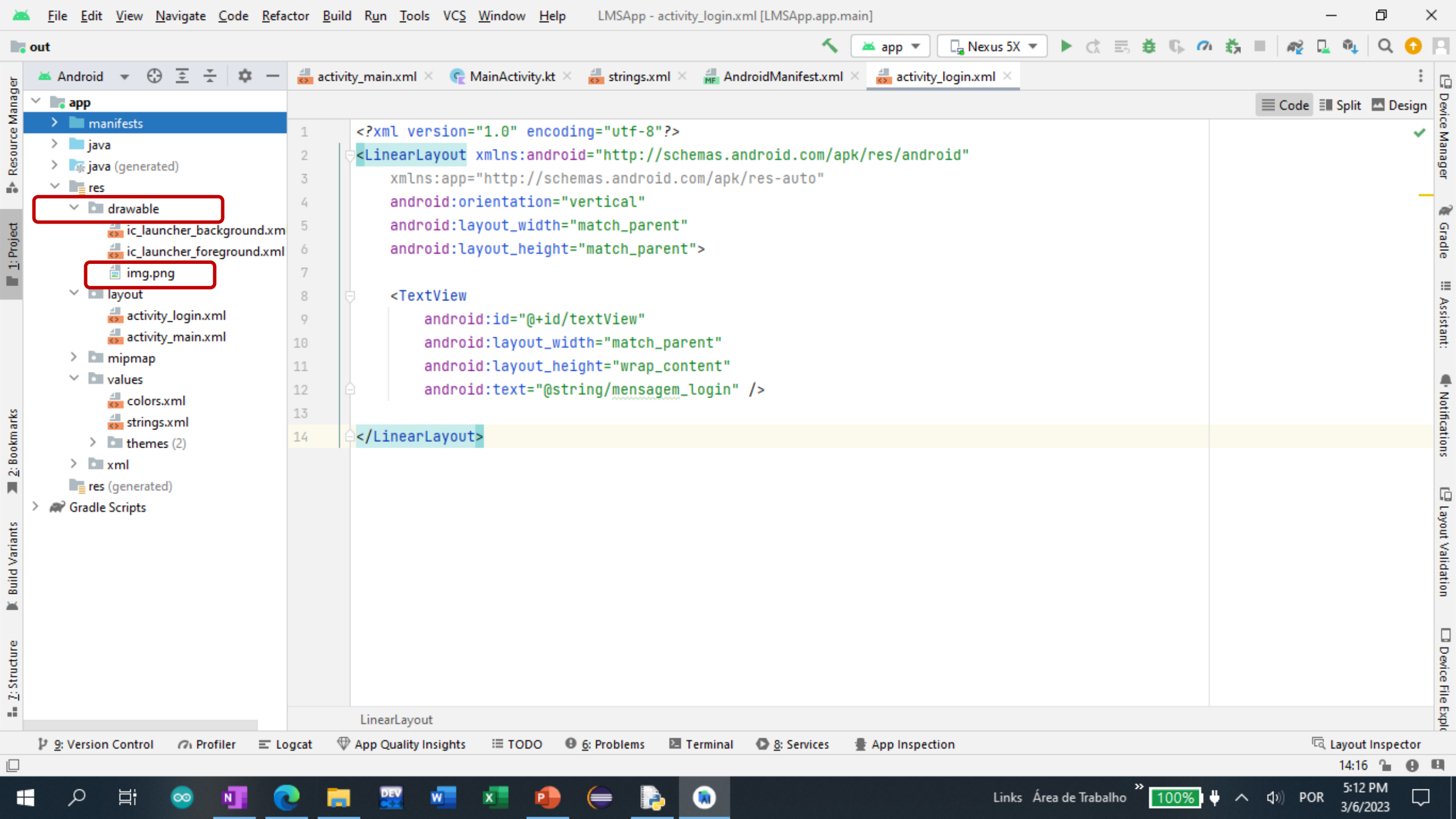
ho de Dinossauro

Visitar

solde para ...

tos autorais. Saiba mais

nado



Recursos de imagens

- Utilizando o editor visual, procure pelo componente **ImageView**, clique, arraste e solte dentro da imagem do SmartPhone.
- Selecione a imagem na tela que abrir.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - activity_login.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout > activity_login.xml

Android activity_main.xml MainActivity.kt strings.xml AndroidManifest.xml activity_login.xml

app

- manifests
- java
- java (generated)
- res
 - drawable
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml
 - img.png
 - layout
 - activity_login.xml
 - activity_main.xml
 - mipmap
 - values
 - colors.xml
 - strings.xml
 - themes (2)
 - xml
 - res (generated)
- Gradle Scripts

Palette

- Common
 - View
 - WebView
 - VideoView
 - CalendarView
 - ProgressBar
 - SeekBar
 - SeekBar (Disc...)
 - RatingBar
 - SearchView
 - TextureView
- Text
- Buttons
- Widgets
- Layouts
- Containers
- Helpers
- Google
- Legacy

Component Tree

- LinearLayout (vertical)
 - Ab textView ~@string/mensa...

activity_login.xml

Nexus 5X 33 LMSApp Default (en-us)

13:00

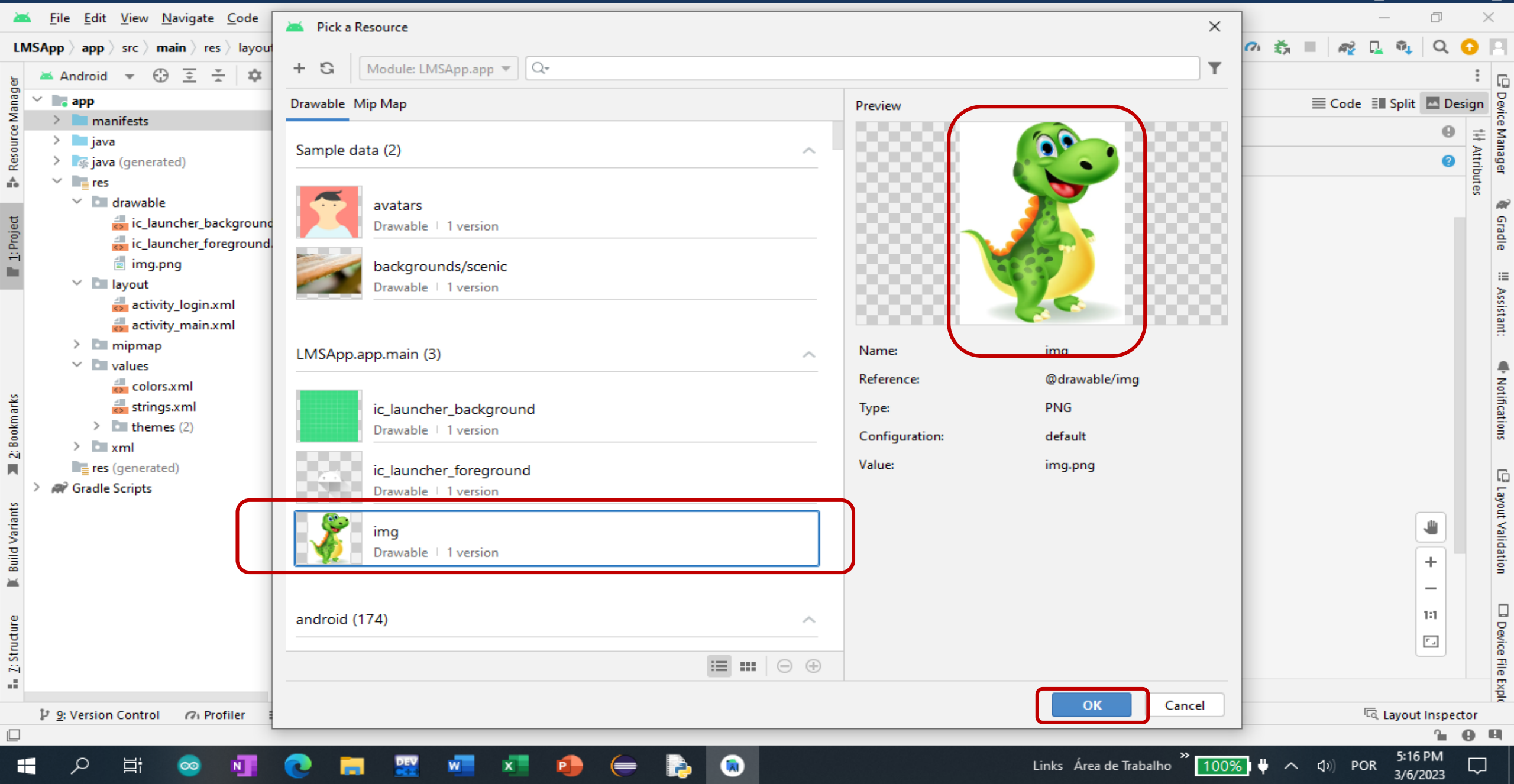
LMSAppNew

Faça seu Login:

LinearLayout

Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

Links Área de Trabalho 100% POR 5:15 PM 3/6/2023



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - activity_login.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout > activity_login.xml

Android > app

- manifests
- java
- java (generated)
- res
 - drawable
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml
 - img.png
 - layout
 - activity_login.xml
 - activity_main.xml
 - mipmap
 - values
 - colors.xml
 - strings.xml
 - themes (2)
 - xml
 - res (generated)
- Gradle Scripts

Palette

- Common
 - View
 - WebView
 - VideoView
 - CalendarView
 - ProgressBar
 - SeekBar
 - SeekBar (Disc...)
 - RatingBar
 - SearchView
 - TextureView
- Text
- Buttons
- Layouts
- Containers
- Helpers
- Google
- Legacy

Component Tree

- LinearLayout (vertical)
 - Ab textView ~@string/mensa...
 - imageView2

activity_login.xml

Nexus 5X

33

LMSApp

Default (en-us)

Design

Attributes

Device Manager

Gradle

Assistant

Notifications

Layout Validation

Device File Expl...

Version Control

Profiler

Logcat

App Quality Insights

TODO

Problems

Terminal

Services

App Inspection

Layout Inspector

Links Área de Trabalho 100%

5:18 PM 3/6/2023

Recursos de imagens

- Veja no arquivo XML que o atributo **src** foi configurado para a imagem selecionada **@drawable/nome_imagem**.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - activity_login.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout > activity_login.xml

Android app Nexus 5X

activity_main.xml MainActivity.kt strings.xml AndroidManifest.xml activity_login.xml

Code Split Design

3

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   android:orientation="vertical"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent">
7
8   <TextView
9     android:id="@+id/textView"
10    android:layout_width="match_parent"
11    android:layout_height="wrap_content"
12    android:text="@string/mensagem_login" />
13
14   <ImageView
15     android:id="@+id/imageView2"
16     android:layout_width="match_parent"
17     android:layout_height="wrap_content"
18     app:srcCompat="@drawable/img" />
19
20 </LinearLayout>

```

LinearLayout

Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

20:16 100% POR 5:21 PM 3/6/2023

Recursos de imagens

- Outra forma de definir o recurso de imagem em um elemento de tela é a partir do código **Kotlin/Java**.
 - Neste caso, utilizamos a referência para a classe **R**.
- Para isso será necessário definir um identificador para a **ImageView** do **layout**.
 - Localize o atributo **android:id** na **ImageView** do seu **Layout**.
 - Repare na sintaxe do valor deste atributo: **@+id/identificador**.
 - **id** é o tipo do recurso.
 - **Identificador** é o nome que o programador cria.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LMSApp - activity_login.xml [LMSApp.app.main]

LMSApp > app > src > main > res > layout > activity_login.xml

Android app Nexus 5X

activity_main.xml MainActivity.kt strings.xml AndroidManifest.xml activity_login.xml

Code Split Design

app

- manifests
- java
- java (generated)
- res
 - drawable
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml
 - img.png
 - layout
 - activity_login.xml
 - activity_main.xml
 - mipmap
 - values
 - colors.xml
 - strings.xml
 - themes (2)
 - xml
 - res (generated)
- Gradle Scripts

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   android:orientation="vertical"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent">
7
8   <TextView
9     android:id="@+id/textView"
10    android:layout_width="match_parent"
11    android:layout_height="wrap_content"
12    android:text="@string/mensagem_login" />
13
14   <ImageView
15     android:id="@+id/imageView2"
16     android:layout_width="match_parent"
17     android:layout_height="wrap_content"
18     app:srcCompat="@drawable/img" />
19
20 </LinearLayout>
  
```

LinearLayout

Version Control Profiler Logcat App Quality Insights TODO 6: Problems Terminal 8: Services App Inspection Layout Inspector

20:16 4 spaces 100% POR 5:25 PM 3/6/2023

Recursos de imagens

- Um **id** é um recurso assim como **imagens**, **estilos**, **strings** e **layout**, porém criados dentro dinamicamente dentro de outros recursos.
 - Por isso utiliza-se **@+id**, indicando que é um recurso novo.
- Os **ids** estão também na classe **R**, e podem ser utilizados nos códigos **Kotlin** e **Java**.

Recursos de imagens

- No seu ImageView da tela de layout remova o valor do atributo **src** ou **srcCompat**.
- Procure pelo atributo id e troque para um nome que identifique seu **ImageView** (por exemplo **campolImagem**).
 - Veja que este recurso está mapeado na classe **R.java**.
- É possível recuperar este **ImageView** e qualquer outro elemento de tela a partir da **Activity** da tela pelo **View Binding**.

View Binding

View Binding

- **View Binding** é um recurso do Android que facilita a programação de códigos que utilizam elementos da interface gráfica (**View**).
- Sempre que o **View Binding** é utilizado, uma classe que representa o arquivo xml de layout é criada automaticamente.
- Quando o app é executado, uma instância dessa classe é criada, com referência a todos os elementos de tela do arquivo xml que ela referencia.

View Binding

- O Nome da classe criada é baseado no nome do arquivo xml.
 - activity_main.xml → ActivityMainBinding.
 - login.xml → LoginBinding.

View Binding

- O primeiro passo para usar **View Binding** é configurar o arquivo **build.gradle** do Módulo.
 - Acesse a árvore de arquivos **Gradle Scripts** do seu projeto.
 - Localize e abra o arquivo **build.gradle** (Module:xxxx).
 - Dentro da seção **android {}** coloque o seguinte código:


```
buildFeatures {
    viewBinding = true
}
```
 - Clique em **Sync now** na barra amarelo em cima do arquivo para atualizar o projeto.

View Binding

- Abra **MainActivity** e no começo da classe (dentro da primeira chave) coloque o seguinte código para instanciar a **View Binding** da tela de login.

```
private val binding by lazy {
    LoginBinding.inflate(layoutInflater)
}
```

- Aperte alt+enter para corrigir a importação de **LoginBinding**.
- A variável **binding** agora tem uma instância da tela **login.xml**, com todas as referências para elementos de tela.

View Binding

- Ainda em **MainActivity**, agora dentro do método **onCreate** altere o parâmetro de **setContentView()** para **binding.root**.

setContentView(binding.root)

View Binding

- Os elementos de tela criados no xml são acessados pelo **id** definido no **xml**, a partir de um atributo do binding criado.
 - Se o **id** contém underline (), ele some e a próxima letra fica maiúscula.
 - **id imagem_login** → **view binding: imagemLogin**.
- Portanto, para acessar o elemento de imagem `campolmagem` criado anteriormente e alterar o atributo de imagem faça o seguinte.
`binding.campolmagem.setImageResource(R.drawable.imagem_login)`
- Execute seu aplicativo para ver o resultado.

Recursos de imagens

Recursos de imagens

- O mesmo procedimento pode ser feito para alterar os textos dos elementos de tela.
- Exercício:
 - Alterar o texto do **TextView** que está no arquivo de **layout** pelo código **Kotlin**, colocando o valor do recurso de string criado anteriormente.
 - Utilize o atributo **text** do objeto de **TextView** recuperado.

Tratamento de eventos

Tratamento de eventos

- Para estudar o tratamento de eventos, fazer os procedimentos.
- Coloque no arquivo **login.xml** elementos para fazer um login.
 - Um campo de **texto** com a palavra **Usuário**.
 - Um campo de **entrada de texto**, para digitar o usuário.
 - Um campo de **texto** com a palavra **Senha**.
 - Um campo de **entrada de texto** para digitar a senha.
 - Um **botão** para fazer o login.

Tratamento de eventos

- Para estudar o tratamento de eventos, fazer os procedimentos.
 - Definir novos **ids** para estes elementos.
 - Mantenha os elementos que existiam antes (**texto inicial** e **imagem**).
 - Recupere os elementos de **entrada de texto** e o **botão** de tela dentro do método **onCreate** da **Activity**.

Tramento de Eventos

Usuário

Senha

LOGIN

Tratamento de eventos

- Os **eventos** de tela são vinculados tratados dentro da **Activity**.
- A vinculação de um evento é feito utilizando o método **setOnClickListener**.
- Este método pode receber como argumento uma **função anônima**, que será responsável por tratar o evento.
 - Ou seja, será executada quando acontecer o **evento**.

Tratamento de eventos

- Por exemplo, mostrar uma mensagem **Toast** após clicar no botão.
 - Repare na sintaxe **lambda** do Kotlin.

```
binding.botaoLogin.setOnClickListener {
    Toast.makeText(this,
                    "Clicou no botão",
                    Toast.LENGTH_LONG).show())
}
```

Tratamento de eventos

- Agora faça os código para mostrar os textos nos campos de usuário e senha, digitados pelo usuário utilizando o **Toast**.
 - Dica: utilize a propriedade text do objeto **EditText**.

Tratamento de eventos

- Segunda opção: delegar para um método.
 - A vantagem é que o método pode ser reutilizado para botões que tem o mesmo tratamento de evento:

```

override fun onCreate(savedInstanceState: Bundle?) {
    // Código do OnCreate //

    segunda forma: delegar para método
    binding.botaoLogin.setOnClickListener {
        onClickLogin()
    }
}

```

Tratamento de eventos

```
fun onClickLogin() {
    val valorUsuario =
        campo_usuario.text.toString()
    val valorSenha = campo_senha.text.toString()
    Toast.makeText(this,
                    "$valorUsuario : $valorSenha",
                    Toast.LENGTH_LONG).show()
}
```

Referências e créditos

- Aula baseada nos textos do livro:
 - LECHETA, R. R. Android Essencial com Kotlin. Edição: 1ª ed. Novatec, 2017.

Github

- Link com os códigos gerados nesta aula:
 - https://github.com/fesousa/aula-android-kotlin2022/tree/LMSApp_01_Intr



Obrigado!

Prof. MSc. Antônio Catani
antonio.catani@faculdadeimpacta.com.br