



++

45697096

Criação de uma Interface Gráfica 2

Hybrid Mobile App Development

45697096

45697096

...



Prof. Alexandre dos Santos Mignon <profalexandre.mignon@fiap.com.br>

Componentes do tipo **Checkbox** são renderizados em forma de campos quadrados onde permite-se **escolher entre duas opções: marcados ou não marcados**. Caso haja vários componentes do tipo **Checkbox** na tela, poderá ocorrer a múltipla seleção destes componentes, sem anular a escolha anterior (**diferente do RadioButton**).



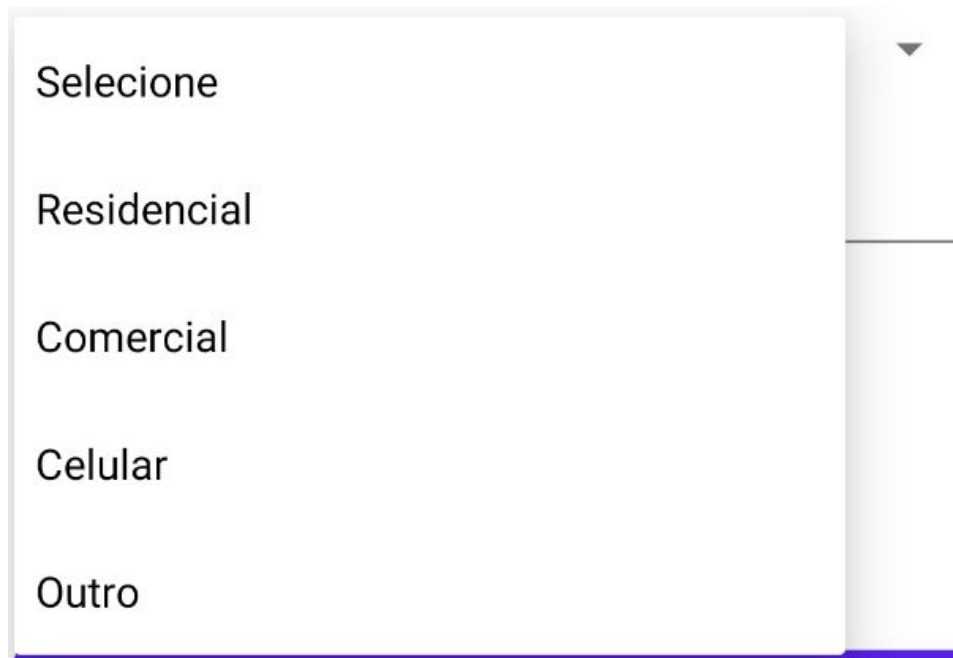
```
<CheckBox  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Marque-me!"  
    android:id="@+id/chkMarqueMe"  
    android:checked="true"/>
```

Obs: Para deixar o CheckBox marcado por padrão, basta definir o atributo **android:checked="true"** na declaração do componente.

Para verificar no Kotlin **se o CheckBox está selecionado**, basta usar propriedade **isChecked** de um **Checkbox** dentro de algum evento (clique de um botão ou envio de um formulário) conforme o exemplo abaixo:

```
//  
// Verificar se o checkbox está marcado.  
// Obs: Pega-se o elemento pelo id do mesmo  
//  
if ( chkMarqueMe.isChecked() ) { // Verdadeiro ou falso  
    Toast.makeText( context: this, text: "O checkbox Marque-Me foi selecionado...", Toast.LENGTH_SHORT).show()  
}
```

Componentes do tipo **Spinner** oferecem uma forma rápida de selecionar um valor de um conjunto.



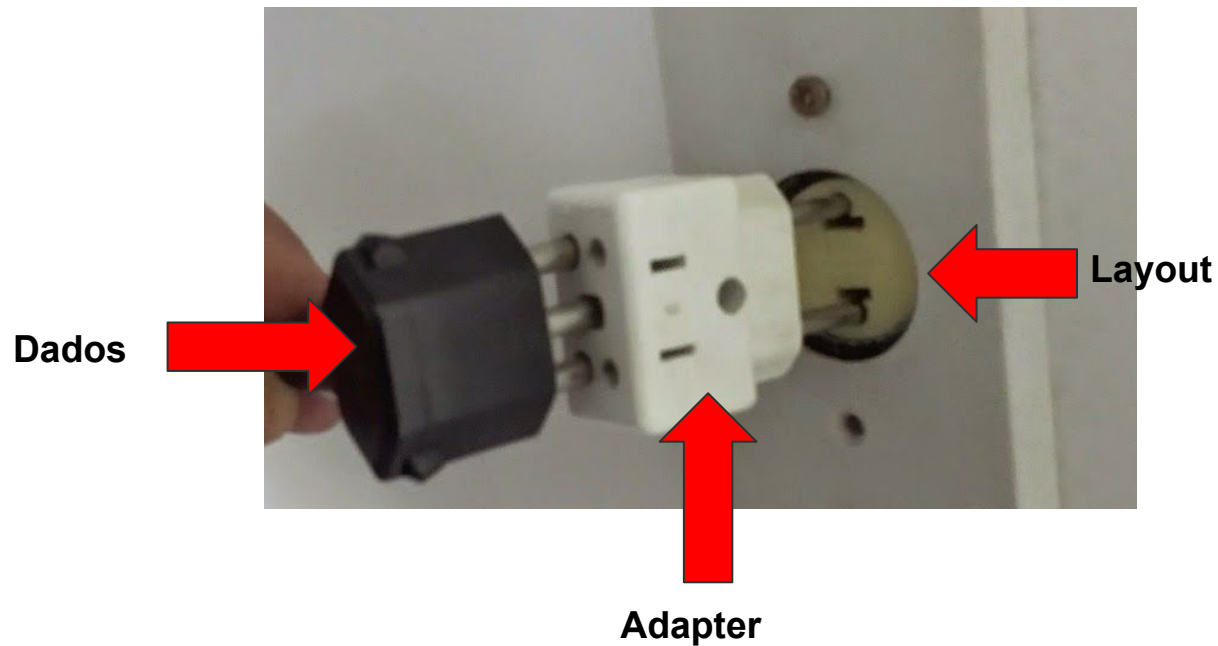
Para preenchermos o nosso **Spinner** iremos criar um **array de Strings** contendo o tipo de telefone informado pelo usuário.

Para adicionarmos os dados no Spinner usaremos um **ArrayAdapter**, que será responsável por ler o valor String de cada item do array e então adicioná-lo ao Spinner.


O **ArrayAdapter** recebe em seu **construtor 3 argumentos**:

1. Contexto (geralmente a própria Activity);
2. Layout que será utilizado, neste caso, será **android.R.layout.simple_spinner_item**, que é um layout já existente dentro da plataforma Android;
3. O array de dados para serem listados.


Analogia de um Adapter



```
val itensSpinner = arrayOf<String>("Selecione", "Residencial", "Comercial", "Celular", "Outro")
val adapter = ArrayAdapter(context: this, android.R.layout.simple_spinner_item, itensSpinner)
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinnerTipoTelefone.adapter = adapter
```



Configure Your Project



Empty Activity

Creates a new empty activity

Name

Exemplo Interface Grafica 2

Package name

br.com.fiap.exemplointerfacegrafica2

Save location


indremignon/AndroidStudioProjects/ExemploInterfaceGrafica2


Language

Kotlin

Minimum SDK

API 26: Android 8.0 (Oreo)

 Your app will run on approximately 60.8% of devices.
[Help me choose](#)

☐ Use legacy android.support libraries 

Cancel

Previous

Next

Finish

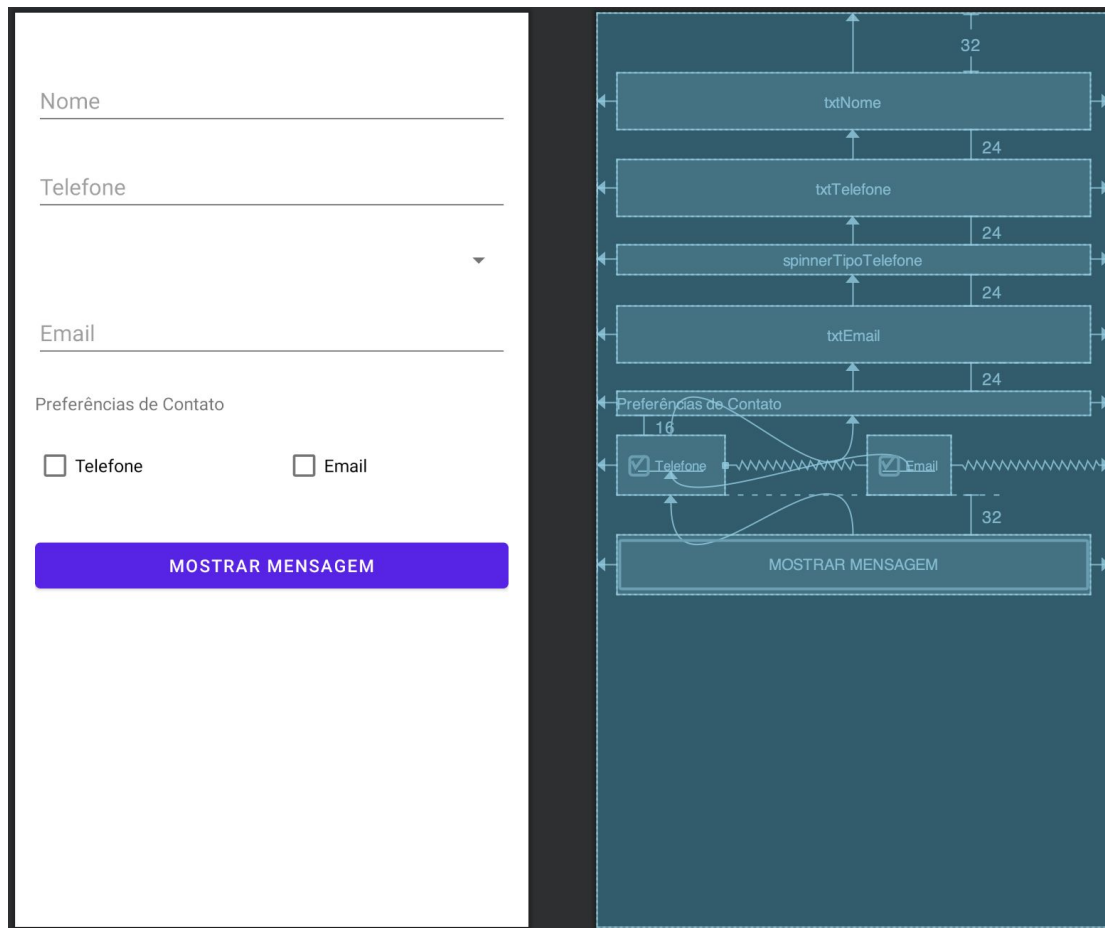
build.gradle

```
plugins {  
    id 'com.android.application'  
    id 'kotlin-android'  
    id 'kotlin-android-extensions'  
}
```

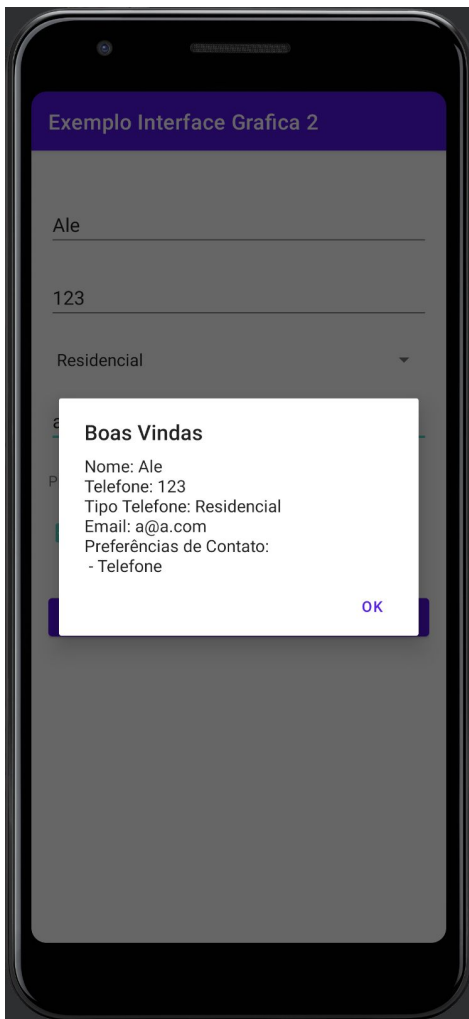
Activity

```
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.view.View  
import kotlinx.android.synthetic.main.activity_main.*
```

Exemplo - Interface Gráfica



Exemplo - Interface Gráfica



```
fun camposValidos() : Boolean {  
    if (txtNome.text.trim().isEmpty() || txtTelefone.text.trim().isEmpty()  
        || spinnerTipoTelefone.selectedItemPosition == 0  
        || txtEmail.text.trim().isEmpty()) {  
        Toast.makeText(context: this, text: "Preencha todos os campos.", Toast.LENGTH_LONG).show()  
        return false  
    }  
    return true  
}  
  
fun alert(titulo: String, mensagem: String) {  
    val builder = AlertDialog.Builder(context: this)  
    builder  
        .setTitle(titulo)  
        .setMessage(mensagem)  
        .setPositiveButton(text: "Ok", listener: null)  
    builder.create().show()  
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    val itensSpinner = arrayOf("Selecione", "Residencial", "Comercial", "Celular", "Outro")  
    val adapter = ArrayAdapter(context: this, android.R.layout.simple_spinner_item, itensSpinner)  
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);  
    spinnerTipoTelefone.adapter = adapter  
  
    btnMsg.setOnClickListener { it: View!  
        if (camposValidos()) {  
            var msg = ""  
            msg += "Nome: ${txtNome.text}  
            |Telefone: ${txtTelefone.text}  
            |Tipo Telefone: ${spinnerTipoTelefone.selectedItem}  
            |Email: ${txtEmail.text}  
            |Preferências de Contato:  
            """.trimMargin( marginPrefix: "|")  
            if (cbTelefone.isChecked) {  
                msg += "\n - Telefone"  
            }  
            if (cbEmail.isChecked) {  
                msg += "\n - Email"  
            }  
            alert( titulo: "Boas Vindas", msg)  
        }  
    }  
}
```

RadioButton/RadioGroup

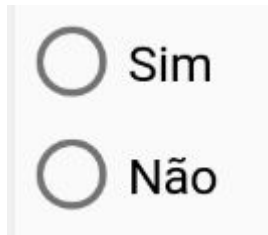
Componentes do tipo **RadioButton**, são componentes renderizados em formato de círculos e que permitem **escolher apenas um RadioButton por vez** que esteja dentro do mesmo **RadioGroup**. O RadioGroup por sua vez, é o componente que agrupa diversos RadioButtons.

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/rdgOpcoes">

    <RadioButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sim"
        android:id="@+id/rdbSim"/>

    <RadioButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Não"
        android:id="@+id/rdbNao"/>

</RadioGroup>
```



Obs: Para deixar o RadioButton marcado por padrão, basta definir o atributo **android:checked="true"** na declaração do componente.

RadioButton/RadioGroup

Para verificar no Kotlin qual dos **RadioButtons** do exemplo anterior está selecionado, basta usar a propriedade **isChecked** do objeto, conforme o exemplo abaixo dentro de algum evento (clique do botão por exemplo):

```
//  
// Verificar se um radio button com a opção SIM está marcado ou se foi a opção não.  
// Obs: Pega-se o elemento pelo o id do mesmo:  
//  
if ( rdbSim.isChecked() ) { // Verdadeiro ou falso  
    Toast.makeText( context: this, text: "A opção sim foi selecionada...", Toast.LENGTH_SHORT).show()  
} else {  
    Toast.makeText( context: this, text: "A opção não foi selecionada...", Toast.LENGTH_SHORT).show()  
}
```

RadioButton/RadioGroup

Uma outra forma mais simples quando se tem muitas opções disponíveis é: comparar através do atributo **checkedRadioButtonId** do **RadioGroup**, qual o **ID do RadioButton selecionado**, conforme a imagem abaixo.

```
//  
// Verifica de uma forma mais fácil pegando o id do RadioButton selecionando  
// dentro de um RadioGroup:  
//  
val selecionado = when ( rdgOpcoes.checkedRadioButtonId ) {  
    R.id.rdbSim -> "Sim"  
    R.id.rdbNao -> "Não"  
    else -> "Nenhuma opção"  
}  
Toast.makeText( context: this, text: "A opção selecionada foi: ${selecionado}", Toast.LENGTH_SHORT ).show()
```


++

45697096

Dúvidas?

◇◇◇

45697096

45697096

■ ■ ■



Copyright © 2020 Prof. Alexandre dos Santos Mignon <profalexandre.mignon@fiap.com.br>

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).