FIAP

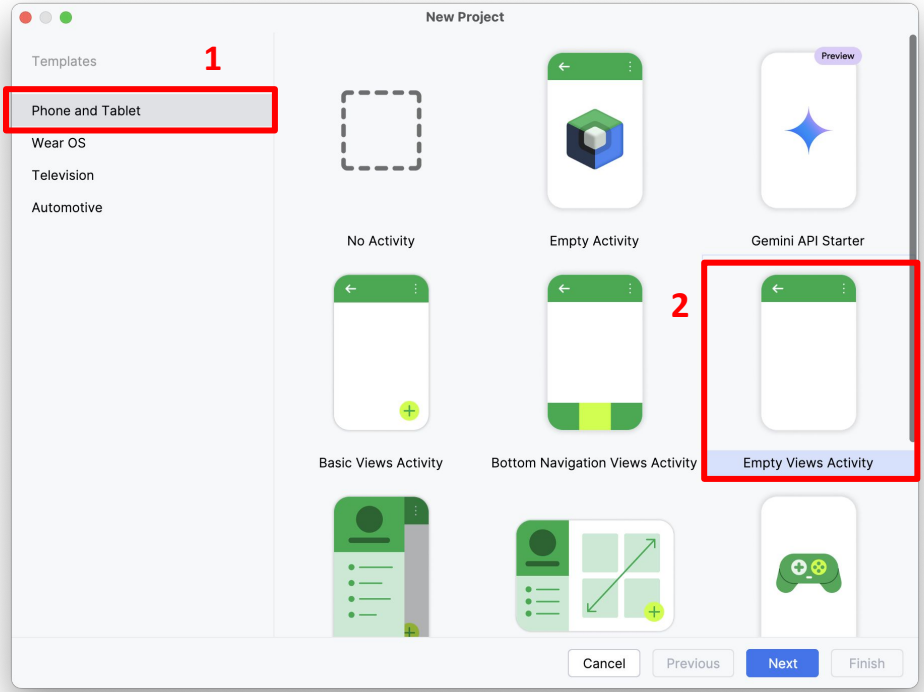
SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO ANDROID

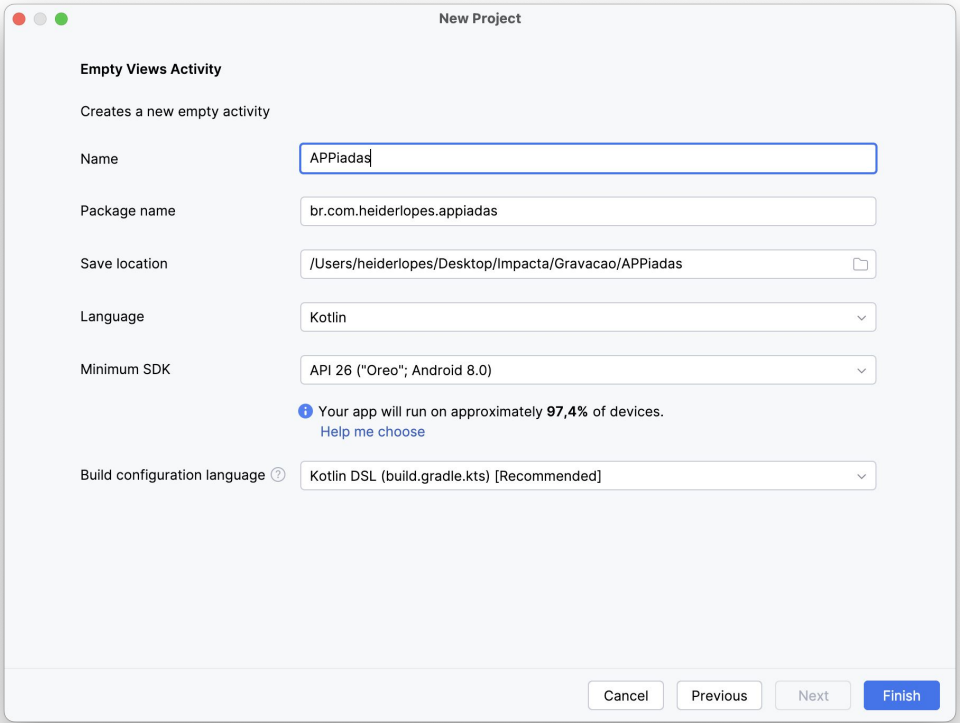
MOBILE DEVELOPMENT

TELAS DO NOSSO PRIMEIRO APLICATIVO

CRIANDO O PROJETO



CRIANDO O PROJETO



DIMENSÕES

Essas serão as primeiras dimensões configuradas no nosso aplicativo. Crie um arquivo chamado **dimens.xml** dentro de **res → values**

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

    <dimen name="joker_font_size">64sp</dimen>
    <dimen name="button_font_size">64sp</dimen>
    <dimen name="rounded_button_radius">32dp</dimen>
    <dimen name="rounded_button_padding">10dp</dimen>
    <dimen name="rounded_button_stroke_width">1dp</dimen>
    <dimen name="label_font_size">22sp</dimen>
    <dimen name="padding_default">16dp</dimen>
    <dimen name="margin_default">16dp</dimen>

</resources>
```

CORES

Abra o arquivo **colors.xml** e adicione as seguintes cores:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="colorAccent">#D81B60</color>

    <!-- Primary e OnPrimary para tema claro -->
    <color name="colorPrimary">#CE93D8</color>
    <!-- Alternativa para Primary mais escuro -->
    <color name="colorPrimaryDark">#9C27B0</color>
</resources>
```

TEXTOS

Para uma melhor padronização do aplicativo temos um arquivo responsável em centralizar as strings da nossa aplicação. Esse arquivo encontra-se dentro da pasta values (res → values) e chama-se **strings.xml**

Abra o arquivo e adicione as seguintes strings:

```
<resources>
    <string name="app_name">APPiadas</string>
    <string name="label_joker_hint">
        Aqui será exibida a piada
    </string>
    <string name="button_new_joker">
        Conte outra piada
    </string>
</resources>
```

TEMAS

Um **Theme (tema)** é um conjunto de atributos que definem a **aparência global** do seu **app** ou de uma **Activity**. Ele controla **cores**, **fontes**, **estilos de botões, barras de navegação**, **status bar,** etc.

Exemplos de coisas que um Theme controla:

- **colorPrimary** → cor principal do app (botões, AppBar, etc.)
- **colorBackground** → cor de fundo padrão
- **colorOnPrimary** → cor do texto/ícone quando o fundo é colorPrimary
- **android:fontFamily** → fonte padrão de todo o app
- **android:statusBarColor** → cor da status bar

TEMAS

Altere o arquivo **themes.xml:**

```
<resources xmlns:tools="http://schemas.android.com/tools" >
   <style name="Base.Theme.APPiadas"
parent="Theme.Material3.DayNight.NoActionBar" >
       <item name="colorPrimary" >@color/colorPrimary</ item>
       <item name="colorOnPrimary" >@color/white</ item>
       <item name="colorSecondary" >@color/colorAccent</ item>
       <item name="colorOnSecondary" >@color/white</ item>
       <item name="android:colorBackground" >@color/white</ item>
       <item name="colorOnBackground" >@color/black</ item>
   </style>
   <style name="Theme.APPiadas" parent="Base.Theme.APPiadas" />
</resources>
```

TEMAS

Altere o arquivo **themes.xml (night):**

```
<resources xmlns:tools="http://schemas.android.com/tools" >
    <!-- Base application theme. -->
    <style name="Base.Theme.APPiadas"
parent="Theme.Material3.DayNight.NoActionBar" >
        <!-- Cores principais -->
        <item name="colorPrimary" >@color/colorPrimaryDark</ item>
        <item name="colorOnPrimary" >@color/white</ item>
        <item name="colorSecondary" >@color/colorAccent</ item>
        <item name="colorOnSecondary" >@color/white</ item>

        <!-- Fundo -->
        <item name="android:colorBackground" >@color/black</ item>
        <item name="colorOnBackground" >@color/white</ item>
    </ style>
</ resources >
```

ESTILOS

Podemos definir estilos para serem aplicados às nossas views.

É um processo semelhante ao utilizado em HTML com CSS.

Um conjunto de estilos podem ser aplicados a uma Activity ou aplicação inteira recebendo aí o nome de **Tema**.

ESTILOS

Para criar um estilo é necessário:

1. Criar um arquivo **styles.xml** na pasta **res** →**values**

2. O conteúdo do arquivo de estilo deve ser:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <style name="meuestilo">
            <item name="android:layout_width">match_parent</item>
        </style>
</resources>
```

style name → identifica unicamente o seu estilo

item name → nome da propriedade (do exemplo: layout_width) – o seu respectivo valor fica entre <item></item> (do exemplo: match_parent)

ESTILOS

Para criar um estilo é necessário:

Abra o arquivo **styles.xml** na pasta **res/values** e adicione o seguinte código:

```
<style name="Container">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">match_parent</item>
    <item name="android:gravity">center</item>
    <item name="android:orientation">vertical</item>
    <item name="android:padding">@dimen/padding_default</item>
</style>
```

ESTILOS

<!-- Continuação do styles.xml -->

```
<style name="Label">
    <item name="android:layout width">match parent</item>
    <item name="android:layout height">wrap content</item>
    <item name="android:gravity">center</item>
    <item name="android:maxLines">5</item>
    <item name="android:minLines">5</item>
    <item name="android:textSize">@dimen/label font size</item>
    <item name="android:textStyle">italic|bold</item>
    <item name="android:textColor">@color/colorPrimary</item>
</style>
```

ESTILOS

<!-- Continuação do styles.xml -->

```
<style name="Button" parent="Widget.Material3.Button">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item
name="android:layout_marginTop">@dimen/margin_default</item>
    <item
name="android:layout_marginBottom">@dimen/margin_default</item>
</style>
```

APLICANDO OS ESTILOS

Arquivo **activity_main.xml**

Abra o arquivo **activity_main.xml**

Entre no modo **Text**

Alterar de **ConstraintLayout** para **LinearLayout**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="App ada"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

```
<?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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="App ada"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

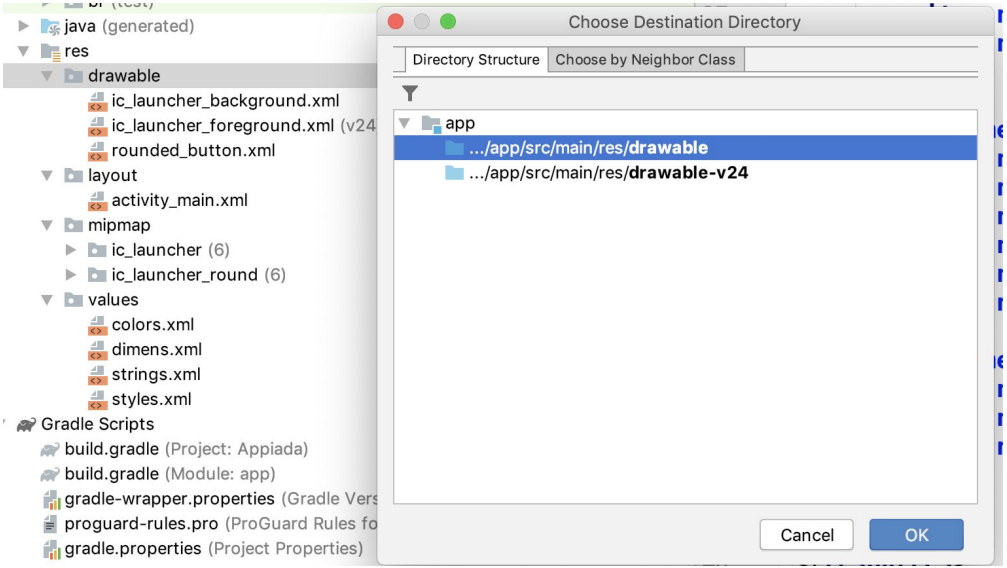
LAYOUT DA TELA

```
<?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"
    style="@style/Container"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvJoker"
        style="@style/Label"
        android:text="@string/label_joker_hint" />


    <Button
        android:id="@+id/btTellJoker"
        style="@style/Button"
        android:text="@string/button_new_joker" />

</LinearLayout>
```

BACKGROUND

Baixar os arquivos do repositório
https://github.com/heiderlopes/appiada_res

Adicionar à imagem na pasta **drawable**



BOTÃO CUSTOMIZADO

Aplique o background ao estilo container:

```
<style name="Container">
   <item name="android:background">@drawable/background_joker</item>
   <item name="android:layout_width">match_parent</item>
   <item name="android:layout_height">match_parent</item>
   <item name="android:gravity">center</item>
   <item name="android:orientation">vertical</item>
   <item name="android:padding">@dimen/padding_default</item>
</style>
```

RODE O APLICATIVO

TRABALHANDO COM ARRAYS

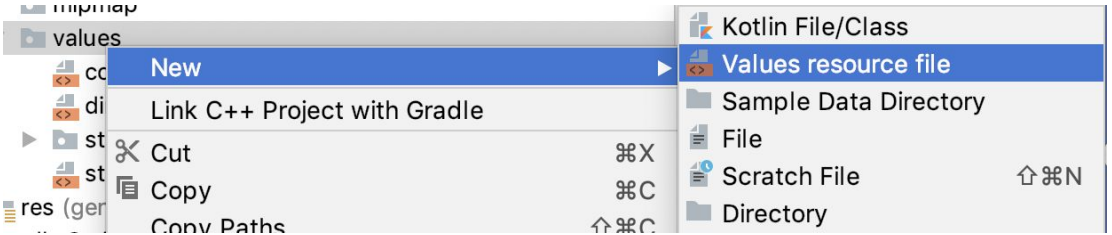
CRIANDO A LISTA DE PIADAS

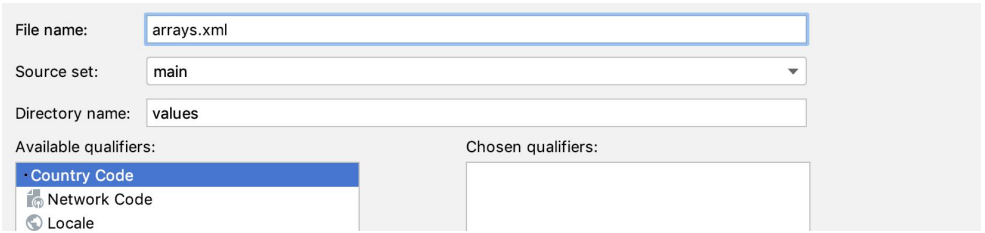
Abra o arquivo **strings.xml** e adicione as piadas em negrito:

```
<string name="joker_one">Por que a aranha é o animal mais carente do mundo?\nPorque ela é um aracneedyou.</ string>
<string name="joker_two">Por que o pinheiro não se perde na floresta?\nPorque ele tem uma pinha.</ string>
<string name="joker_three">O que um pagodeiro foi fazer na igreja?\nFoi cantar PÁ GOD.</ string>
<string name="joker_four">Por que o Napoleão era chamado sempre pras festas?\nPorque ele era BOM NA PARTY.</ string>
<string name="joker_five">Qual é o rei dos queijos?\nÉ o reiqueijão.</ string>
<string name="joker_six">Por que a velhinha não usa relógio?\nPorque ela é senhora.</ string>
<string name="joker_seven">Quando os americanos comeram carne pela primeira vez?\nQuando chegou Cristovão Com Lombo.</ string>
```

CRIANDO A LISTA DE PIADAS

Dentro da pasta **values** crie um arquivo chamado **arrays.xml**





CRIANDO A LISTA DE PIADAS

Dentro desse arquivo adicione o **string-array** contendo as piadas.

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

    <string-array name="jokers">
        <item>@string/joker one</item>
        <item>@string/joker two</item>
        <item>@string/joker three</item>
        <item>@string/joker_four</item>
        <item>@string/joker five</item>
        <item>@string/joker six</item>
        <item>@string/joker_seven</item>
    </string-array>

</resources>
```

FAZENDO O BIND DAS VIEWS

BIND DE VIEWS

Existem algumas formas de fazermos o bind de views. No decorrer do curso veremos algumas dessas formas.

Vamos começar baseado no nosso layout do arquivo **activity_main.xml**

```
<?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"
    style="@style/Container"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvJoker"
        style="@style/Label"
        android:text="@string/label_joker_hint" />

    <Button
        android:id="@+id/btTellJoker"
        style="@style/Button"
        android:text="@string/button_new_joker" />

</LinearLayout>
```

IMPLEMENTANDO VIA VIEWBINDING

Abra o arquivo **build.gradle (app)** e adicione a seguinte configuração:

```
buildFeatures {
    viewBinding = true
}
```

BIND DE VIEWS

Abra a **MainActivity** e configure o **binding**:

```
class MainActivity : AppCompatActivity() {

    lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }
}
```

BIND DE VIEWS

Adicione o **listener** ao **clique do botão** e execute a **função** para **contar uma piada.**

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    binding.btTellJoker.setOnClickListener {
        showJoker()
    }
}
```

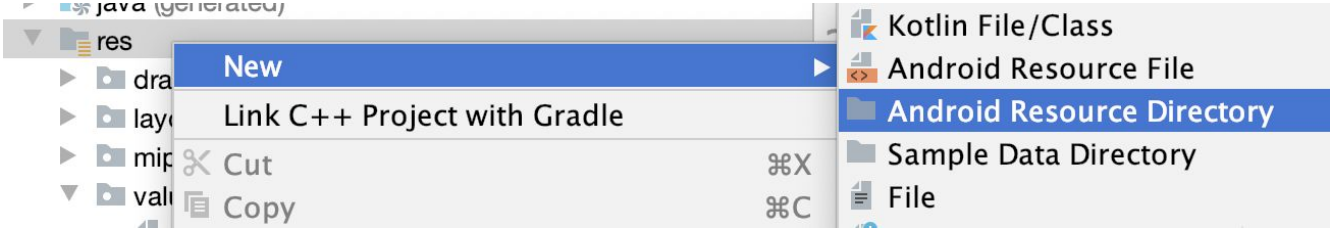
BIND DE VIEWS

Adicione o **listener** ao **clique do botão** e execute a **função** para **contar uma piada.**

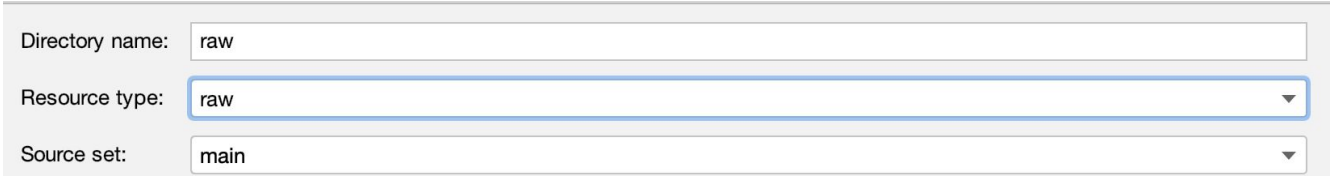
```
private fun showJoker() {
    val jokers = resources.getStringArray(R.array.jokers)
    val numberJoker = Random.nextInt(jokers.size)
    val joker = jokers[numberJoker]
    binding.tvJoker.text = joker
}
```

TOCANDO SOM NO APP

Crie uma pasta chamada **raw** dentro de **res.** Para isso, clique com o botão direito sobre **res** → **New** → **Android Resource Directory**



Altere o **Resource type** para **raw** e defina o **Directory name** para **raw.**



TOCANDO SOM NO APP

Copie o arquivo **badumtss.mp3** para dentro da pasta **raw** disponível em:
https://github.com/heiderlopes/appiada_res

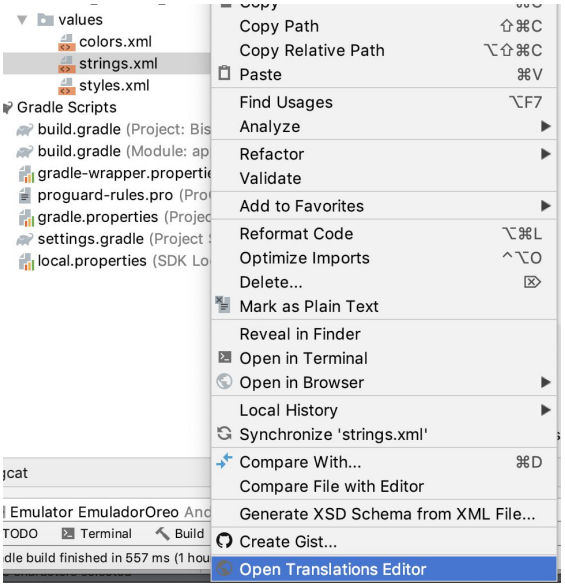
Tocando som na aplicação

Adicione o método e sua chamada conforme o código abaixo:

```
private fun showJoker() {
    val jokers = resources.getStringArray(R.array.jokers)
    val numberJoker = Random.nextInt(jokers.size)
    val joker = jokers[numberJoker]
    binding.tvJoker.text = joker
    playSong()
}


private fun playSong() {
    val mediaPlayer = MediaPlayer.create(this, R.raw.badumtss)
    mediaPlayer.start()
}
```

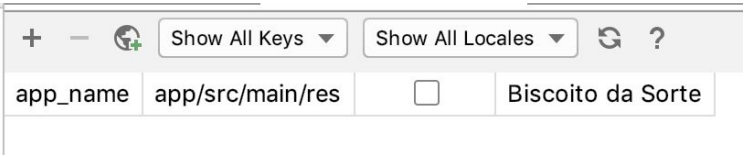
INTERNACIONALIZAÇÃO

INTERNACIONALIZAÇÃO

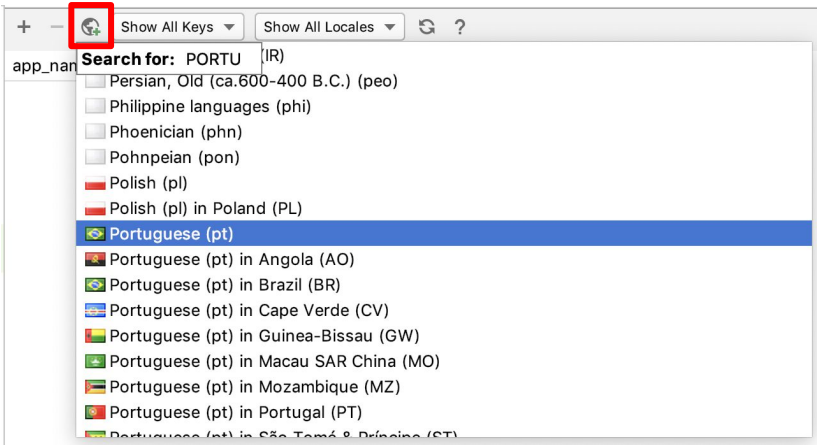


Clicar com o botão direito sobre o arquivo **strings.xml** (dentro de values)

Selecionar Open Translation Editor



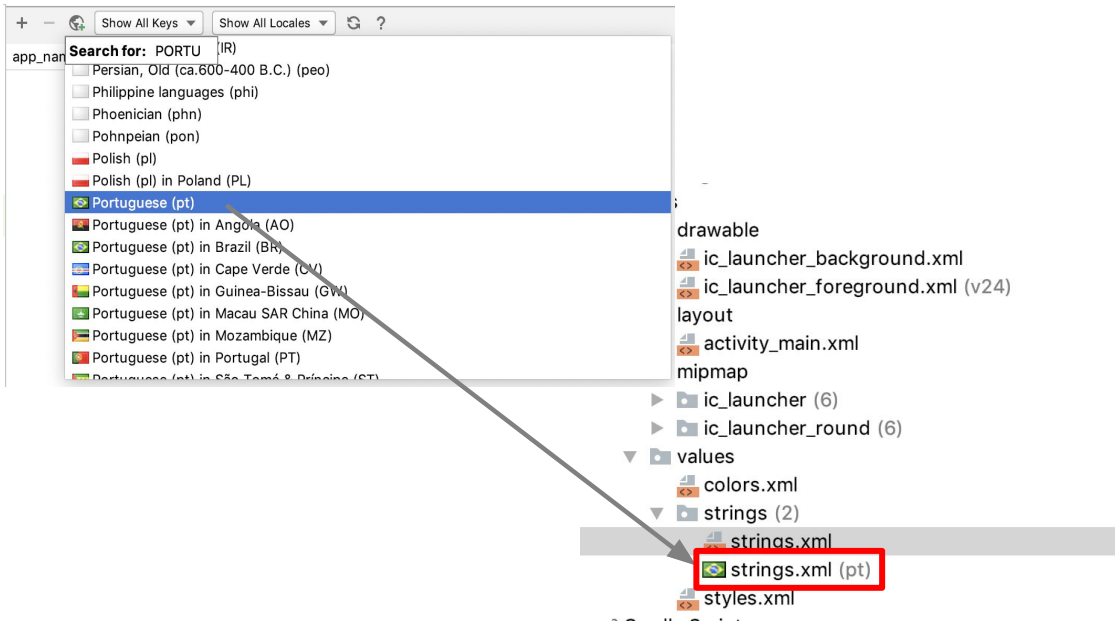
INTERNACIONALIZAÇÃO



Selecione o idioma que deseja adicionar e clique sobre ele

Após isso, você terá o arquivo com o idioma criado. Basta adicionar o texto correspondente ao idioma na sua respectiva coluna

INTERNACIONALIZAÇÃO



Para cada idioma selecionado será criado um arquivo **strings.xml** para o respectivo idioma. Então, basta preencher os campos Default value com os valores para as chaves (keys) que identificam os textos no idioma selecionado

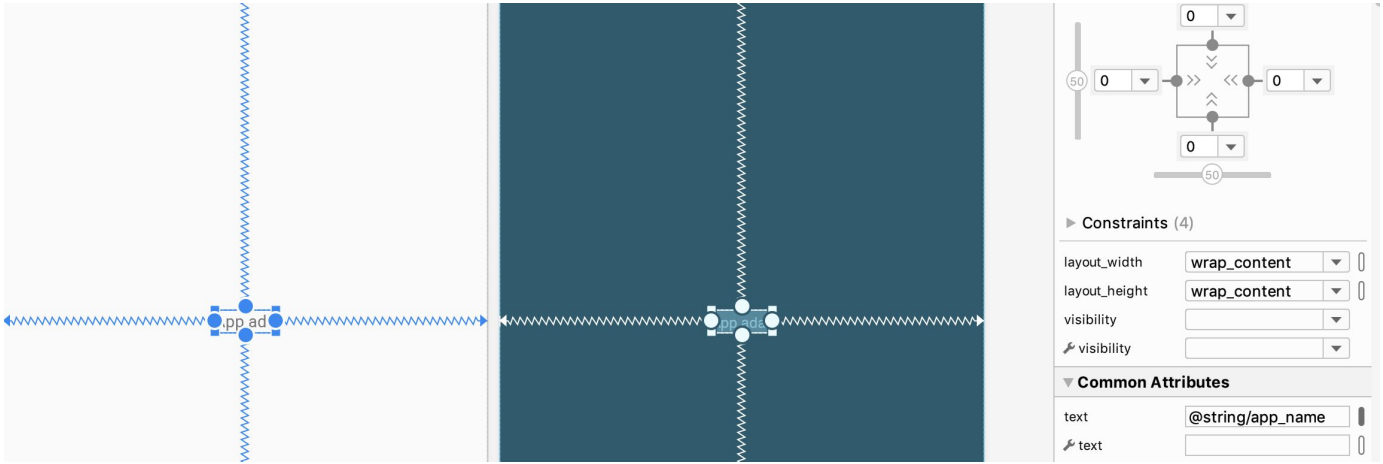
INTERNACIONALIZAÇÃO

No translator editor pode-se criar textos que serão utilizados na aplicação (em views, mensagens, etc...)

Cada texto possui uma chave (key) que o identifica unicamente e o seu respectivo valor, no idioma desejado

Para adicionar um novo texto, clique sobre o “+” na parte superior da tela

INTERNACIONALIZAÇÃO



Basta referenciar a chave (key) do texto dentro da propriedade text das views para que o texto, no respectivo idioma, seja exibido

Caso não exista tradução para a key no idioma, então o valor default (Default Value) será utilizado automaticamente

EXERCÍCIO 1

Exercício 1

Melhorando seu código.

Crie uma lógica para caso a piada selecionada não seja igual a última piada contada ele selecione outra.

Exercício 1 - Resposta 1

```
private var lastJokerIndex = -1

private fun showJoker() {

    val jokers =
resources.getStringArray(R.array.jokers)
    var numberJoker =
Random().nextInt(jokers.size)
    while (numberJoker ==
lastJokerIndex) {
        numberJoker =
Random().nextInt(jokers.size)
    }

    val joker = jokers[numberJoker]

    binding.tvJoker.text = joker
    lastJokerIndex = numberJoker

    playSong()
}
```

Exercício 1 - Resposta 2

```
private var lastJokerIndex = -1

private fun showJoker() {
    val jokers =
resources.getStringArray(R.array.jokers)
    var numberJoker: Int

    do {
        numberJoker =
Random.nextInt(jokers.size)
    } while (numberJoker ==
lastJokerIndex)

    val joker = jokers[numberJoker]
    binding.tvJoker.text = joker
    lastJokerIndex = numberJoker

    playSong()
}
```

EXERCÍCIO 2

Exercício 2

Adicione novas piadas para que seu app tenha uma maior variedade (em breve iremos consumir as informações através de webservices).

Altera o layout e as cores dos botões e da label do aplicativo

Adicione suporte para idioma **espanhol**.

Crie **dois arrays**, um contendo as **perguntas** e outro contendo as **respostas**. Ao clicar no botão, aguardar 3 segundos e exibir à resposta.

Adicione um novo **TextView** para exibir à resposta

Para esconder uma view:

nomeDaView.visibility = View.GONE

Para mostrar uma view:

nomeDaView.visibility = View.VISIBLE

EXERCÍCIO 3

Exercício 3 - Dica

Um exemplo de como programar para executar uma ação após um determinado tempo

```
val timeDelay = 2000
Handler(Looper.getMainLooper()).postDelay
ed({

  //o que deseja executar

}, timeDelay)
```

EXERCÍCIO COMPLEMENTAR

Exercício Complementar

Crie um aplicativo chamado **Biscoito da Sorte**

Ao clicar no botão o aplicativo deverá exibir uma mensagem.

O aplicativo deverá ter suporte inglês (default), português e espanhol.

Criar um botão customizado

Adicionar as medidas dentro do arquivo de dimensões

Adicionar as cores dentro do arquivo de cores

OBRIGADO

 /heider.lopes

 /in/heider-lopes-a06b2869/

FIAP


FIAP