****

*Olá, tudo bem com vocês?*

*Espero que estejam bem, muitos aqui já devem me conhecer, mas para aqueles que não me conhecem, eu sou a Ana!*

*Com um imenso prazer, eu compartilharei TODOS os meus resumos, anotações e dicas baseados nas aulas do Bootcamp everis Kotlin Developer / Docs / Comunidades / Artigos / Vídeos / Plataformas de estudos / outros, para agregar o conhecimento de todos, inclusive o meu com o Feedback de vocês!*

*Cabe ressaltar que, assim como muitos, iniciei meus estudos de programação este ano e sei o quão é difícil se orientar nesse novo “mundão” de Dev’s. Pensando nisso, quero ajudá-los a se direcionarem no Bootcamp e nesta incrível e desafiadora jornada!*

***Obs.: Disponibilizarei os resumos em doses fracionadas.***

*Espero que vocês gostem!*

*Ass.: Ana Guerra*

**Introdução ao tema:**

É extremamente necessário conhecer a área da tecnologia que você deseja se aprofundar. Neste caso, é importante que aos poucos você aprimore o seu conhecimento e este documento diz respeito a base principal para iniciar a sua jornada.

**Como baixar o que você precisa:**

Inicialmente, caso se interesse por fazer Download da ferramenta (IDE) utilizada pela instrutora neste módulo, siga o seguinte vídeo passo a passo:

Intellij IDEA e Java: <https://www.youtube.com/watch?v=Ag0uQDMTxMI>

**Como participar da aula sem baixar essa ferramenta?**

Agora não é necessário você ter o Intellij IDEA em sua máquina. Entretanto, ao decorrer do Bootcamp será extremamente necessário fazer o download do tópico anterior. Por enquanto, use o Kotlin por aqui:

Kotlin online: <https://play.kotlinlang.org/>

**Por que baixar uma IDE e o que é isso?**

Para ter muito mais conhecimento do tema e do módulo do mesmo Bootcamp, “ Lógica Essencial da Programação”, acesse o meu artigo na plataforma da Digital Innovation One:

<https://web.digitalinnovation.one/articles/vem-ver-a-base-da-programacao?back=%2Farticles&page=1&order=oldest>

**O que você aprenderá no módulo “*Projetando o Primeiro Aplicativo Android Usando Kotlin”*?**

1. Noções Básicas do App:
   1. Introdução:

. Guia do desenvolvedor;

. Anotação importante.

* 1. Criar Seu Primeiro Projeto Kotlin;

. Compreendendo a parte básica da estrutura do código exemplificado acima;

. Ampliando o Programa;

. Como lidar com erros;

. Anotações importantes.

1. Criar Seu Primeiro Aplicativo Android:
   1. Visão Geral:

. Criar o seu primeiro aplicativo simples;

. Anotação Importante.

* 1. Criar um Projeto para o Android:

. Criando um projeto Android;

. Executar seu app.

* 1. Criar um Interface de usuário simples:

. Pontos importantes para ressaltar;

. Abrir o Layout Editor;

. Adicionar uma caixa de texto para o projeto e diferente da original;

. Adicionar um botão;

. Alterar as Strings da Interface do Usuário;

. Atribuir as Keys para cada Widget;

. Tornar o tamanho da caixa de texto flexível.

* 1. Iniciar outra Atividade:

. Responder ao Botão;

. Explicando cada palavra;

. Criar uma Intent;

. Passo a passo o que fora feito para criar a Intent;

. Criar a segunda Atividade;

. Adicionar uma visualização de texto para a segunda Atividade;

. Exibir a mensagem;

. Adicionar a navegação para voltar à página anterior (Botão Up).

1. Fundamentos do Aplicativo:

. Tópicos.

1. Componentes de um aplicativo:
   1. Atividade (activity);
   2. Serviços (services);
   3. Broadcast Receivers;
   4. Provedores de conteúdo (ContentProvider).
   5. Ativação de componentes;



**Noções Básicas do App**

**Introdução**

**- Guia do desenvolvedor:**

Estes documentos ensinam a criar app Android usando APIs no framework do Android e outras bibliotecas.

Para saber mais sobre o Desenvolvimento Android, através de outros recursos, acesse:

Codelabs: São tutoriais curtos e individualizados que abrangem tópicos específicos. Exemplo: Como criar o seu primeiro app passo a passo;

<https://codelabs.developers.google.com/?cat=Android>

Cursos: Treinamentos mais longos que as Codelabs que ensinam a criação de apps Android;

<https://developer.android.com/courses>

Treinamentos on-line: Treinamentos longos ou curtos a respeito da criação de apps Android;

<https://www.udacity.com/course/developing-android-apps-with-kotlin--ud9012>

https://developer.android.com/guide#videos

**- Anotação Importante:**

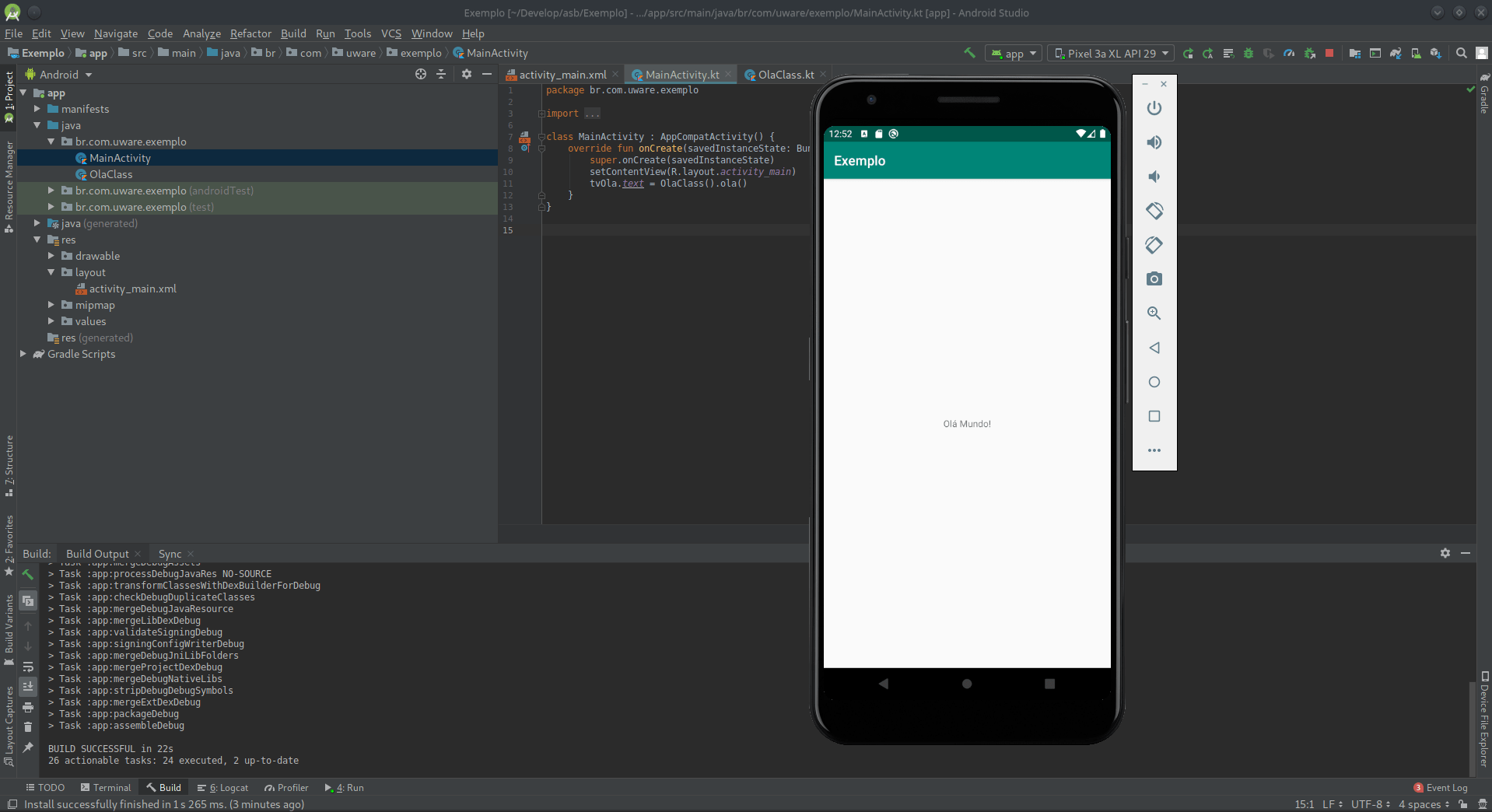
Nunca se baseie em apenas um artigo, documento ou qualquer outro tipo de conteúdo teórico. É importante que sempre leia a documentação básica do Android e mantenha-se também atualizado(a) no Youtube e nas comunidades.

* Exemplo de leitura importante:

<https://developer.android.com/guide>

**Obs.:**

* APIs (Application Programming Interface–Interface de programação de Aplicativos): É um conjunto de rotinas e padrões de programação para acesso a um aplicativo de Software ou plataforma na Web.
* Software: É um conjunto de lógica de um computador;
* Computador: É toda a máquina que tem uma entrada, processamento e saída de um dado;
* Framework: É um conjunto de ferramentas e/ou códigos prontos que auxiliam o programador a trabalhar com uma determinada linguagem.



**Cor destaque para termos importantes**

**Criar Seu Primeiro Projeto em Kotlin**

Esta etapa é extremamente importante para prosseguir a criação de um app. Ou seja, ela é a base para uma melhor interpretação do que está acontecendo no programa;

- Primeiro passos:

Caso não tenha o Intellij IDEA instalado em seu computador, opte por instalar ou usar o Kotlin direto do navegador, links a seguir:

1) <https://www.youtube.com/watch?v=uIU6symX2H8>

2) <https://developer.android.com/training/kotlinplayground>

- Segundo passo:

1. Digitar o seguinte código na tela, caso esteja utilizando o programa WEB ou já saiba criar um projeto e arquivo no Intellij IDEA:



1. Para executar o programa, basta encontrar o ícone de “play”, como o indicado abaixo, e esperar o computador compilar (ler o que você pediu para ele fazer). Ou seja, para ver o que o computador nos retornará.



1. Quando o computador finalizar a leitura, aparecerá na sua tela a seguinte “resposta”:



1. Ao alterar o “Hello, world!” para “Happy Birthday!” e depois repetir o processo de compilação, o computador trará como resultado:



- Compreendendo a parte básica da estrutura do código exemplificado acima:

É importante ressaltar que cada detalhe no código é extremamente importante para a compilação ocorrer de forma correta. Ou seja, sem erros e traga como resultado o que você quer.

1. *fun* é uma palavra da linguagem de programação que significa função. A fun pode ser nomeada como palavra-chave ou palavra reservada;
2. *main* é o nome da função. As funções têm nomes para que seja possível diferenciá-las uma das outras. A fun main é a função principal do Kotlin, tudo o que você fizer neste programa é necessário chamar na main;
3. *( )* Uma função SEMPRE possuirá em seu seguimento os parênteses. Dentro deles são armazenadas as informações a serem usadas pela função, essa entrada é chamada de argumento ou args.
4. *{}* Em uma função, não é sempre, mas teremos em muitos casos um par de chaves após os parênteses. Na parte interna deles, terá um bloco de códigos ou um código que denominou uma tarefa a esta função;
5. *println* diz ao sistema o que deverá ser exibido após a execução;
6. “ ” Representa a parte do texto, podendo conter internamente letras, caracteres, palavras, números e outros;

- Ampliando o Programa:

Algumas informações serão ditas com base os exemplos dados no tópico anterior;

1. Uma função pode ter quantas linhas de instruções desejar;
2. A função println() pode ser repetida quantas vezes quiser dentro de outra função;

- Como lidar com erros:

Cometer alguns erros durante a programação é normal e a maioria dos ambientes já estão preparados para auxiliar o programador a corrigir o seu erro;

1° O Intellij IDEA indica o erro usando as palavras em vermelho;

2° No painel de saída, quando ocorrer algum erro, o Intellij IDEA exibirá um ícone de exclamação e o erro em seguida;

- Anotações Importantes:

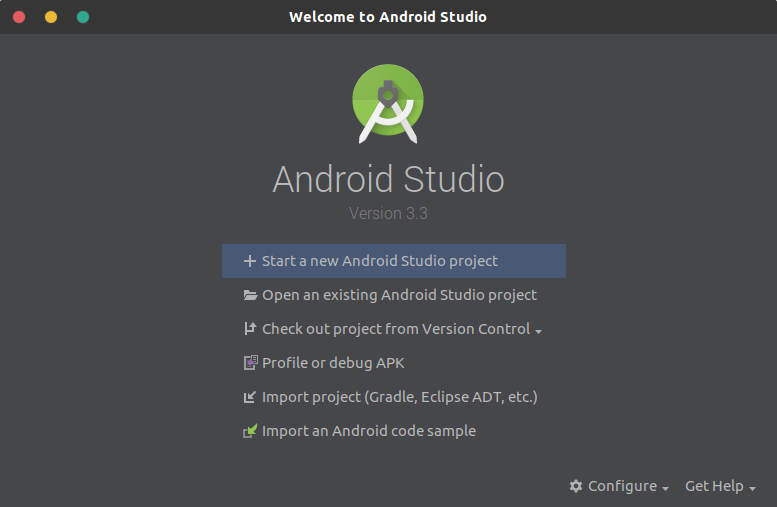
Lembre-se que assim como na língua Portuguesa “existem n maneiras” de falar uma mesma frase, em programação também “existem n maneiras” de chegar em uma mesma solução. Este conceito refere-se a um logaritmo;

1. *println ()* executa um código a baixo do outro, por conta do ln (line);
2. *print ()* executa um código do lado do outro, agrupado ou não;
3. O *“\n”* é utilizado *durante um texto* para quebrar linha;

É importante lembrar que, quando precisar, pode usar o println (“ ”) ou print (“ ”) para exibir uma linha vazia ou espaço, depende de como usar;

5º Caso deseje aumentar o seu vocabulário em Kotlin, use o seguinte Link para estudar:

https://developer.android.com/courses/android-basics-kotlin/android-basics-kotlin-vocab

****

**Criar Seu Primeiro App no Android**

**Visão Geral:**

- Criar o seu Primeiro App Simples:

*Objetivo:* A princípio o objetivo é criar um projeto “Hello, World!” com o Android Studio e executá-lo. Depois, criaremos uma nova interface para o aplicativo que receberá entradas do usuário e mudará para uma nova tela no app para exibir a saída.

*Os Apps oferecem vários pontos de entrada*: Os apps são criados com uma combinação de componentes que podem ser invocados individualmente (lembre-se da construção da IU do app no layout);

Exemplo: Uma *atividade* é uma IU.

* *Atividade Principal:* Ocorre quando o usuário toca no ícone, abre a aba de notificação que direciona para esta atividade ou vai para esta atividade por outro app;
* *Broadcast receivers e serviços:* Permitem o uso do app em segundo plano sem a ativação da IU;

*Os Apps se adaptam a diferentes dispositivos*: O Android permite criar recursos diferentes para tamanhos de telas variados. Ou seja, o sistema é capaz de determinar qual é o tamanho de tela do dispositivo atual.

O seu app pode ter uma série de requisições, como determinar qual versão sistema e hardware do dispositivo é necessário para utilizar o aplicativo, quais os recursos que o dispositivo precisa conter para o app funcionar, caso não tenha o app pode funcionar limitando algumas funções e assim por diante.

É importante lembrar que o Google Play é quem não permitirá o usuário instalar o app caso não tenha os pré-requisitos estabelecidos pelos desenvolvedores.

- Anotação Importante:

Caso queira ver algumas aulas para fixar o conteúdo na memória:

<https://developer.android.com/courses/android-basics-kotlin/unit-1>

**Criar um projeto para Android:**

Esta etapa diz respeito a criação de um projeto após a instalação do Android Studio. Caso não tenha o Android Studio segue o link para o download:

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

- Criando um projeto no Android:

Nesta etapa deve-se seguir alguns passos para a criação do projeto.

1. Na janela Welcome to Android Studio selecione a opção Create New Project;
2. Na janela Select a Project Template selecione a opção Empty Activity (Tela com o Layout vazio) e clique Next;
3. Na janela Configure your Projec, faça o seguinte:

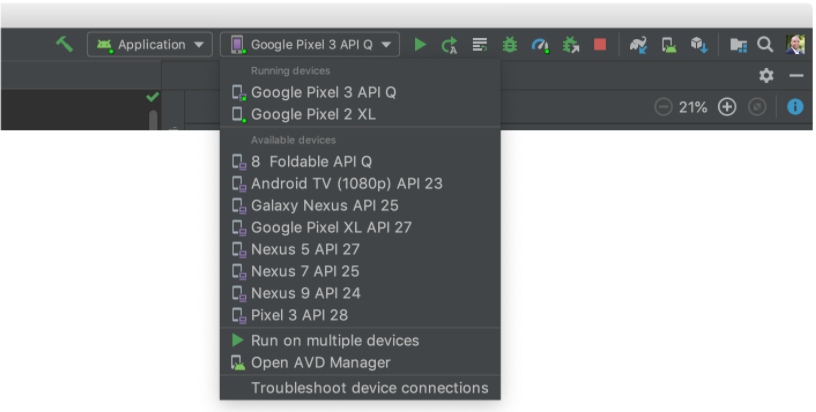
* Digite “My first app” no campo Name;
* Digite “com.example.myfirstapp” no campo Package name;
* Selecione Kotlin no menu suspenso Language;
* Selecione uma versão qualquer do Android que deseja para ser compatível ao seu app, no campo Minimum SDK;

*Obs.: Caso deseje a biblioteca compatível ao app, marque a caixa de seleção Use Legacy android.support libraries;*

1. Clique em Finish para encerrar as alterações;

**Executar seu app:**

Há duas formas de executar um app criado no Android Studio, através de um Dispositivo Real ou um Emulador, o processamento é o mesmo e a única diferença entre eles está relacionada com a capacidade do computador. Por isso, eu indico utilizar o Dispositivo Real com um cabo USB;



- Executar em um DISPOSITIVO REAL:

1. Conecte o dispositivo a sua máquina por um cabo USB;

Em seu celular, siga os seguintes passos do vídeo:

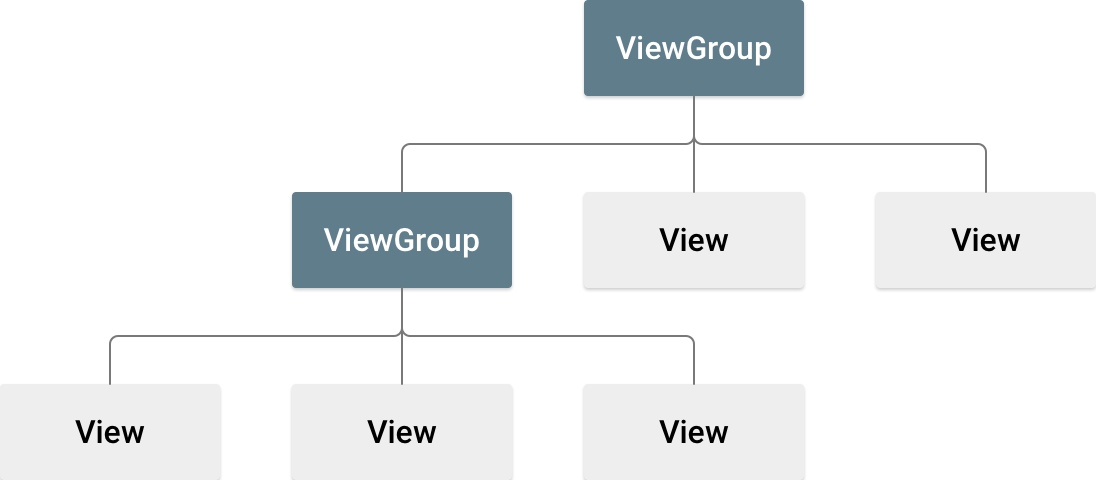
https://www.youtube.com/watch?v=0bMC19SEy4U

- Executar em um EMULADOR:

No Android Studio, crie um Dispositivo Virtual Android (AVD) em open avd Manager;

Selecione as opções que deseja, descubra.

**Criar uma IU simples:**



Nesta etapa você verá como usar o Layout Editor do Android Studio, onde nós, os desenvolvedores, criamos a parte da Interface do Usuário;

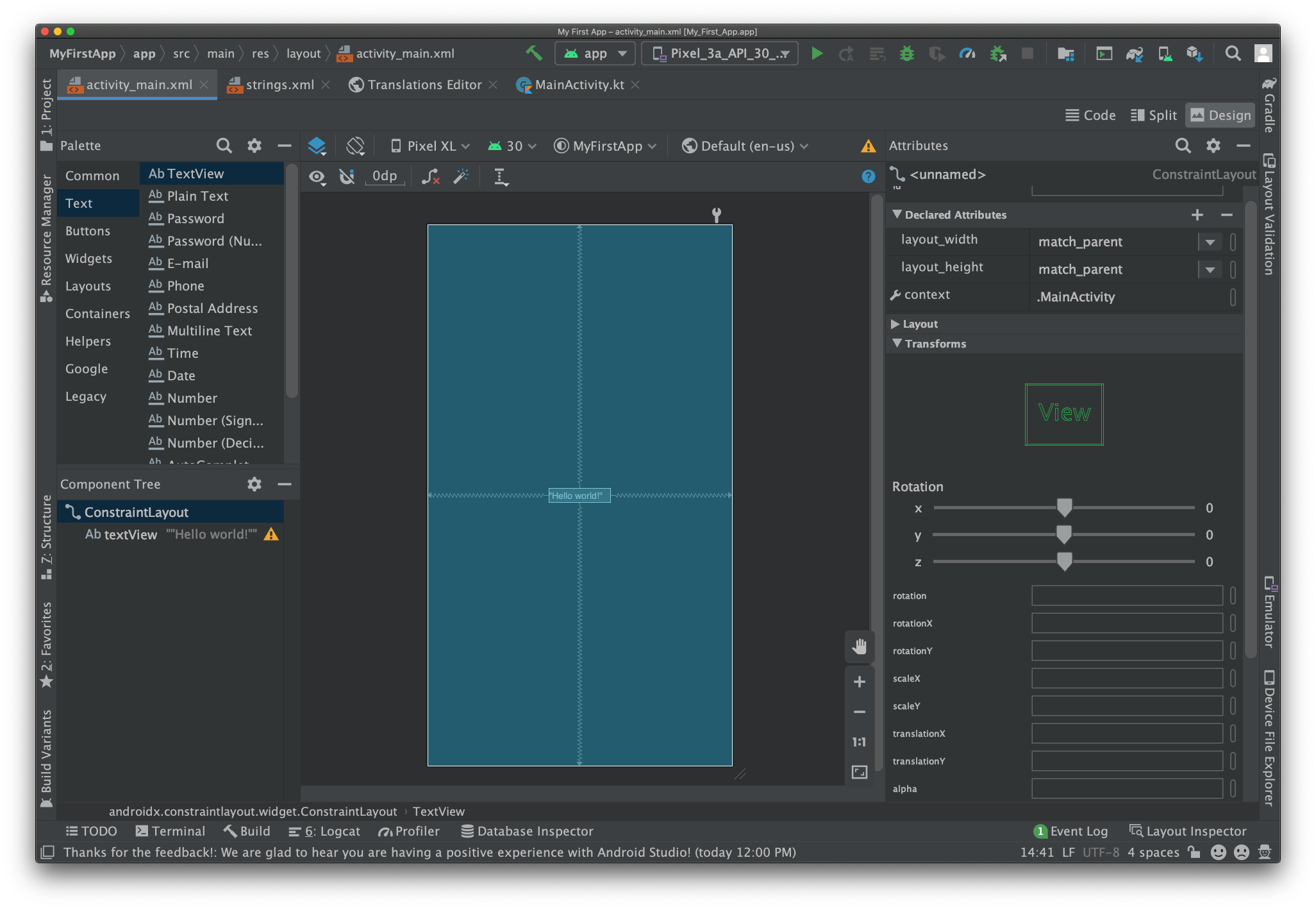
- Pontos IMPORTANTES para ressaltar:

* Para criar uma interface de “qualidade” deve-se usar o conceito de hierarquia de layouts e widgets;
* Os layouts são objetos ViewGroup. O ViewGroup são containers que controlam como as visualizações filhas são posicionadas na tela;
* Os Widgets são objetos View. O view são componentes da IU (interface do usuário)
* Pense: um objeto é uma view, podendo ser um botão, uma caixa de texto. Então, um ViewGroup é um conjunto de objetos, ou seja, é o que contém as views;
* É importante ter conhecimento que a maior parte da IU é escrita com um vocabulário XML (é importante, caso esteja interessado em evoluir mais, pesquisar cursos sobre o tema);

- Abrir o Layout Editor:

Esses passos de como utilizar o Android Studio e localizar cada ferramenta que você precisa para programar, é extremamente importante! Foque em memorizar os conceitos para enriquecer o seu vocabulário e memorizar, também, outras formas que eu não consigo falar aqui porque é muita coisa (vocês têm que pesquisar) para melhor executar o que deseja. Enfim, siga os próximos passos:

1. Na Janela à esquerda do Android Studio (em seu local de origem), abra os arquivos: app> res > layout > activity > acitivity\_main.xml
2. Em seguida, clique em Design no canto superior direito;
3. Investigue as seguintes ferramentas que possuem estes ícones:     
4. Component Tree: diz respeito a hierarquia das Views, que neste caso possui apenas uma view, o TextView. Ou seja, apenas um objeto;
5. ConstraintLayout (layouts aninhados): É um layout que define a hierarquia das views e do container. Este tipo de Layout aumenta o tempo para desenhar a IU;
6. LinearLayout: É um layout que define a hierarquia linear, ou seja, apenas horizontal e/ou apenas vertical;



- Adicionar uma ***caixa de texto*** para o projeto e diferente da original:

Siga as seguintes etapas para adicionar uma caixa de texto;

1. Deletar a caixa de texto já existente. Ou seja, de um delet na caixa “TextView”;
2. Abrir o painel Palette e nele estará localizado as possíveis Views do layout;
3. Adicionar um EditText do tipo PlainText(que é uma View é um widget), a partir do Palette;

- Adicionar um Botão (outro objeto ao nosso ViewGroup):

Siga as seguintes etapas para adicionar um botão ao layout:

1. Abra o Painel Palette e selecione o Button;
2. Arraste o widget button para o lado direito da caixa de texto (PlainText);

- Alterar as Strings da IU:

Iniciaremos a modificação da Interface do Usuário em relação ao título e nome dos Widgets (Button e o PlainText). Siga passo a passo para a melhor compreensão:

1. Seguir a sequência: app > res > values > strings.xml
2. Lembrar que os arquivos com o final xml são escritos com o vocabulário XML;
3. Clique no "open editor” na parte superior da janela. Essa ação abrirá o Translations Editor que mantém as Strings atualizadas e fáceis de encontrar;
4. Encontre o sinal de operação matemática soma para abrir o add Key criar uma nova String;
5. Key é o id do texto. Ou seja, aquele que não aparecerá na IU (Estrutura: nome\_nome);
6. Default Value é a parte que aparecerá na IU (nome);
7. Clique em Ok.

**Obs.:** Caso queira seguir a minha nomeação das Strings, para o Botão coloque a Key button\_send e para o Default Value, coloque “Send”;

Ainda na parte de Strings, analise o código escrito em xml:

<string name="edit\_message">Texto de dica</string>

1. A key está sendo representada em verde;
2. O Default Text está sendo representado em azul;
3. E em amarelo é o tipo de representação.

- Atribuir as Keys para cada Widget:

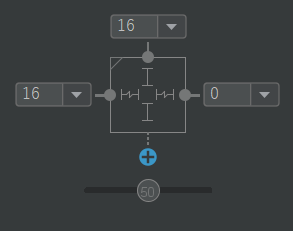
Siga passo a passo para sair do arquivo strings.xml e ir para a activity\_main:

1. Clique na caixa de texto do Layout;
2. No canto direito e superior, abra a aba de Attributes;
3. Localize pela lupa na parte superior do Attributes;
4. Pesquise pela palavra Text, atualmente definida como name e exclua o valor;
5. Para o texto mudar conforme a digitação do usuário, pesquise na lupa a palavra hint;

- Tornar o tamanho da caixa de texto flexível:

Este tópico está relacionado à ConstraintLayout (conceito de widgets aninhados).

1. Com o objetivo de criar um layout responsivo ao diferentes tamanhos de tela e as rotações do dispositivo;
2. Lembre-se de um dos ícones que eu comentei no texto, vou deixar você exercitar a memória para lembrar e quando achá-lo rotaciona a tela para testar o “aninhamento” dos widgets e o ContraintLayout do viewGroup;
3. Vá em Attributes no Design e localize o Layout, lá estará contido o Constraint Widget do widget que você deseja alterar e se atente a estas figuras:



Na imagem, note que:

- O reto (fixed) não altera a altura do bloco;

- O “quebrado” permite a alteração do bloco;

**Iniciar outra atividade:**

Ao finalizar o tópico anterior, você terá em sua tela apenas dois widgets sem resposta, um botão e uma caixa de texto. Neste novo tópico, iremos aprender a criar um código básico (PRESTE ATENÇÃO É A BASE PARA COMEÇAR APRENDER) na MainActivity para iniciar uma atividade apresentando uma mensagem do usuário;

- Responder ao botão:

* Para dar uma “responsividade ao botão” é necessário criar um método. Os métodos são as funções que iniciam ação a um botão;
* Este método (ou função, mas é melhor chamar de método) é inserido na classe MainActivity que inicializa as ações da atividade principal. Para realizar o que fora dito, siga os seguintes passos:

1. No arquivo: app > java > com.example.myfirstapp > **MainActivity**;
2. Adicione à ela o método sendMessage () ele serve para enviar uma mensagem logo que pressionar o button;
3. Seguindo a estrutura: fun sendMessage(view: View)

. Explicando cada palavra:

* fun nós já vimos anteriormente;
* sendMessage() método que inicia a reação do botão;

1. (view: View) é a propriedade, o view é argumento do método e o View é o objeto;
2. O :Unit está implícito, não é necessário ser inserido. Diz respeito a uma função sem retorno.
3. Aparecerá um erro na palavra View, este erro existe porque não importamos a biblioteca da activity\_main.xml;
4. Para solucioná-lo, aperte alt+enter para importar a biblioteca e pronto o erro será solucionado;
5. A partir daqui teremos que acessar a **activity\_main.xml** para chamar o método do botão;
6. Dica: procure na tela MainActivity o arquivo que deseja ir, no caso o activity\_main.xml e clique na palavra pressione ctrl+botão esquerdo do mouse;
7. Vá na parte do Design > Attributes > na lupa > procure por onClick > selecione a opção do método criado lá na MainActivity;
8. Agora ao clicar no botão, o sistema chamará o método sendMessage();

- Criar uma Intent (PARTE MAIS COMPLICADA):

* O Intent representa a intenção do app fazer algo, que neste caso é mostrar uma nova atividade;
* O Intent oferece a opção de vincular componentes separados, como duas atividades;

*Lembrar: atividade é uma tela de Interface do usuário;*

* O Intent serve para diversas finalidades, mas nesse caso iremos usar para conectar telas;

-Estrutura de um Intent (IMPORTANTE):

val intent = Intent (context: this, nomeDaTela2::class.java)

* O objeto intent representa a “nomeação de uma segunda tela”;
* A classe Intent tem uma estrutura padrão Intent (context, tela::class.java)
* A classe Intent recebe o parâmetro context do tipo this;
* A classe Intent recebe o parâmetro nomeDaTela2 do tipo class.java;

const val *EXTRA\_MESSAGE* = "com.example.androidcriandomeuprimeiroapp.MESSAGE"  
  
class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 }  
 fun sendMessage(view: View): Unit{  
  
 val editText = findViewById<EditText>(R.id.*text*)  
 val message = editText.*text*.toString()  
 val intent = Intent(this, DisplayMessageActivity::class.java)  
 putExtra(*EXTRA\_MESSAGE*, message)  
 }  
 startActivity(intent)  
}

1. Note que após ter declarado a intent iniciamos o putExtra();
2. O método putExtra() adiciona o valor do EditText à Intent;
3. Note que o EditText foi transformado em message;
4. o startActivity(intent) inicia a nova tela;



const val *EXTRA\_MESSAGE* = "com.example.androidcriandomeuprimeiroapp.MESSAGE"  
  
class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 }  
 fun sendMessage(view: View): Unit{  
  
 val editText = findViewById<EditText>(R.id.editText)  
 val message = editText.*text*.toString()  
 val intent = Intent(this, DisplayMessageActivity::class.java).apply{  
 putExtra(*EXTRA\_MESSAGE*, message)

}  
 }  
 startActivity(intent)  
}

- Passo a passo o que fora feito para criar a Intent

1. Declaramos uma palavra reservada IMUTÁVEL (const val) que como uma chave de acesso para recuperar o valor de um texto;
2. Na tela de criação (onCreat()) iniciaremos um método que dará uma reação ao clique do botão;
3. Adicionaremos o objeto editText que representa a caixa de texto;
4. O valor do editText será a invocação de ser Id;
5. Criaremos um objeto do tipo mensagem, que será o resultado de quando clicarmos no botão e será aquilo o que aparecerá na segunda tela;
6. Esse objeto mensagem receberá o valor que diz respeito à identificação de um texto que fora alterado na atividade principal. Ou seja, o editText em forma de texto(.text) e transformado para String (.toString);
7. Criaremos um outro objeto que diz respeito a criação de uma segunda atividade, o intent.
8. O intent receberá a class Intent() para identificar em qual contexto parte e para onde vai (os parâmetros);
9. Depois adicionaremos uma class putExtra() para dizer o que irá acontecer na próxima tela, que no caso será uma EXTRA\_MENSAGEM com a estrutura de texto desejado na tela principal e do tipo message porque fora capturada na tela principal pelo editText;
10. Iniciaremos a nova activity logo que eu fechar a reação do botão (sendMessage());

- Criar a segunda atividade:

Siga os passos a seguir para criar uma nova atividade:

1. Vá na ferramenta Projeto > app > clique com o botão direito do mouse > New> Activity > EmptyActivity;
2. Dê o nome para essa nova atividade igual ao que você declarou na class Intent na atividade principal. No caso é: DisplayMessageActivity e clique em Finish;

Observe que o Android Studio tomou algumas ações automaticamente:

1. Criou um arquivo DisplayMessageActivity que é semelhante a MainActivity;
2. Criou um layout respectivo a segunda atividade: activity\_display\_message\_activity.xml;
3. Se necessário, adiciona também no AndroidManifest o elemento <activity>;

- Adicionar uma visualização de texto para a segunda atividade:

Siga os passos a seguir para criar um widget que responderá à ação do botão;

1. Abra a ferramenta de Projetos;
2. Vá para app > res > layout > activity\_display\_message.xml
3. Clique no Design e depois selecione o Enable Autoconnection to Parent (ícone de ferradura);
4. Vá em Palette e adicione um TextView que será a resposta do texto inserido na atividade principal;
5. Altere o textAppearance caso queira:

*Vá em Attributes > Common Attributes > textAppearance.*

- Exibir a mensagem:

Nesta etapa modificaremos o EditText inserido para obter a saída da mensagem escrita na atividade principal. Siga os seguintes passos:

1. Vá para DisplayMessageActivity, não indicarei onde está para você ir atrás e treinar;
2. Adicione o seguinte código ao método onCreate() que irei explicar cada detalhe;

package com.example.androidcriandomeuprimeiroapp  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.TextView  
  
class DisplayMessageAcitivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_display\_message\_acitivity*)  
  
 val message = *intent*.getStringExtra(*EXTRA\_MESSAGE*)  
  
 val textView = findViewById<TextView>(R.id.*textView*).*apply***{**  
*text* = message  
 **}**  
}  
}

Estes passos servem para adicionar o código no método onCreate( ):

1. Criar o objeto message (note que podemos repetir objetos ao mudar de atividade);
2. Chame no objeto message a intent da atividade principal que fornece a reação do botão. Ou seja, o texto que aparecerá nele, como fora definido na MainActivity;
3. Chame a classe getStringExtra (mensagem da atividade principal) e de a propriedade de texto do objeto determinado na atividade principal;
4. Crie o objeto textView que retornará a reação do botão e aparecerá na segunda atividade;
5. Chame pela view que está inserida no segundo layout pelo findViewById<tipo de view>(lolcalização.key.tipoDeTexto) e a sua aplicação (.apply) nele será o text igual a message da atividade principal;

- Adicionar a navegação para voltar à página anterior (**Botão Up**):

Todas as telas que não são a atividade principal precisam retornar para o seu “pai lógico”. Ou seja, uma nova hierarquia de navegação. Para adicionar o botão Up na barra de apps, siga os passos a seguir:

1. *É necessário declarar qual atividade é o pai lógico no arquivo AndroidManifest.xml;*
2. Encontre no AndroidManifest a tag <activity> para o DisplayMessageActivity e substitua pela imagem a seguir:
3. Note a parte em negrito, ela será nomeada como meta-data e indicaremos o nome sendo do tipo android.support.PARENT\_ACTIVITY;
4. Valor dele sendo como a atividade principal igual a .MainActivity.

<activity android:name=".DisplayMessageActivity" />  
<activity android:name=".MainActivity">  
 ***<meta-data  
 android:name="android.support.PARENT\_ACTIVITY"  
 android:value=".MainActivity" >***

**</meta-data>**  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />

**Fundamentos do App**

* Antes de prosseguirmos é importante ter em mente alguns conceitos.
* Os aplicativos Android são desenvolvidos em Java, Kotlin e outras linguagens C++;
* Todos os aplicativos desenvolvidos para Android utilizam os arquivos de APK.
* Arquivos de APK (Android Package), são aqueles que possuem o sufixo .apk e contém dados e recursos suficientes para rodar o aplicativo.
* Cada aplicativo android é ativado em sua Sandbox;
* A Sandbox ela possui alguns recursos de segurança:
  + - * + O Sistema Operacional Android possui o seu núcleo estruturado pelo Kernel Linux. Este núcleo possibilita que cada aplicativo se cadastre no dispositivo como se fosse um usuário diferente;
        + Cada aplicativo possui um código de usuário do Linux, o aplicativo não tem conhecimento desse código, mas o usuário possui e somente ele consegue acessar os arquivos do mesmo;
        + O código de um aplicativo é isolado de outros aplicativos, essa característica diz respeito a cada aplicativo possui uma máquina virtual;
* *Princípio do privilégio mínimo:* todos os aplicativos possuem acesso apenas em arquivos que precisam para executar o seu processo. Este “conceito”, viabiliza uma maior segurança, onde, uma vez que o aplicativo não tem acesso a outros arquivos os dados não serão compartilhados sem permissão;
* Por mais que o aplicativo não tenha acesso para acessar os dados, existe uma maneira de um aplicativo compartilhar dados com outros aplicativos e acessar serviços do sistema:
  + - * + Dois aplicativos podem compartilhar o mesmo código do Linux. Ou seja, acessar arquivos um do outro;
        + Dois aplicativos podem ser executados ao mesmo tempo e terem acesso ao mesmo VM (Virtual Machine – Máquina Virtual) do Linux;
        + Um aplicativo pode solicitar permissão de acesso a dados do dispositivo, como contatos do usuário, mensagens, SMS, armazenamento no SD;

A partir de agora, os próximos conceitos, aprenderemos melhor como funciona o que foi dito anteriormente.

**- Componentes de aplicativo:**

Os componentes do aplicativo são os blocos de construção de um app Android. Cada componente é um ponto de entrada por meio onde o usuário ou o sistema pode acessar o aplicativo. Aliás, um componente depende de outro.

Há quatro tipos de componentes de aplicativos (“O que o aplicativo possui?”):

1. Atividades;
2. Serviços;
3. Broadcast Receivers;
4. Provedores de Conteúdo;
5. **Atividades(Activity):**

*Obs.: Lembrando que a Atividade Principal é a tela inicial do app*

* Uma atividade é o ponto de interação do usuário com o aplicativo;
* O conjunto de atividades forma um app;
* A interface pode interagir com o usuário de diversas formas, como:
  + - * + Hospedagem ativa, que forneça informações, na tela, ao que realmente interessa para o usuário;
        + Atividades interrompidas (onPause())que podem ser reiniciadas por outra atividade. Podendo ter um retorno do ponto onde o usuário parou;
        + Toda a atividade (activity) criada, após a primária é uma subclass da class Atividade Principal. Ou seja, o que acontece na Atividade Principal diz respeito ao resultado que aparecerá na secundária;

1. **Serviços (Service):**

* Os serviços são os responsáveis por carregar informações em segundo plano;
* Os serviços não precisam de uma Atividade para interagir com você, ele simplesmente não interage, ele apenas executa um processo sem você ver;
* Há dois tipos de serviços, mas antes vou citar alguns exemplos para associar melhor o que é um serviço:
  + - * + Quando estamos utilizando o Spotify para ouvir uma música, mas ao mesmo tempo queremos acessar o Instagram, a música não para. Ou seja, o serviço está executando um processo de tocar uma música, sem ter uma interface interagindo com você;
        + Quando estamos fazendo download de um aplicativo enquanto vamos abrir o Instagram e ouvir música no Spotify, assim por diante.
* Podemos notar, a partir dos exemplos citados, que os serviços são bastante importantes para um aplicativo;
* Existem tipos diferentes de serviços, vamos continuar no conceito dos exemplos para você ver a diferença entre os serviços:
  + - * + O usuário escuta uma música no Spotify, mas imagina só se, de repente, o acesso a troca de música ou ao retorno por meio de uma notificação? Isso prejudicaria a boa interação do usuário com o dispositivo, certo? Então, esse é o primeiro tipo de serviço, manter o processo de execução sempre ativo;
        + No caso de um download, o serviço consegue manter a música tocando e informar o usuário de uma possível falta de armazenamento para continuar o processo de download;

1. **Broadcast Receivers:**

* Note que a palavra Receivers ao contrário é semelhante à Services;
* É um componente que faz o Sistema entregar eventos ao aplicativo fora de fluxo;
* Não possui uma Interface do Usuário (IU);
* Pode criar uma notificação na barra de status;

*Exemplo: o uso de alarmes no celular, o aplicativo sabe que horas ele precisa enviar um Broadcast que avise o usuário aquilo que ele deseja, como pouca bateria;*

1. **Provedores de Conteúdo:**

* Gerenciam um conjunto de dados do aplicativo que você pode armazenar nos sistemas de arquivos, em banco de dados SQLite, na Web ou em qualquer local de armazenamento;
* Caso o provedor do conteúdo permita (lembre do que fora dito na introdução), os aplicativos podem consultar ou até modificar os dados, caso o provedor de conteúdo permita;
* O provedor de conteúdo é útil para ler e gravar dados privados no aplicativo e não compartilhados;
* O provedor de conteúdo é um ponto de entrada em um aplicativo para a publicação de itens de dados nomeados, identificados por um esquema de URI;
* O esquema de URI (Identificador Uniforme de Recursos), como diz o nome, é um identificador que nomeia o dado e auxilia o provedor de conteúdo organizar os dados em um namespace URI;
* Essa organização de dados permite que outros aplicativos consigam acessar os dados de uma forma mais eficaz e segura;
  + - * + Um dado dentro do “sistema” de URI não precisa ter o aplicativo, o qual pertence, em execução para ter as suas informações acessadas por outro dispositivo;
        + O aplicativo pode substituir a URI por uma imagem que esteja na área de transferência dos dados do aplicativo na URI correspondente, podendo ter controle do acesso de seus dados pelos provedores de conteúdo;

- Ativação de componentes:

* Apenas 3 dos 4 componentes são ativados: Atividade, Serviços e o Broadcast receivers;
* A ativação destes componentes funciona por meio de uma mensagem assíncrona;
* Uma mensagem assíncrona é aquela que foi enviada para um receptor, mas o receptor não irá recebê-la imediatamente. Ou seja, precisa de uma condição;
* Os 3 componentes vão ser ativados pela mensagem assíncrona chamada intent;
* As Intents, são mensagens assíncronas que vinculam componentes individuais uns aos outros no ambiente de execução;
* Resumindo, pense nos intents como mensageiros que solicitam uma ação de outros componentes pertencentes ao app ou não.
* É VÁLIDO lembrar que as duas palavras em chave neste tópico são: mensagem assíncrona!

- Na prática: Um objeto Intent é criado para definir uma mensagem de ativação de um componente específico. Os intents podem ser explícitos ou implícitos;

- A atuação das Intents:

1. Atividade (IU) e serviços: (atividade “secundária”): Os Intents definem a ação de executar (visualizar ou enviar) e especificar a URI dos dados usados na ação. Um Intent pode transmitir uma solicitação de uma Atividade para exibir uma imagem ou abrir uma página Web. Uma atividade, também pode retornar como resultado um intent para selecionar um dado e retorná-lo;
2. Broadcast Receivers (notificação na barra de tarefa): Os Intents atuam como indicadores em String da ação conhecida como nível baixo de bateria;

- Se somente a Atividade, os Serviços e o Broadcast Receivers recebem a ativação de seus componentes por Intent, como funciona com os Provedores de Conteúdo?

* Os Provedores de Conteúdo não são ativados por intents (mensagens assíncronas);
* Os Provedores de Conteúdo são ativados por uma solicitação de um ContentResolver;
* O ContentResolver (Resolvedor de Conteúdo) trata as mensagens entre os componentes e o Provedor de conteúdo de forma direta e segura;

- Há dois métodos de ativar cada tipo de componente (Atividade, Serviços, Broadcasts Receivers e Provedor de conteúdo):

1. A Atividade pode ser iniciada por uma Intent sendo passada a startActivity(), para retornar um resultado definitivo ou startActivityForResult() retornando um resultado quando for desejado;
2. Para iniciar um serviço, pode-se usar JobScheduler para programar ações futuras (nas versões posteriores ao Android 5.0), nas versões anteriores podemos usar um Intent passado a um startService() ou passando um Intent para o bindService();
3. Para iniciar uma transmissão passando um Intent para métodos como sendBroadcast(), sendOrderedBroadcast() ou sendStickyBroadcast();
4. Para iniciar um Provedor de conteúdo, chamamos um conteúdo query() em um ContentResolver;

**- O arquivo Manifesto:**

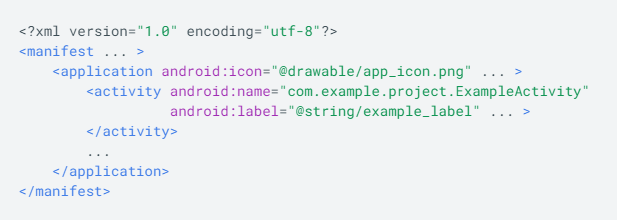
Para o sistema Android iniciar um aplicativo é necessário que ele leia o AndroidManifest.xml para reconhecer o componente do aplicativo;

O AndroidManifest faz n outras coisas além de declarar os componentes de um app:

* Identifica as permissões do usuário que o app precisa;
* Declara o mínimo de API que app precisa para funcionar;
* Declara os recursos de Hardware e Software que necessita;
* Declara as Bibliotecas API que necessita vincular;

- Declaração de componentes:

A principal função do AndroidManifest é informar os componentes do aplicativo para o sistema;



* O elemento <application> possui como atributos o android:icon que identifica o ícone do aplicativo;
* O elemento <activity> possui dois atributos o android:name que identifica o nome da classe qualificada da subclass de Activity e o atributo android:label que especifica a atividade visível ao usuário;

- Todos os componentes do aplicativo que usam os seguintes elementos precisam ser identificados no AndroidManifest:

1. Elementos <activity> para atividades;
2. Elementos <service> para serviços;
3. Elementos <receiver> para Broadcast Receivers;
4. Elementos <provider> para provedores de conteúdo.

*Obs.: Os elementos anteriores fazem parte dos componentes (Atividade, Serviços, Broadcast Receivers e Provedores de conteúdo). Entretanto, apenas o Broadcast Receivers pode ser declarado no manifesto dinamicamente no código e registrado no sistema, chamando-se registerReceiver.*