



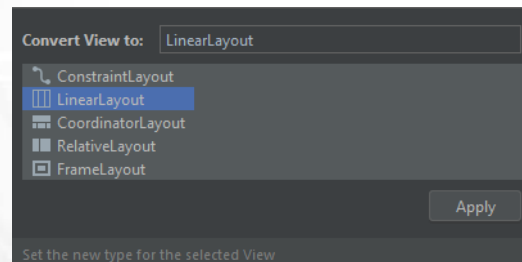
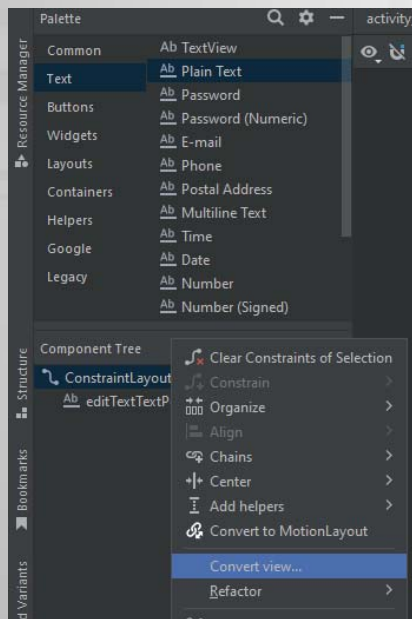
Componentes de Interface

Professor: Hércules Santhus

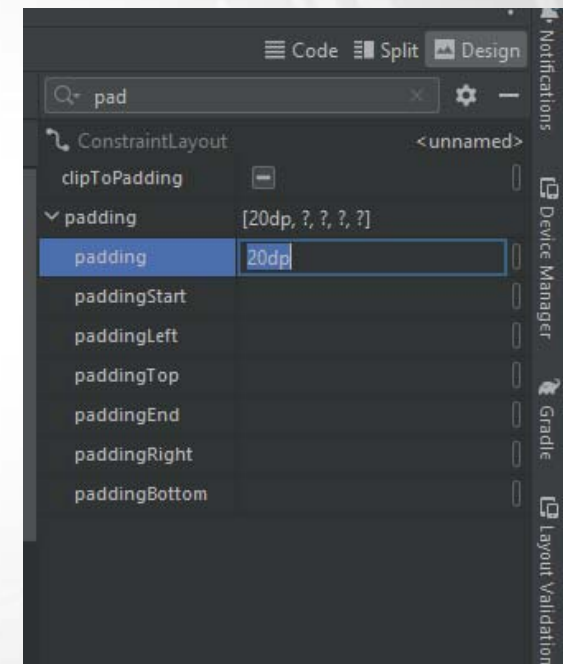


Configurando Layout

Antes de add qualquer componente



2. Escolha a opção **LinearLayout**



3. Configure um espaçamento nas laterais (**padding**) de **20dp**

1. Click em **ContrintLayout** com botão direito e escolha **Convert view**



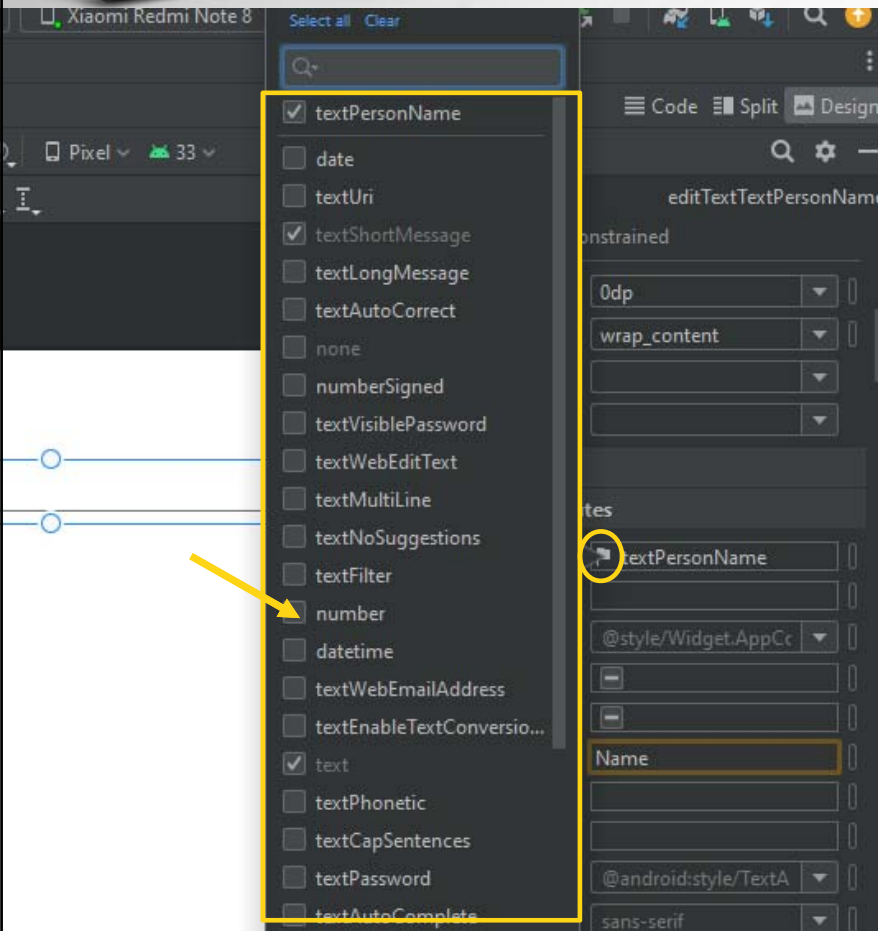
Caixas de texto

Se você clicar na bandeirinha em inputType, verá várias opções de formatação para sua caixa de texto

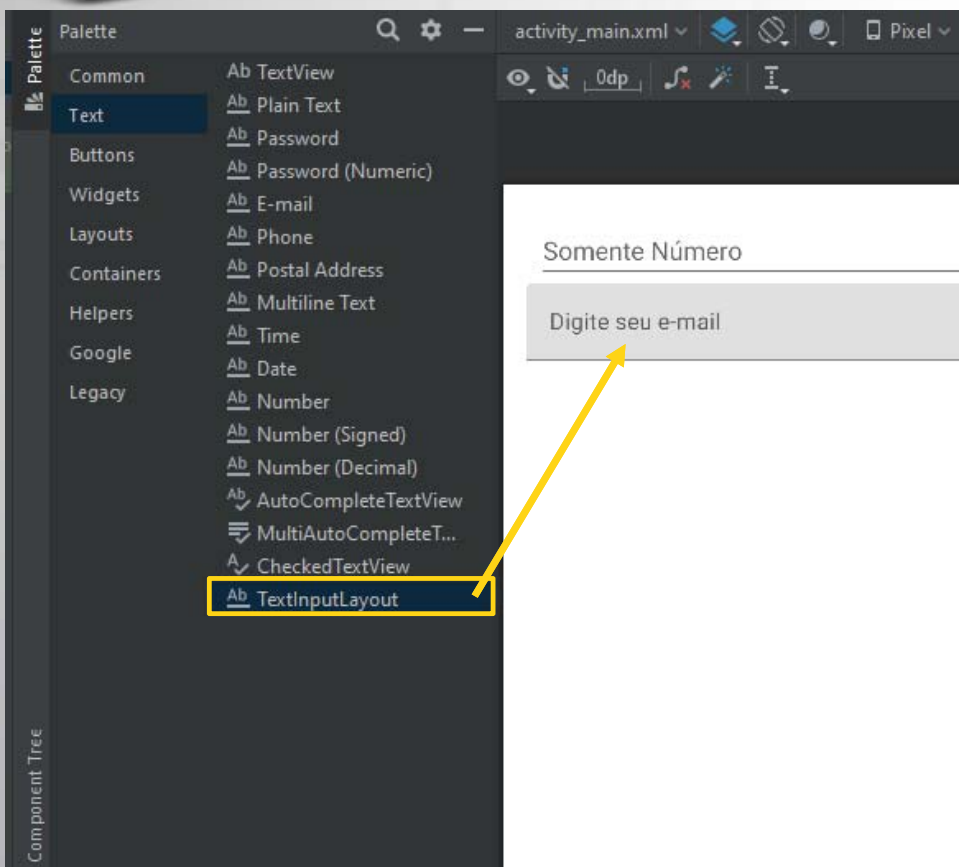
The screenshot displays the Android Studio interface. On the left, the 'Palette' shows the 'Text' widget selected. In the center, a text input field labeled 'Name' is shown on a design surface. On the right, the 'Attributes' panel for the 'EditText' widget is visible. The 'inputType' attribute is set to 'textPersonName', and a yellow arrow points to this attribute. The 'text' attribute is also visible, set to 'Name'.



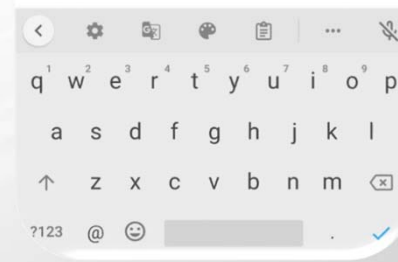
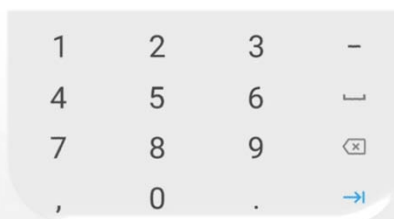
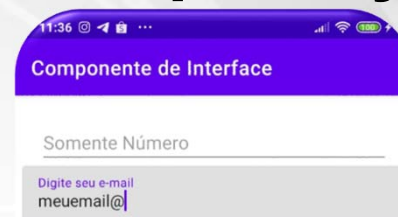
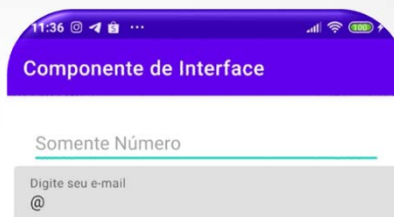
Tipos de caixas



Se você escolher por exemplo **number**, a sua caixa de texto só permitirá números.
Ou seja, o teclado se adapta ao formato da caixa de texto.



TextInputLayout



Veja que o teclado se adapta à caixa de texto. No caso do **TextInputLayout**, o **init** permanece logo acima da caixa



Capturando valores dos componentes

Tipo do componente

Nome do objeto

id do componente

```
EditText campoEmail = findViewById(R.id.editTextEmail);  
String email = campoEmail.getText().toString();
```

Variável String que recebe o valor do componente



Setando valores ao componente

Tipo do componente

Nome do objeto

id do componente

```
TextView resultado = findViewById(R.id.textViewResultado);  
resultado.setText(email);
```

Variável String que recebeu o valor do componente

Método que inseri o valor ao componente



Crie uma interface que receba o número de telefone e email.

Ao clicar no botão enviar os valores da caixa de texto devem ser carregados no textView a baixo dos botões, conforme a imagem.

Obs: As caixas de texto devem está configurada para receber os tipos de dados específicos

Atividade

16:05

Componente de Interface

Telefone

Digite seu E-mail

ENVIAR

LIMPAR

testecomponentes@hotmail.ufr | 9591234567 |

16:05

Componente de Interface

Telefone

9591234567

Digite seu E-mail

testecomponentes@hotmail.ufr

ENVIAR

LIMPAR

testecomponentes@hotmail.ufr | 9591234567 |



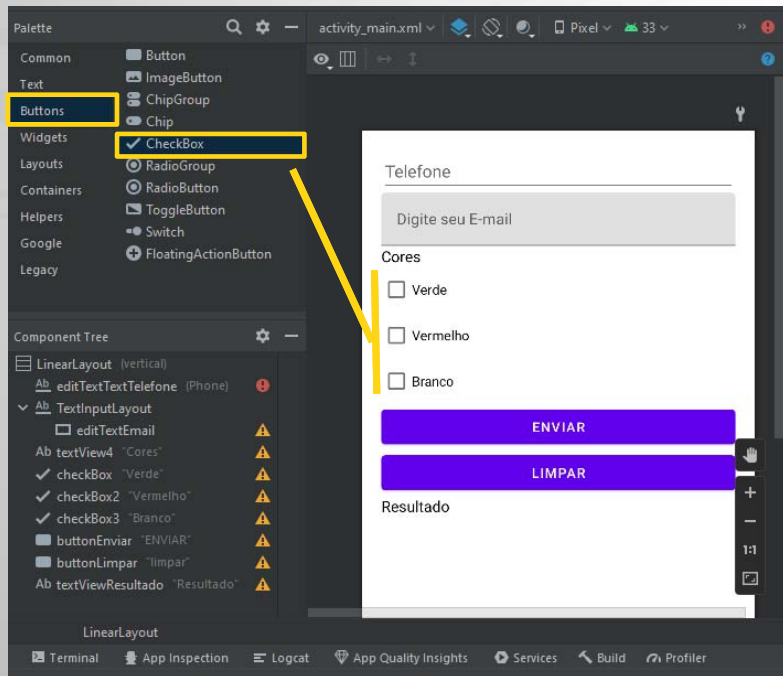
CheckBox

Para verificar se um **CheckBox** foi selecionado, você deve usar o método **isChecked()**. Ele irá retornar um valor Booleano. **True** ou **False**

```
CheckBox checkVerde = findViewById(R.id.checkBoxVerde);
```

```
if (checkVerde.isChecked()){  
  
    String corSelecionada = checkVerde.getText().toString();  
    resultado.setText(corSelecionada);  
  
}
```

O **CheckBox** permite que você marque mais de uma opção.





Pratica

```
private CheckBox checkVerde, getCheckVermelho, getCheckBranco;
```

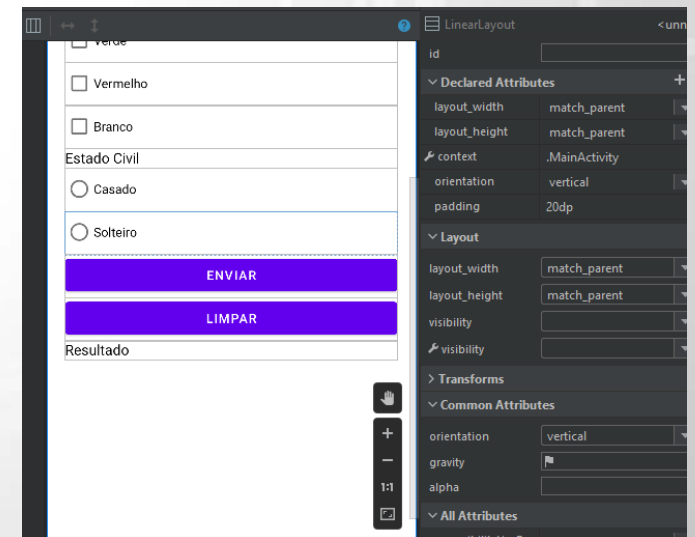
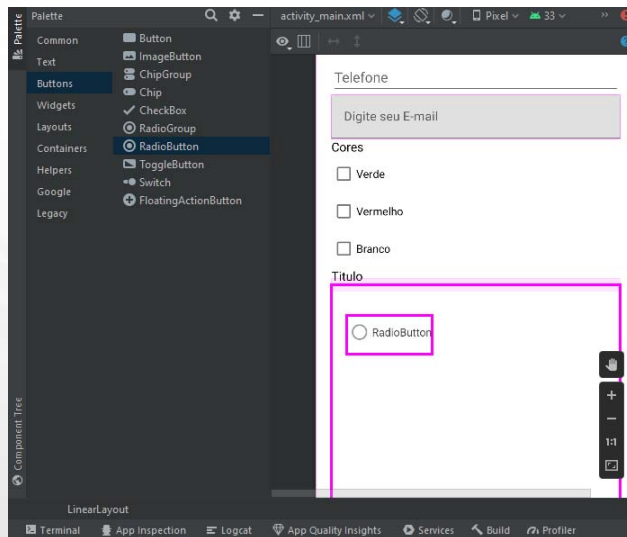
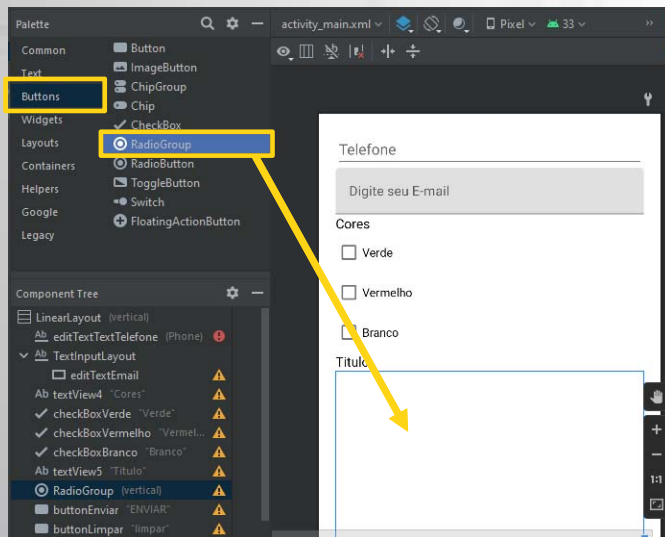
```
checkVerde = findViewById(R.id.checkBoxVerde);  
getCheckVermelho = findViewById(R.id.checkBoxVermelho);  
getCheckBranco = findViewById(R.id.checkBoxBranco);
```

```
public void verificar_check(){  
  
    String texto = "";  
  
    if (checkVerde.isChecked()){  
        texto = "Verde Selecionado";  
    }  
    if (getCheckVermelho.isChecked()){  
        texto = texto + "Vermelho Selecionado"; //concatena caso outro checkBox seja selecionado  
    }  
    if (getCheckBranco.isChecked()){  
        texto = texto + "Branco Selecionado";  
    }  
    resultado.setText(texto);  
}
```



RadioGroup e RadioButton

Diferente do **CheckBox**, o **RadioButton** permite apenas um item selecionado. Para adicionar mais de um **RadioButton**, é necessário adicionar primeiramente o **RadioGroup**, em seguida adicionar as opções necessárias





Prática

```
private RadioButton RadioCasado, RadioSolteiro;
```

```
RadioSolteiro = findViewById(R.id.radioButtonSolteiro);  
RadioCasado = findViewById(R.id.radioButtonCasado);
```

```
public void radioEstadoCivil () {  
  
    if(RadioCasado.isChecked()){  
        resultado.setText("Casado");  
    } else if (RadioSolteiro.isChecked()){  
        resultado.setText("Solteiro");  
    }  
  
}
```

Telefone

Digite seu E-mail

Cores

☐ Verde

☐ Vermelho

☐ Branco

Estado Civil

☐ Casado

☐ Solteiro

ENVIAR

LIMPAR

Resultado



```
private RadioGroup opcaoEstadoCivil;
```

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    radioEstadoCivil();  
}
```

```
public void radioEstadoCivil() {  
    opcaoEstadoCivil.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {  
        @Override  
        public void onCheckedChanged(RadioGroup radioGroup, int i) {  
            if (i == R.id.radioButtonCasado) {  
                resultado.setText("Casado");  
            } else if (i == R.id.radioButtonSolteiro) {  
                resultado.setText("Solteiro");  
            }  
        }  
    });  
}
```

Adicionando ChangeListener

Podemos utilizar o **RadioGroup** para saber qual opção o usuário digitou, sem precisar usar o **RadioButton**.

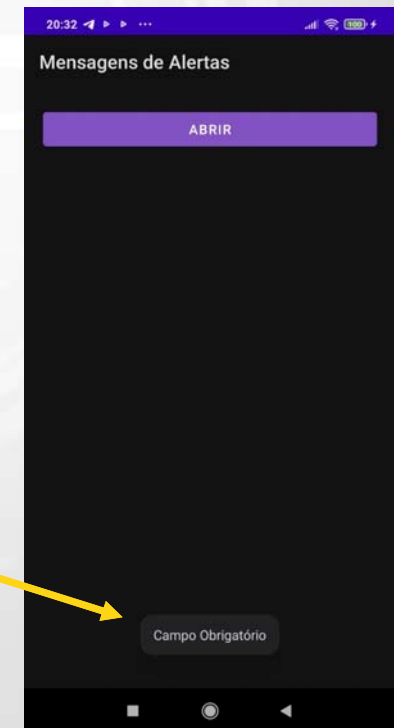
Dessa forma basta criarmos ouvinte para que o App detecte automaticamente ao usuário clicar na opção, sem precisar usar o botão ENVIAR. Para isso precisamos chamar o método dentro do **onCreate** para que seja carregado antes da interface ser construída para o usuário.



Toast

Exibe uma mensagem, quando o usuário faz uma ação e depois de um tempo a mensagem desaparece automaticamente

```
public void abrirToast (View v){  
    Toast.makeText(this, "Campo Obrigatório", Toast.LENGTH_SHORT).show();  
}
```

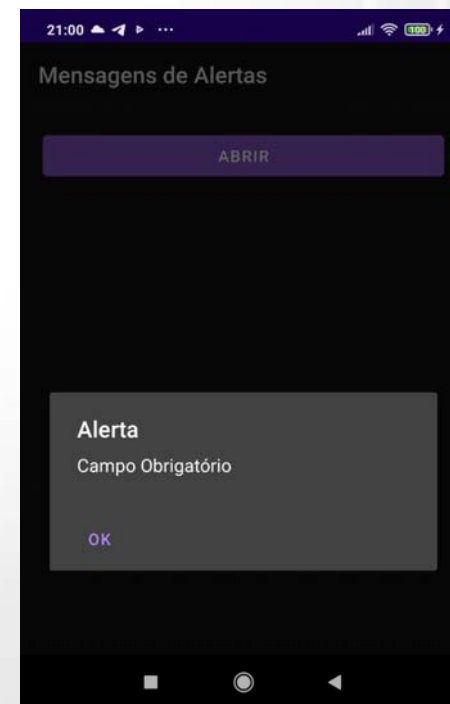




AlertDialog

Outra forma de apresentar uma mensagem ao usuário é por meio do **AlertDialog**

```
public void abrirAlerta (View v){  
  
    AlertDialog.Builder dialogo = new AlertDialog.Builder(this);  
    dialogo.setTitle("Alerta");  
    dialogo.setMessage("Campo Obrigatório");  
    dialogo.setNeutralButton("ok", null);  
    dialogo.show();  
  
}
```





Atividade

Criar um App de combustível que calcule a melhor opção de abastecimento para o usuário, se é melhor usar álcool ou gasolina.

Se ($\text{valor_do_álcool} / \text{valor_da_gasolina} < 0.7$)
é melhor usar gasolina

O campo valores é obrigatório

The screenshot shows a mobile app interface with a blue header bar containing the text "Alcool ou gasolina". Below the header is a white area with a blue car icon and a blue bar containing the text "Álcool • OU • Gasolina". Below this is a text prompt "Saiba a melhor opção de abastecimento do seu carro!". There are two input fields: "Preço álcool ex: 1.90" with the value "1.90" entered, and "Preço gasolina ex: 2.60" with the value "2.60" entered. Below the input fields is a grey button labeled "CALCULAR". Below the button is the text "Melhor utilizar Gasolina". The app is running on a white smartphone with a black navigation bar at the bottom.