

INSTITUTO FEDERAL

Paraná

Campus Paranaguá

Android Studio

Projetos Android

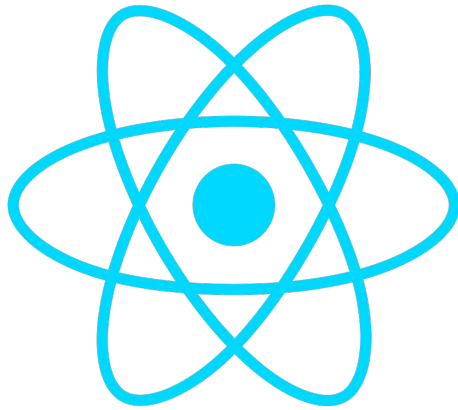
Dispositivos Móveis

Prof. Diego Stiehl

Aplicações Nativas

- Nesta disciplina aprenderemos a criar aplicações Android Nativas
 - Aplicações diretamente compiladas para execução na arquitetura do Android
 - Podemos usar as linguagens Java ou **Kotlin**
 - Usaremos o SDK e plataformas oficiais
 - Nosso código conhecerá diretamente as classes do Java API Framework
 - Possibilidade de controle total de toda a App

Alternativas



React Native



Flutter



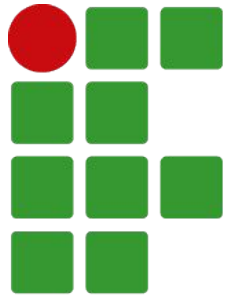
Xamarin



ionic



Adobe PhoneGap



INSTITUTO FEDERAL

Paraná

Campus Paranaguá

Android Studio, SDK, Emuladores e a estrutura de projetos Android

Ambiente de Desenvolvimento

Plataformas

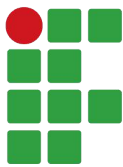
- Desde as primeiras versões, o Android utiliza um conjunto de ferramentas padrão
 - ~~— Eclipse IDE / Android Developer Tools (ADT)~~
 - Java Development Kit (JDK)
 - Android Software Developer's Kit (SDK)
 - Android Studio
 - Linguagens: Java e Kotlin
- Embora esta não seja a única forma de desenvolver para Android

Android SDK

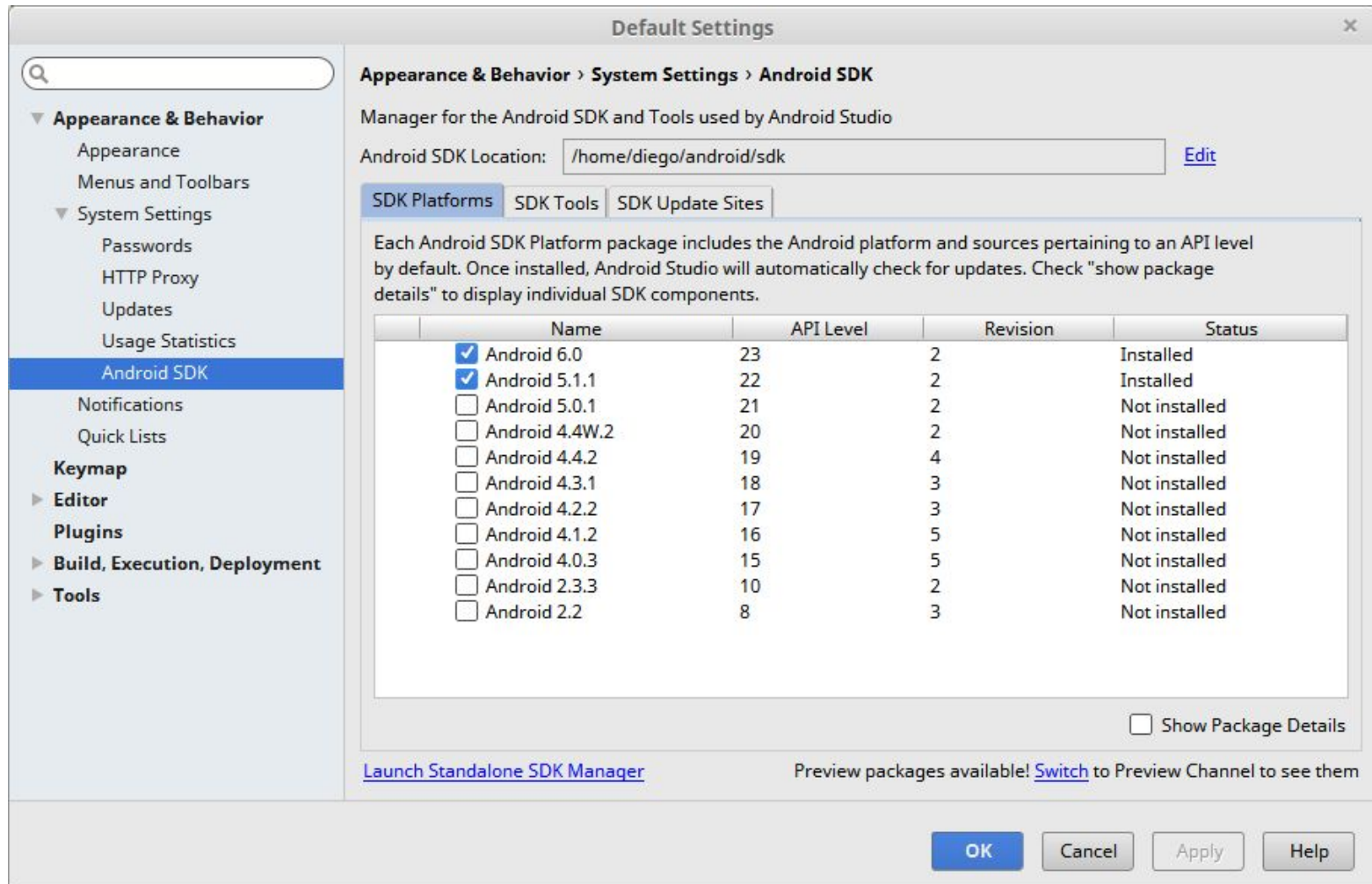
- É a ferramenta responsável por oferecer um ambiente de desenvolvimento e testes de aplicações
 - Pode ser baixado isoladamente em:
 - <http://developer.android.com/sdk/>
 - Mas o ideal é utilizar o Android Studio
 - Ele já coordena a instalação do SDK e suas ferramentas
 - Pode ser baixado no mesmo link

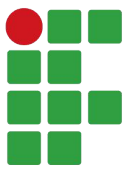
Android SDK

- O SDK oferece as diversas opções de sistemas Android para o desenvolvedor
 - SDKs de versões específicas (Java/Kotlin)
 - Dispositivos virtuais
 - Código-fonte e documentação
 - Outras ferramentas
- O programador pode criar um emulador
 - No formato de máquina virtual
 - Qualquer versão do Android



SDK Manager





Emulador

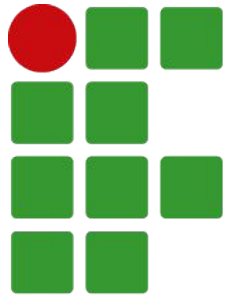
- É uma boa ferramenta para testes
- Não demonstra o uso perfeito e completo de um dispositivo Android
 - Características de difícil emulação:
 - Acelerômetro
 - GPS
 - 3G
 - Vibração
 - ...
 - Mas podemos executar e debugar nossos aplicativos em um dispositivo real (USB)

Emulador

- Algumas funcionalidades:
 - https://storage.googleapis.com/androiddevelopers/videos/studio-emulator-overview_2-2.mp4

<https://developer.android.com/studio/run/emulator.html>





INSTITUTO FEDERAL

Paraná

Campus Paranaguá

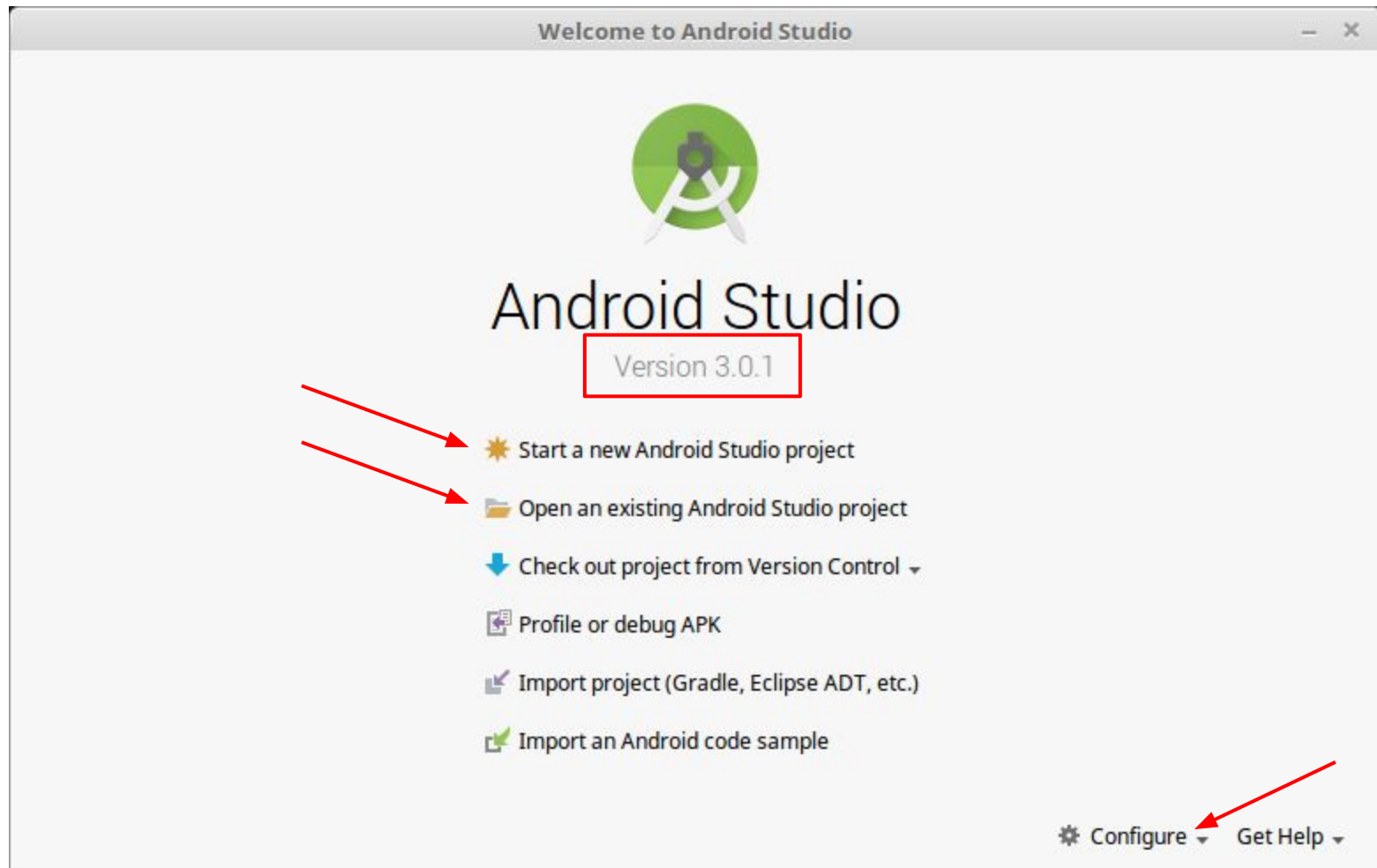
Criando e explorando um projeto simples

Android Studio

Android Studio

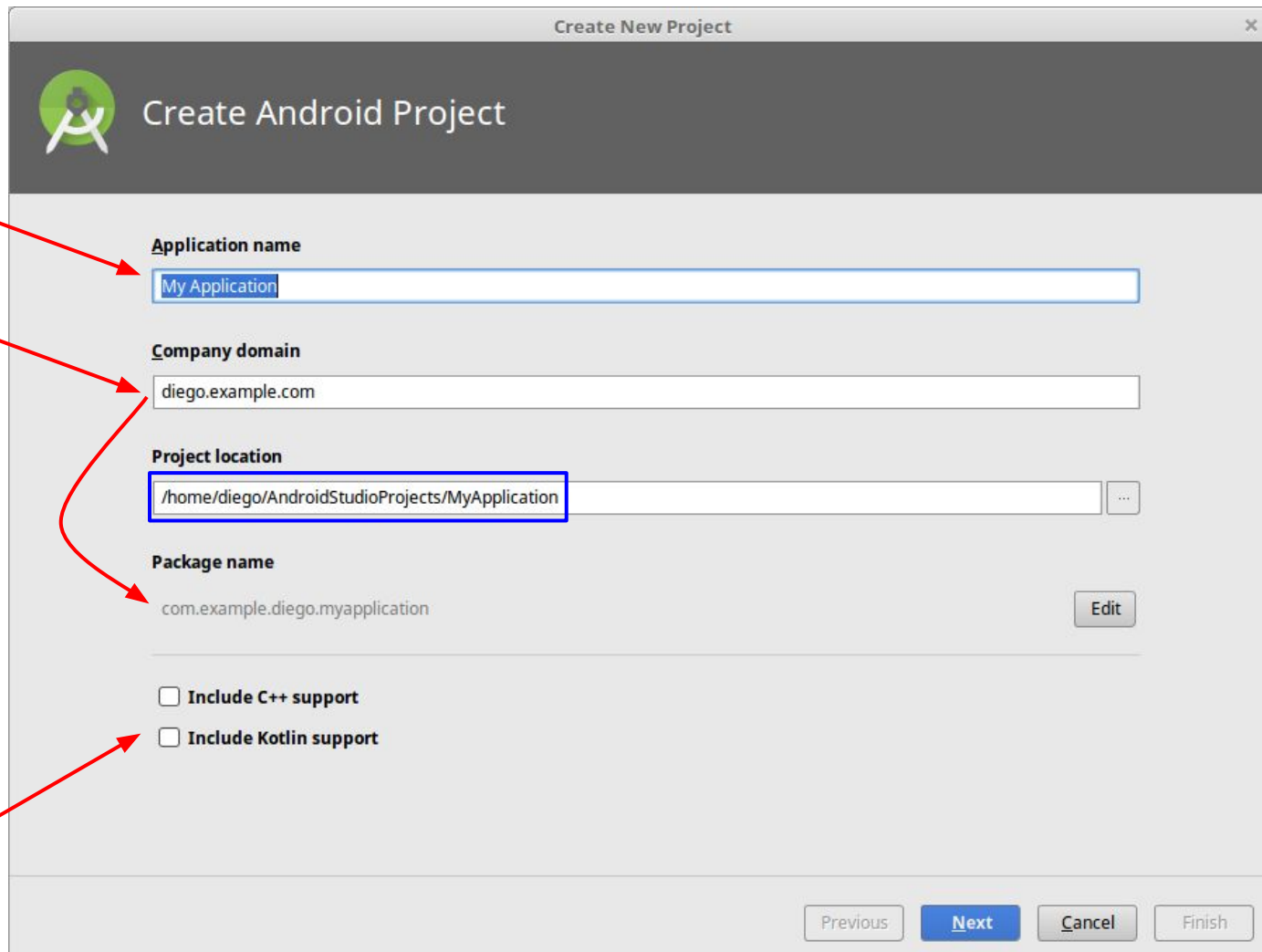
- É uma IDE para desenvolvimento Android
 - Baseado em IntelliJ IDEA
- Se conecta às configurações do SDK
 - Importa emuladores
 - Permite execução e debug de aplicações nos emuladores e dispositivos reais
 - Compila e empacota aplicações
 - ...
- É por ele que faremos “tudo”

Android Studio



O layout e a navegação desta e das próximas telas costumam mudar de uma versão para outra do Android Studio

Criando um Projeto



Create New Project

Create Android Project

Application name
My Application

Company domain
diego.example.com

Project location
/home/diego/AndroidStudioProjects/MyApplication

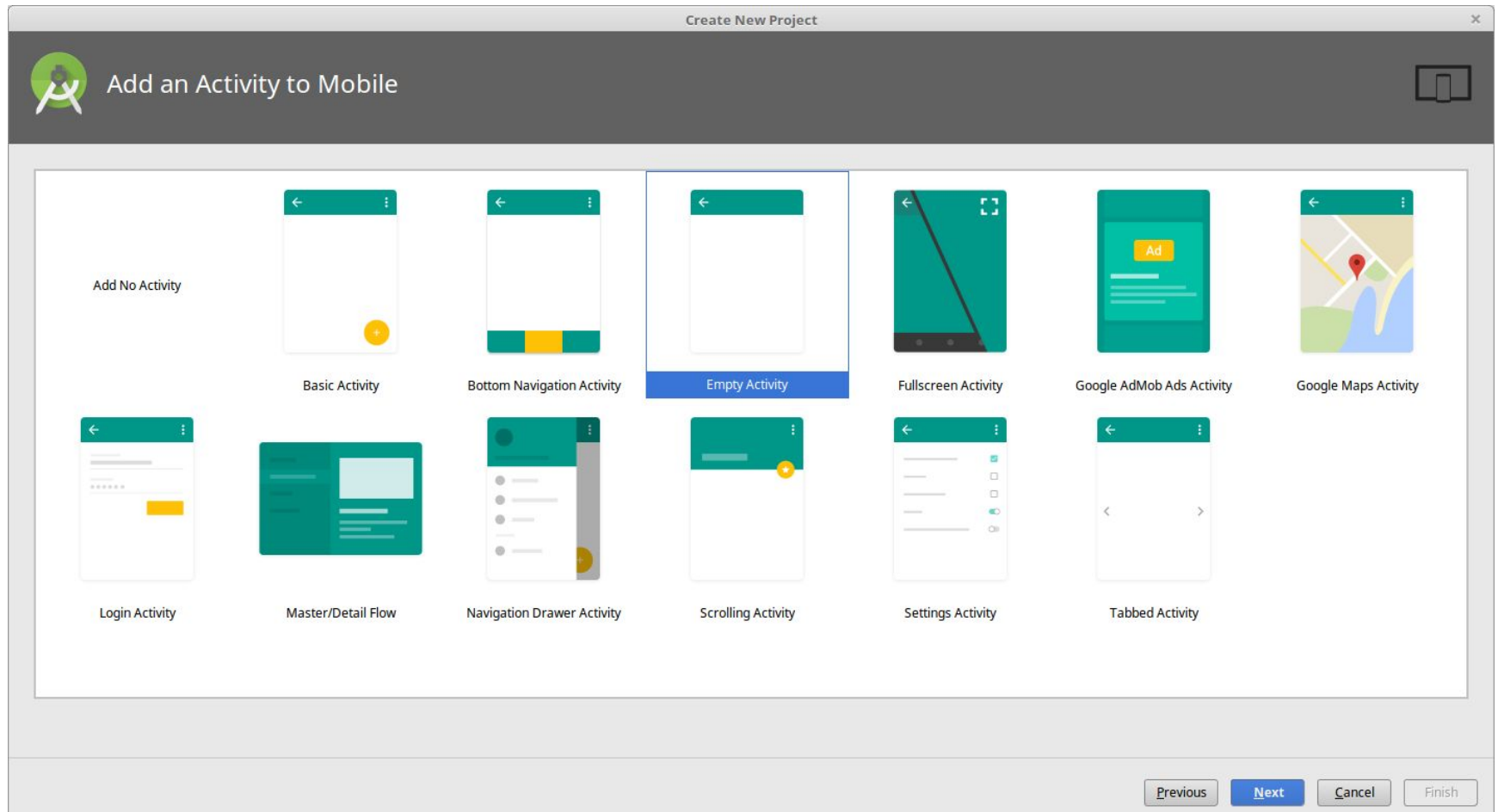
Package name
com.example.diego.myapplication Edit

☐ Include C++ support

☐ Include Kotlin support

Previous Next Cancel Finish

Nova Activity



Nova Activity

Create New Project

Configure Activity

Creates a new empty activity

Activity Name

MainActivity

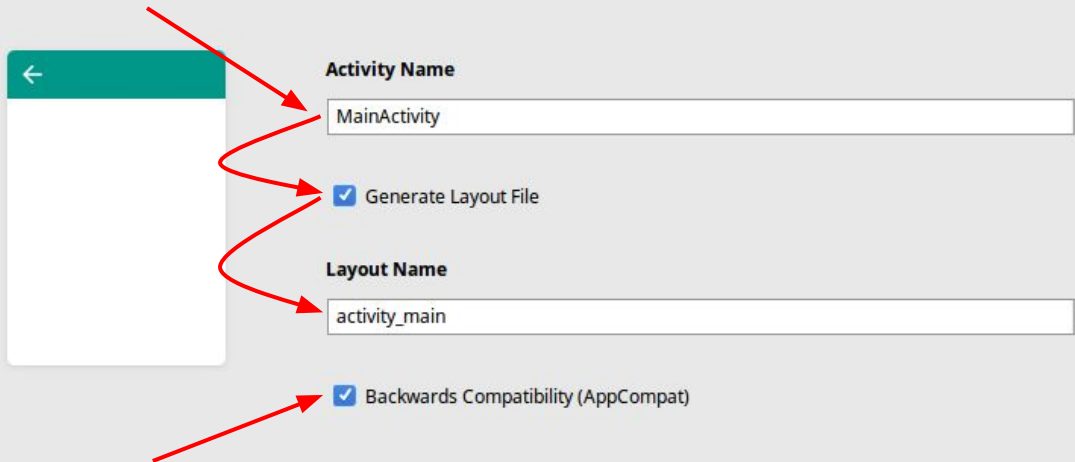
☒ Generate Layout File

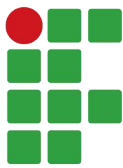
Layout Name

activity_main

☒ Backwards Compatibility (AppCompat)

Previous Next Cancel Finish





INSTITUTO
FEDERAL

Projeto Aberto

The screenshot displays the Android Studio 3.0.1 interface. The top toolbar includes icons for running the app, debugging, and other development tools. The left sidebar shows the Project, Structure, and Gradle Scripts views. The main editor displays the MainActivity.java file.

Project Structure (Left Sidebar):

- app
 - manifests
 - AndroidManifest.xml
 - java
 - com.example.diego.meuapp
 - MainActivity
 - com.example.diego.meuapp (androidTest)
 - com.example.diego.meuapp (test)
 - res
 - drawable
 - ic_launcher_background.xml
 - ic_launcher_foreground.xml (v24)
 - layout
 - activity_main.xml
 - mipmap
 - ic_launcher.png (5)
 - ic_launcher.xml (anydpi-v26)
 - ic_launcher_round.png (5)
 - ic_launcher_round.xml (anydpi-v26)
 - values
 - colors.xml
 - strings.xml
 - styles.xml
- Gradle Scripts
 - build.gradle (Project: MeuApp)
 - build.gradle (Module: app)
 - gradle-wrapper.properties (Gradle Version)
 - proguard-rules.pro (ProGuard Rules for app)
 - gradle.properties (Project Properties)
 - settings.gradle (Project Settings)
 - local.properties (SDK Location)

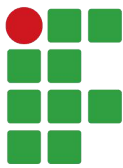
Main Activity Code (MainActivity.java):

```
1 package com.example.diego.meuapp;
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 }
13
14
```

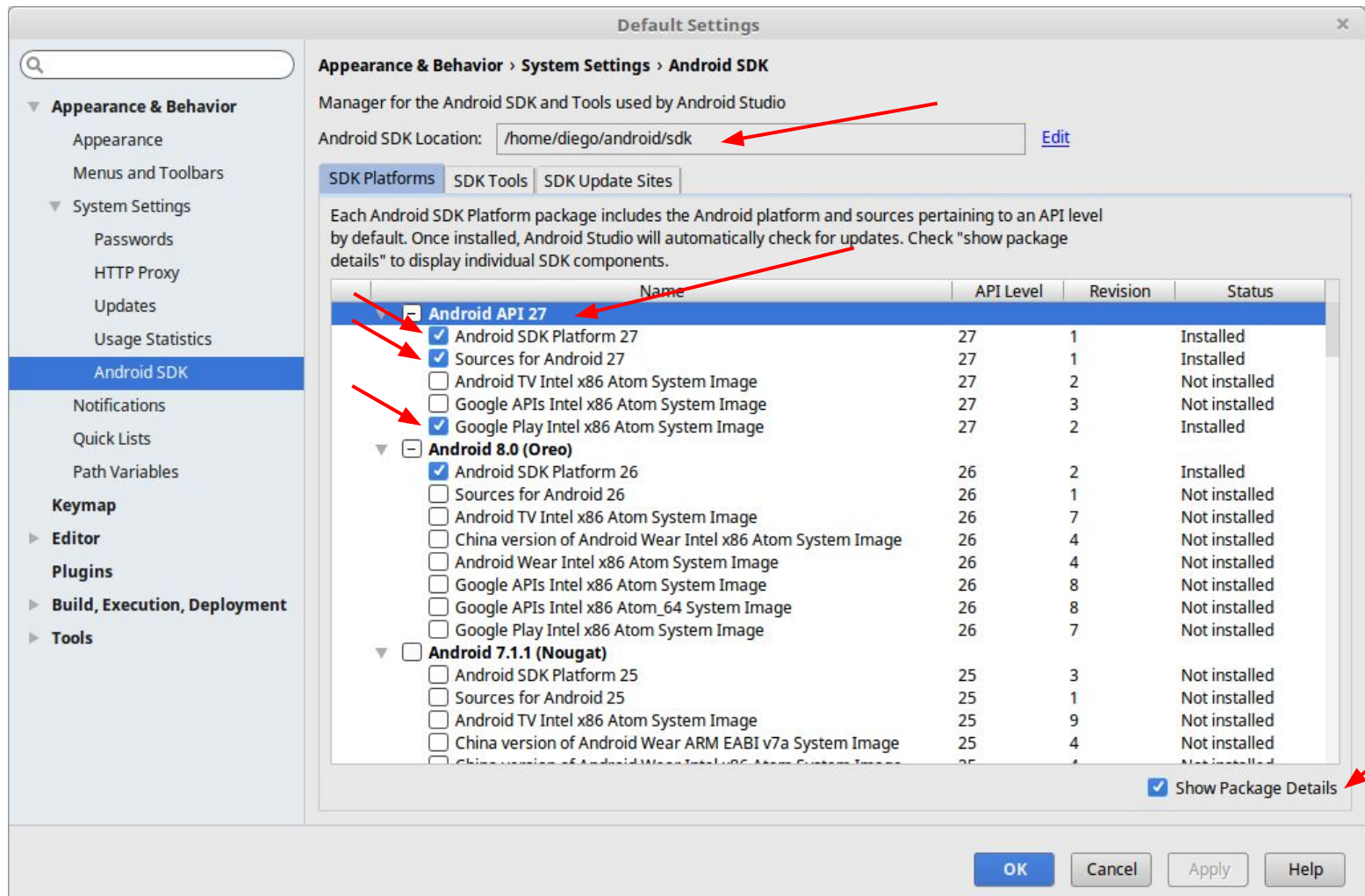
The bottom status bar shows the emulator process finished with exit code 0 (16 minutes ago). The bottom right corner displays the time 13:2, LF, UTF-8, and Context: <no context>.

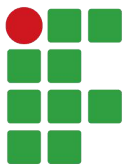
IDE e Estrutura do Projeto

- SDK Manager
- Android Virtual Device (AVD) Manager
 - Emulador
- LogCat
 - Android Monitor
- File → Project Structure
 - SDK location
 - Guia Modules/app
 - Min SDK
 - App Id
- Gradle scripts
 - build.gradle (app)



SDK Manager





LogCat

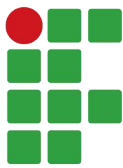
The screenshot shows the Android Studio IDE with the MainActivity.java file open. The LogCat window at the bottom displays the following log entries:

```
02-11 19:58:35.685 11351-11351/? I/zygote: Not late-enabling -Xcheck:jni (already on)
02-11 19:58:35.700 11351-11351/? W/zygote: Unexpected CPU variant for X86 using defaults: x86
02-11 19:58:35.719 11351-11358/? I/zygote: Debugger is no longer active
02-11 19:58:35.795 11351-11351/? I/InstantRun: starting instant run server: is main process
02-11 19:58:35.891 11351-11351/? I>Hello: Faaaaaala galera de TADS!
02-11 19:58:35.904 11351-11370/? D/OpenGLRenderer: HWUI GL Pipeline
02-11 19:58:35.945 11351-11370/? I/zygote: android:hardware:configstore::V1_0::ISurfaceFlingerConfigs::hasWideColorDisplay retrieved: 0
02-11 19:58:35.945 11351-11370/? I/OpenGLRenderer: Initialized EGL, version 1.4
02-11 19:58:35.945 11351-11370/? D/OpenGLRenderer: Swap behavior 1
02-11 19:58:35.945 11351-11370/? W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
02-11 19:58:35.945 11351-11370/? D/OpenGLRenderer: Swap behavior 0
02-11 19:58:35.948 11351-11370/? D/EGL_emulation: eglCreateContext: 0xa5485120: maj 2 min 0 rcv 2
02-11 19:58:36.010 11351-11370/? D/EGL_emulation: eglMakeCurrent: 0xa5485120: ver 2 0 (tinfo 0xa5483290)
02-11 19:58:36.054 11351-11370/? D/EGL_emulation: eglMakeCurrent: 0xa5485120: ver 2 0 (tinfo 0xa5483290)
```

Red arrows in the image point to the LogCat window, the 'Verbose' filter, the search bar, and the 'Show only selected application' button.

LogCat

- É nele que serão mostradas todas as mensagens vindas do dispositivo
 - Erros, Alertas, Informação, ...
 - Não somente do nosso aplicativo
- Podemos escrever nossas mensagens de log
 - Log.e(...), Log.w(...), Log.i(...), ...
- Podemos filtrar por tipo e conteúdo



INSTITUTO
FEDERAL

Debug

The screenshot shows the Android Studio 3.0.1 IDE with the MainActivity.java file open. The debug menu is open, showing various actions and their keyboard shortcuts. Red arrows point to specific actions and the Variables panel.

Debug Menu Actions and Shortcuts:

- Run 'app' (Shift+F10)
- Apply Changes (Ctrl+F10)
- Debug 'app' (Shift+F9)
- Run 'app' with Coverage
- Profile 'app'
- Run...
- Debug...
- Profile...
- Record Espresso Test
- Attach to Local Process...
- Edit Configurations...
- Import Test Results
- Apply Changes (Ctrl+F10)
- Stop 'app'
- Show Running List
- Restart Activity
- Step Over
- Force Step Over
- Step Into
- Force Step Into
- Smart Step Into
- Step Out
- Run to Cursor
- Force Run to Cursor
- Drop Frame
- Pause Program
- Resume Program
- Evaluate Expression...
- Quick Evaluate Expression
- Show Execution Point
- Toggle Line Breakpoint
- Toggle Method Breakpoint
- Toggle Temporary Line Breakpoint
- Toggle Breakpoint Enabled
- View Breakpoints...
- Get thread dump
- Attach debugger to Android process

Code Snippet:

```
package com.example.diego.meuapp;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i("tag", "Altera de TADS!");
    }
}
```

Debugger Console:

main@5.025 in group "main": RUNNING

onCreate:11, MainActivity (com.example.diego.meuapp)

performCreate:6999, Activity (android.app)

performCreate:6990, Activity (android.app)

callActivityOnCreate:1214, Instrumentation (android.app)

performLaunchActivity:2731, ActivityThread (android.app)

handleLaunchActivity:2856, ActivityThread (android.app)

-wrap11:-1, ActivityThread (android.app)

handleMessage:1589, ActivityThread\$H (android.app)

dispatchMessage:106, Handler (android.os)

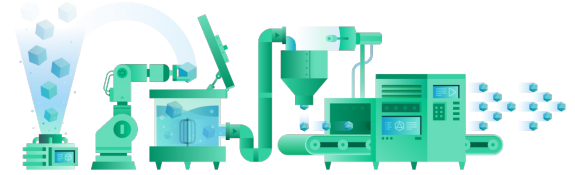
Variables Panel:

- this = {MainActivity@5154}
- savedInstanceState = null

Gradle?

- Gerencia dependências da aplicação
- Executa scripts
- Vocês conhecem?
 - Maven
 - Ant
 - Composer (PHP)
 - Bundler (Ruby)
 - **NPM (Node / JavaScript)**

Gradle



- Ferramenta de automação de builds e gestão de dependências e configurações
 - Baseado em Ant e Maven
- Android Studio utiliza o Gradle em todo o processo de criação dos aplicativos
 - Da informação das dependências de projeto
 - Ao empacotamento para publicação do App
- `build.gradle (Project: nome)`
- `build.gradle (Module: app)` ← Neste que vamos mexer mais

build.gradle (Module: app)

apply **plugin: 'com.android.application'**

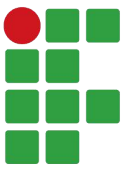
```
android {  
    compileSdkVersion 26  
    defaultConfig {  
        applicationId "com.example.diego.meuapp"  
        minSdkVersion 15  
        targetSdkVersion 26  
        versionCode 1  
        versionName "1.0"  
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    }  
    buildTypes {  
        release {  
            minifyEnabled false  
            proguardFiles getDefaultProguardFile("proguard-android.txt"), "proguard-rules.pro"  
        }  
    }  
}  
  
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'com.android.support:appcompat-v7:26.1.0'  
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.1'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'  
}
```

IDE e Estrutura do Projeto

- AndroidManifest.xml
- Pacote do projeto
- Resources
 - drawable
 - mipmap
 - layout
 - values
 - <https://developer.android.com/guide/topics/resources/providing-resources.html>
- Classe R
- APIs de compatibilidade
- Java e o Android
- Kotlin
- Compilando e executando

AndroidManifest.xml

- Informações “públicas” de aplicação Android
 - Todas configurações para ela executar
- Vários recursos de um App estão declarados dentro deste arquivo
 - Telas (Activities)
 - E alguns atributos
 - Serviços em background
 - Capacidades do aplicativo
 - Permissões do aplicativo



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.diego.meuapp">

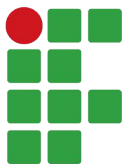
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

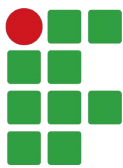
        <activity android:name=".OutraActivity" />
        <service android:name=".AlgumServico" />
    </application>

</manifest>
```



Recursos

- Tudo o que fica na pasta res do projeto
 - Arquivos ficam agrupados por tipo/função
 - Recursos gráficos em geral
 - PNG, JPG, XML vetoriais, ...
 - Ícones da aplicação
 - Layout de telas ← XML
 - Configuração de menus ← XML
 - Strings da aplicação ← XML
 - Temas, cores e estilos ← XML
 - ...



Classe R

- Classe Java gerada automaticamente
 - Faz a ligação entre os recursos existentes no diretório res e o código em Java/Kotlin
- Acesso a recursos:
 - R.layout.activity_main
 - Arquivo **res/layout/activity_main.xml**
 - R.string.nome
 - Tag “**string**” no arquivo **res/values/string.xml**
 - R.drawable.fundo ← desenhável
 - Arquivo **res/drawable/fundo.jpg**

Inflação de Componentes

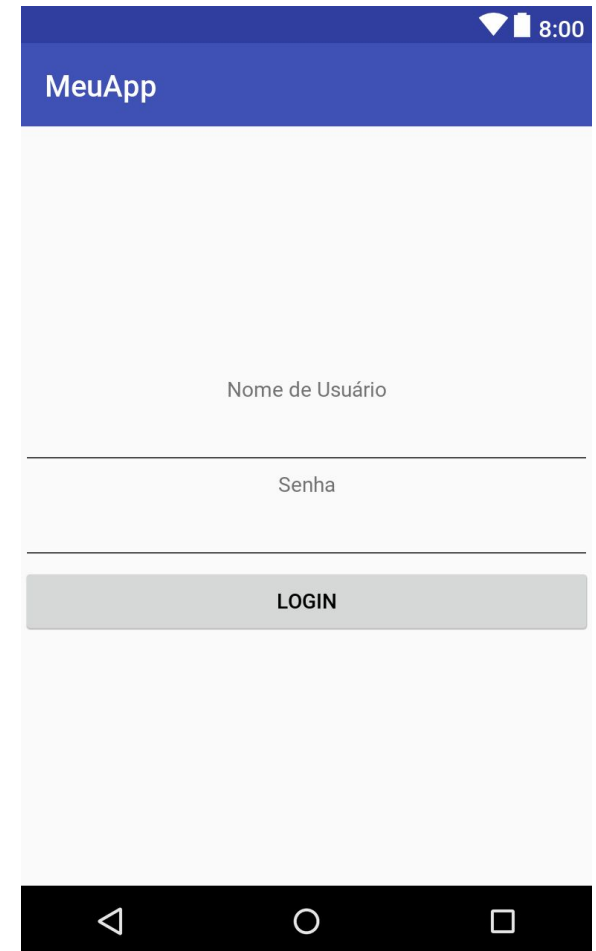
- activity_main.xml
 - XML
 - Editor gráfico
- MainActivity.java
 - Vínculo com classe R
 - `setContentView(R.layout.activity_main);`
 - `findViewById(...);`

Mais comum quando usando Java



Prática - Tela de Login

- Altere o `activity_main.xml`
 - Deixe com este aspecto →
- Dicas:
 - `LinearLayout`
 - `gravity` → `center`
 - `textAlignment` → `center`




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:gravity="center"
    android:orientation="vertical"
    tools:context="com.example.diego.meuapp.MainActivity">
```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Nome de Usuário"
    android:textAlignment="center" />
```

```
<EditText
    android:id="@+id/tUsuario"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAlignment="center" />
```

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Senha"
    android:textAlignment="center" />
```

```
<EditText
    android:id="@+id/tSenha"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:textAlignment="center" />
```

```
<Button
    android:id="@+id/btLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Login"
    android:textAlignment="center" />
```

```
</LinearLayout>
```

MainActivity.java

- Durante a criação da tela, precisamos:
 - Informar qual o XML de **layout** a ser carregado
 - Veio feito na criação do projeto
 - **Inflar** componentes que precisamos manipular
 - Os dois campos de texto e o botão de Login
 - **Inflar: Transformar XML em um objeto Java**
 - Ficar aguardando por cliques no **botão** de Login
 - Ao ser clicado
 - Ler os dados dos campos de texto
 - Validar Login e dar uma resposta ao usuário

```
package com.example.diego.meuapp;
```

```
import ...
```

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
```

```
    private TextView tUsuario;
```

```
    private TextView tSenha;
```

```
    private TextView btLogin;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    tUsuario = findViewById(R.id.tUsuario);
```

```
    tSenha = findViewById(R.id.tSenha);
```

```
    btLogin = findViewById(R.id.btLogin);
```

```
    btLogin.setOnClickListener(this);
```

```
}
```

```
@Override
```

```
public void onClick(View view) {
```

```
    if (view == btLogin) {
```

```
        String usuario = tUsuario.getText().toString();
```

```
        String senha = tSenha.getText().toString();
```

```
        String mensagem = "Login " + (loginValido(usuario, senha) ? "VÁLIDO" : "INVÁLIDO");
```

```
        Toast.makeText(this, mensagem, Toast.LENGTH_SHORT).show();
```

```
    }
```

```
}
```

```
private boolean loginValido(String usuario, String senha) {
```

```
    return usuario.equals("diego") && senha.equals("123");
```

```
}
```

```
}
```