

Aplicativo Media

Criar aplicativo calcular media com

ConstraintLayout

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]

Calculodemedia > app > src > main > res > layout > activity_main.xml

activity_main.xml MainActivity.java colors.xml

Project Resource Manager Structure Bookmarks Build variants

activity_main.xml Pixel 6 API 30

Code Split Design

Paleta

EditText

```
8
9
10
11    <EditText
12        android:id="@+id/nota1"
13        android:layout_width="wrap_content"
14        android:layout_height="wrap_content"
15        android:background="@color/light_gray"
16        android:gravity="center"
17        android:hint="Nota 1"
18        android:inputType="number"
19        android:maxLength="2"
20        android:padding="20dp"
21        android:text=""
22        app:layout_constraintTop_toTopOf="parent"
23        app:layout_constraintStart_toStartOf="parent"
24        android:layout_marginTop="50dp"
25        android:layout_marginStart="50dp"
26        app:layout_marginStart="50dp"
27    />
```

Component Tree

No constraint layout o elemento fica preso no lugar através de ancoras, que são essas alcas que ligam a esquerda e a direita. Percebiam que ao colocar uma constraint ou ancora, a alca fica azul. Precisamos ao menos uma horizontal e uma vertical para conseguir uma coordenada de fixação. Aqui eu criei alcas de fixação a esquerda e no topo ligadas ao parent, ou seja ao container que eh o layout

androidx.constraintlayout.widget.ConstraintLayout > EditText

Version Control TODO Problems Terminal Logcat App Inspection Build Profiler Event Log Layout Inspector

Gradle sync finished in 2 m 44 s 258 ms (today 09:59) 20:31 LF UTF-8 4 spaces

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]

Calculodemedia > app > src > main > res > layout > activity_main.xml

MainActivity.java colors.xml

activity_main.xml

Code Split Design

Resource Manager

Structure

Bookmarks

Build Variants

23 app:layout_constraintStart_toStartOf="parent" ① 5 ▲ 9 ✘ 11 ^ v

24 android:layout_marginTop="50dp"

25 android:layout_marginStart="50dp"

26 app:layout_marginStart="50dp"

27 />

28

29 <EditText

30 android:id="@+id/nota2"

31 android:layout_width="wrap_content"

32 android:layout_height="wrap_content"

33 android:background="@color/light_gray"

34 android:gravity="center"

35 android:hint="Nota 2"

36 android:inputType="number"

37 android:maxLength="2"

38 android:padding="20dp"

39 android:text=""

40 app:layout_constraintEnd_toEndOf="parent"

41 app:layout_constraintTop_toTopOf="parent"

42 android:layout_marginTop="50dp"

43 android:layout_marginEnd="50dp"/>

44

androidx.constraintlayout.widget.ConstraintLayout > EditText

Version Control TODO Problems Terminal Logcat App Inspection Build Profiler

Event Log Layout Inspector

Gradle Sync finished in 2 m 44 s 258 ms (today 09:59) 43:39 LF UTF-8 4 spaces

The screenshot shows the Android Studio interface with the XML code editor open for 'activity_main.xml'. The code defines a ConstraintLayout containing an EditText component. The XML includes attributes like 'app:layout_constraintStart_toStartOf="parent"', 'android:layout_marginTop="50dp"', and 'app:layout_constraintEnd_toEndOf="parent"'. The right side of the interface shows the 'Design' tab, where a visual representation of the layout is displayed. It shows a yellow rectangular container labeled 'Nota 1' containing a smaller gray rectangle labeled 'Nota 2'. A blue constraint grid is overlaid on the layout, with arrows indicating top and end margins of 50dp. A yellow callout box highlights the 'Nota 2' box. A vertical yellow bar on the left indicates the current scroll position.

Para a segunda nota, vou ligar acima no layout e a direita também no layout. O layout é o componente pai, onde estão todos os elementos dentro. Pai em inglês é parent, por isso que ligo em parent. Depois vamos colocar uma margem top e uma margem end de 50dp

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]

Calculodemedia > app > src > main > res > layout > activity_main.xml

activity_main.xml MainActivity.java colors.xml

Project Resource Manager Structure Bookmarks Build Variants

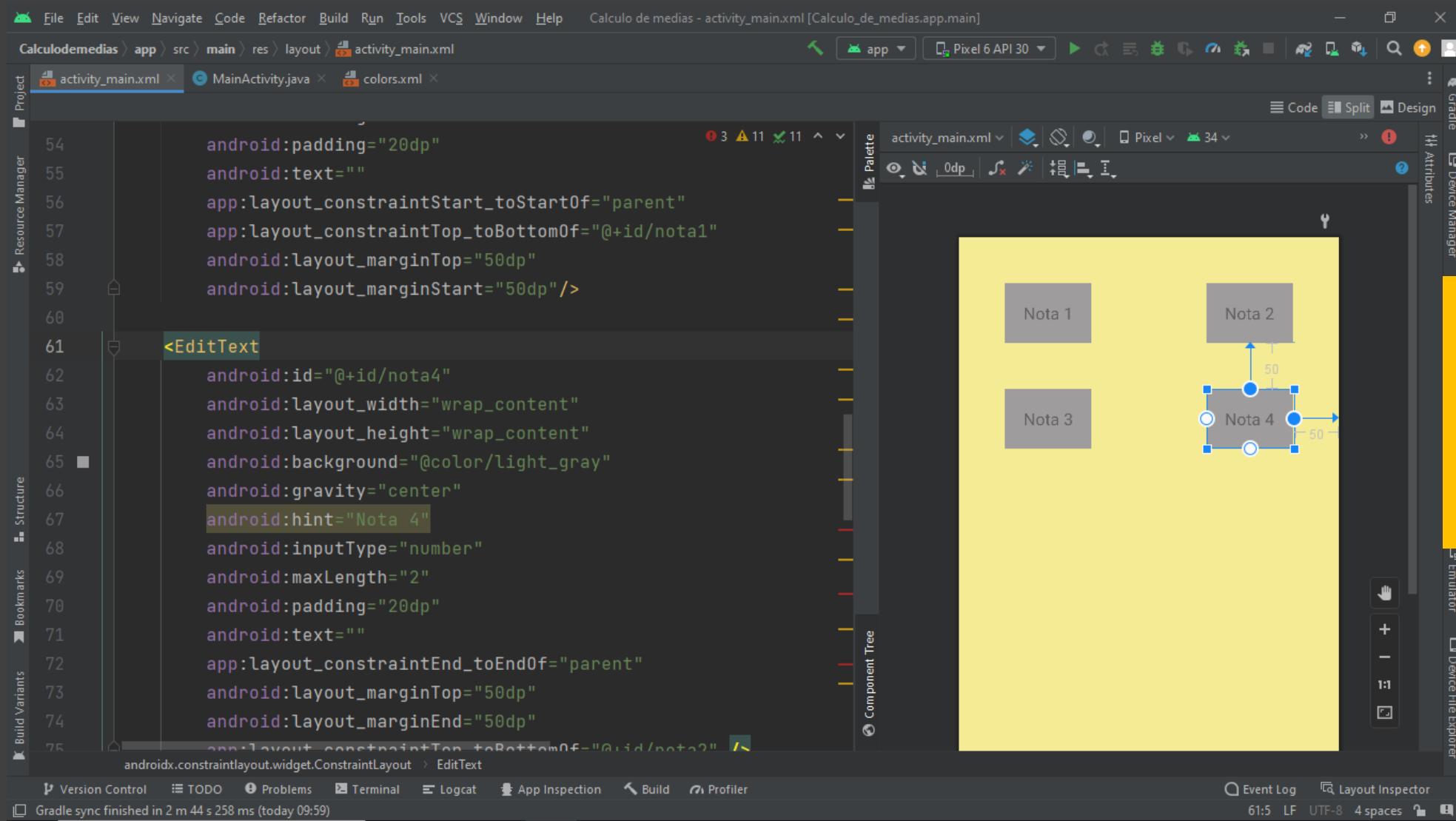
```
39     android:text=""  
40     app:layout_constraintEnd_toEndOf="parent"  
41     app:layout_constraintTop_toTopOf="parent"  
42     android:layout_marginTop="50dp"  
43     android:layout_marginEnd="50dp"/>/  
  
44  
45     <EditText  
46         android:id="@+id/nota3"  
47         android:layout_width="wrap_content"  
48         android:layout_height="wrap_content"  
49         android:background="@color/light_gray"  
50         android:gravity="center"  
51         android:hint="Nota 3"  
52         android:inputType="number"  
53         android:maxLength="2"  
54         android:padding="20dp"  
55         android:text=""  
56         app:layout_constraintStart_toStartOf="parent"  
57         app:layout_constraintTop_toBottomOf="@+id/nota1"  
58         android:layout_marginTop="50dp"  
59         android:layout_marginStart="50dp"/>/  
  
    androidx.constraintlayout.widget.ConstraintLayout > EditText
```

Version Control TODO Problems Terminal Logcat App Inspection Build Profiler

Event Log Layout Inspector 59:41 LF UTF-8 4 spaces

Gradle sync finished in 2 m 44 s 258 ms (today 09:59)

A terceira nota vai ficar ligada a esquerda no container e acima vai ficar ligada a nota1. Igamos a parte de cima da nota3 a parte de baixo da nota1 e a parte esquerda da nota3 a parte esquerda do layout. O primeiro elemento, que eh a nota1 vai ligado ao layout ou pai. Os outros elementos abaixo dele vao ligados uns aos outros, assim se baixar o primeiro, todos os outros vao baixar junto



Para a quarta nota,
ligamos a parte de
cima dela com a parte
de baixo da nota2 e a
parte direita dele a
parte direita do layout

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]

Calculodemedia > app > src > main > res > layout > activity_main.xml

activity_main.xml MainActivity.java colors.xml

Project Resource Manager Structure Bookmarks Build Variants

```
70     android:padding="20dp"
71     android:text=""
72     app:layout_constraintEnd_toEndOf="parent"
73     android:layout_marginTop="50dp"
74     android:layout_marginEnd="50dp"
75     app:layout_constraintTop_toBottomOf="@+id/nota2" />
76
76
77 <EditText
78     android:id="@+id/numeroFalta"
79     android:layout_width="wrap_content"
80     android:layout_height="wrap_content"
81     android:background="@color/light_gray"
82     android:gravity="center"
83     android:hint="Numero de faltas"
84     android:inputType="number"
85     android:padding="20dp"
86     android:text=""
87     android:layout_marginTop="50dp"
88     app:layout_constraintEnd_toEndOf="parent"
89     app:layout_constraintStart_toStartOf="parent"
90     app:layout_constraintTop_toBottomOf="@+id/nota3" />
```

Component Tree

activity_main.xml Pixel 6 API 30 Pixel 34

Nota 1 Nota 2
Nota 3 Nota 4

Numero de faltas

Para o campo das faltas, ligamos a parte de cima do elemento a parte de baixo da nota3, e como quero que o elemento fique centralizado na tela, ligo a parte esquerda ao layout e a parte direita ao layout, assim crio uma mola puxando para um lado e outra puxando para o outro lado, e isso deixa o elemento centralizado

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]
- Project Tab:** Calculodemedia > app > src > main > res > layout > activity_main.xml
- Code Editor:** Displays the XML code for activity_main.xml, defining a ConstraintLayout with various views like TextViews for notes and a Button for calculating.
- Preview:** Shows the visual representation of the layout on a Pixel 6 API 30 device, featuring a yellow background with a gray header bar containing a "Calcular" button and some text fields.
- Right Panel:** A large yellow box contains text in Portuguese explaining layout concepts like padding, margins, and constraints.

Para o botão, eu ligo a parte de cima a parte de baixo do campo das notas, e ligo a esquerda e a direita. O campo deveria ficar centralizado como o campo acima, mas ele esta pegando todo o tamanho da tela. Isso porque a largura esta definida como match_parent. Para desgrudar das magrgens podemos colocar uma margem esquerda e uma direita de uns 32dp

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]

Calculodemedia > app > src > main > res > layout > activity_main.xml

MainActivity.java colors.xml

Project Resource Manager Structure Bookmarks Build Variants

activity_main.xml

88 app:layout_constraintEnd_toEndOf="parent" 12 14 ^ ▾
89 app:layout_constraintStart_toStartOf="parent"
90 app:layout_constraintTop_toBottomOf="@+id/nota3" />
91
92 <Button
93 android:id="@+id/btn_calcular"
94 android:layout_width="match_parent"
95 android:layout_height="wrap_content"
96 android:layout_below="@+id/numeroFalta"
97 android:backgroundTint="@color/gray"
98 android:padding="20dp"
99 android:text="Calcular"
100 android:textAllCaps="false"
101 android:textSize="18sp"
102 android:textStyle="bold"
103 android:layout_marginEnd="32dp"
104 android:layout_marginStart="32dp"
105 app:layout_constraintEnd_toEndOf="parent"
106 app:layout_constraintStart_toStartOf="parent"
107 android:layout_marginTop="50dp"
108 app:layout_constraintTop_toBottomOf="@+id/numeroFalta" />
109

Component Tree

Palette

activity_main.xml Pixel 34

Nota 1 Nota 2
Nota 3 Nota 4
Numero de faltas
Calcular
Aluno foi Aprovado

Code Split Design

Attributes

Barcode

Emulator

Event Log Layout Inspector

92:5 LF UTF-8 4 spaces

Gradle sync finished in 2 m 44 s 258 ms (today 09:59)

Agora o botão ficou mais legal sem estar colado nas extremidades do layout

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help.
- Title Bar:** Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]
- Project Tab:** Calculodemedia > app > src > main > res > layout > activity_main.xml
- Code Editor:** Shows the XML code for activity_main.xml. A specific line is highlighted:

```
    android:layout_marginEnd="32dp"
    android:layout_marginStart="32dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="50dp"
    app:layout_constraintTop_toBottomOf="@+id/numeroFalta" />
```
- Design Tab:** Shows the layout preview. It consists of four gray boxes labeled "Nota 1", "Nota 2", "Nota 3", and "Nota 4" arranged in a 2x2 grid. Below them is a gray box labeled "Numero de faltas". At the bottom is a large dark gray button labeled "Calcular". A blue dashed line with arrows indicates a constraint from the "Aluno foi Aprovado" TextView to the "Calcular" button, with a vertical offset of 50dp.
- Bottom Bar:** Version Control, TODO, Problems, Terminal, Logcat, App Inspection, Build, Profiler, Event Log, Layout Inspector, Gradle sync status.

O TextView vai ficar ligado pela parte de cima ao botão e ligo a esquerda e a direita, mas ele não estica porque o comprimento esta definido como wrap_content. Então ele fica centralizado

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Calculo de medias - activity_main.xml [Calculo_de_medias.app.main]

Calculodemedia > app > src > main > res > layout > activity_main.xml

MainActivity.java colors.xml

Project Resource Manager Structure Bookmarks Build Variants

activity_main.xml

88 app:layout_constraintEnd_toEndOf="parent" 12 14 ^ ▾
89 app:layout_constraintStart_toStartOf="parent"
90 app:layout_constraintTop_toBottomOf="@+id/nota3" />
91
92 <Button
93 android:id="@+id/btn_calcular"
94 android:layout_width="match_parent"
95 android:layout_height="wrap_content"
96 android:layout_below="@+id/numeroFalta"
97 android:backgroundTint="@color/gray"
98 android:padding="20dp"
99 android:text="Calcular"
100 android:textAllCaps="false"
101 android:textSize="18sp"
102 android:textStyle="bold"
103 android:layout_marginEnd="32dp"
104 android:layout_marginStart="32dp"
105 app:layout_constraintEnd_toEndOf="parent"
106 app:layout_constraintStart_toStartOf="parent"
107 android:layout_marginTop="50dp"
108 app:layout_constraintTop_toBottomOf="@+id/numeroFalta" />
109

Component Tree

Palette

activity_main.xml Pixel 34

Nota 1 Nota 2
Nota 3 Nota 4
Numero de faltas
Calcular
Aluno foi Aprovado

Code Split Design

Attributes

Barcode

Emulator

Event Log Layout Inspector

92:5 LF UTF-8 4 spaces

Gradle sync finished in 2 m 44 s 258 ms (today 09:59)

Agora o botão ficou mais legal sem estar colado nas extremidades do layout

Tipos de medias para os elementos de visualizacao

Existem três tipos de medidas que podemos utilizar para configurar os elementos e essas medidas podem ser usadas tanto na horizontal quanto na vertical:

`Wrap_content`: nesse tipo de medida o tamanho do elemento se ajusta ao tamanho do conteúdo. Entao se temos um texto pequeno dentro do elemento, o elemento também fica pequeno. Aumentando o texto, o elemento se ajuta para caber.

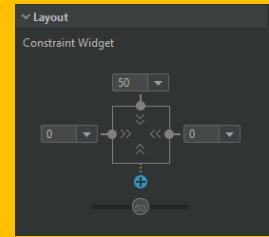
`Match_parent`: Nesse tipo de medida, o elemento ocupa todo espaço disponível do dispositivo.

`Fixed`: tamanho fixo.

No modo design podemos ver três tipos de ícones para as medidas:

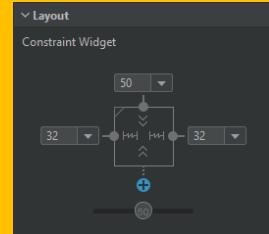
Existem três tipos de medidas que podemos utilizar para configurar os elementos e essas medidas podem ser usadas tanto na horizontal quanto na vertical:

Wrap_content



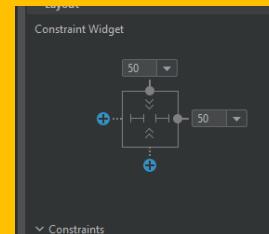
Aqui o ícone eh duas setas na horizontal e duas na vertical o que indica que os dois estão ajustáveis

Match_parent:



Aqui o ícone eh como uma mola indicando que ele vai se ajustar ao dispositivo não ao elemento

Fixed:



Aqui temos uma medida fixa e o ícone mostra um traco delimitado

Passos para criação de um app:

No arquivo xml:

=====

1. Criação da tela (layout)
2. Identificação dos elementos (colocar um id em cada elemento)

Na Classe:

=====

1. Para cada elemento do layout, declarar um elemento do mesmo tipo na classe
2. Ligar o layout com a classe através do método findViewById()
3. Adicionar evento de clique para os botões.
4. Desenvolver a logica. (A logica será desenvolvida dentro do método onCreate, que é o primeiro método que roda ao abrirmos um app.

The screenshot shows the Android Studio code editor with the file `MainActivity.java` open. The code defines a class `MainActivity` that extends `AppCompatActivity`. The class contains private fields for four edit texts (`nota1`, `nota2`, `nota3`, `nota4`), one button (`btnCalcular`), and one text view (`resultado`). It also has a private string field `nome`. A tooltip at the bottom of the code editor shows options like `Extract` and `Surround`.

```
package br.com.fabioclaret.calculodemedia;
import ...;
public class MainActivity extends AppCompatActivity {
    private EditText nota1, nota2, nota3, nota4, numeroFaltas;
    private Button btnCalcular;
    private TextView resultado;
    private String nome;
}
```

Logo no inicio da classe, vamos declarar todos os elementos do layout que iremos usar. Esses elementos são classes.
E toda classe sempre começa com letra maiúscula
Aqui temos a classe `MainActivity` que esta herdando da classe `AppCompatActivity`

The screenshot shows the Android Studio interface with the file `MainActivity.java` open. The code is as follows:

```
public class MainActivity extends AppCompatActivity {
    private Toolbar toolbar;
    private DrawerLayout drawerLayout;
    private NavigationView navigationView;
    private EdgeToEdge edgeToEdge;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable($this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);

        initComponentes();
    }

    private void initComponentes() {
        WindowInsetsCompat insets = v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
    }
}
```

A tooltip appears over the `initComponentes()` method, suggesting to "Create method 'initComponentes' in 'MainActivity'" and "Rename reference". The status bar at the bottom shows the file path: `CalculodeMedia > app > src > main > java > br > com > fabioclarret > calculodemedia > MainActivity > onCreate`, and the time: 28:19.

Aqui temos o método `onCreate`, esse método é abstrato na classe `AppCompatActivity`, então aqui temos que sobrescrevê-lo. E dentro dele (após o `setContentView`) vamos criar um método `initComponents()`. Metodos e variáveis são sempre escritos com letra minúscula no inicio. Para criar basta escrever como esta aqui: `initComponents();` Ele vai ficar vermelho, então clicamos na lâmpada e pedimos para criar o método.

The screenshot shows the Android Studio interface with the file `MainActivity.java` open. The code is as follows:

```
14     public class MainActivity extends AppCompatActivity {
15         ...
16
17         ...
18
19         ...
20         protected void onCreate(Bundle savedInstanceState) {
21             ...
22             initComponentes();
23
24             super.onCreate(savedInstanceState);
25             EdgeToEdge.enable( $this$enableEdgeToEdge: this );
26             setContentView(R.layout.activity_main);
27             ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), ( View v, WindowInsetsCompat insets) -> {
28                 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
29                 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
30                 return insets;
31             });
32         }
33
34         private void initComponentes() {
35
36     }
37 }
```

A context menu is open at the bottom of the code editor, showing options: Extract, Surround, and others.

Esse método será criado aqui dentro da `MainActivity`, depois do método `onCreate`. Todo método e classe sempre tem um escopo, que é o bloco entre chaves que delimitam esse métodos e classes.

The screenshot shows the Android Studio interface with the file `MainActivity.java` open. The code is as follows:

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, WindowInsetsCompat insets) -> {
            insets.getSystemBars() = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}

private void initComponentes() {
    nota1      = findViewById(R.id.nota1);
    nota2      = findViewById(R.id.nota2);
    nota3      = findViewById(R.id.nota3);
    nota4      = findViewById(R.id.nota4);
    numeroFaltas = findViewById(R.id.numero_falta);
    btnCalcular = findViewById(R.id.btn_calcular);
    resultado   = findViewById(R.id.txt_resultado);
}
```

The code is annotated with several icons from the Java editor's tool palette, including a lightbulb, a blue diamond, a red circle, and a question mark.

Nesse método, a gente vai ligar todos os elementos do layout com a classe. Então criamos a variável `nota1` que é do tipo `EditText` para receber o valor que será digitado lá no layout no campo `nota1`. `findViewById()` é o método que localiza um elemento pelo seu id, por isso temos que colocar o id nos elementos lá no layout.

The screenshot shows the Android Studio interface with the code editor open to `MainActivity.java`. The code is as follows:

```
14     public class MainActivity extends AppCompatActivity {
15         private String nome,
16
17         @Override
18         protected void onCreate(Bundle savedInstanceState) {
19             initComponentes();
20
21             super.onCreate(savedInstanceState);
22             EdgeToEdge.enable($this$enableEdgeToEdge: this);
23             setContentView(R.layout.activity_main);
24             ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, WindowInsetsCompat insets) -> {
25                 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
26                 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
27                 return insets;
28             });
29         }
30     }
31
32     private void initComponentes() {
33         nota1      = findViewById(R.id.nota1);
34         nota2      = findViewById(R.id.nota2);
35         nota3      = findViewById(R.id.nota3);
36         nota4      = findViewById(R.id.nota4);
37     }
38
39
40 }
```

The line `initComponentes();` at line 24 is highlighted with a red circle and a red bar underlining it. A yellow callout box on the right side of the screen contains the following text:

Aqui então na linha 25 o programa desvia para o método `initComponents`, que vai ligar o layout com a classe, depois volta na linha abaixo que eh a 26

The screenshot shows the Android Studio interface with the code editor open to MainActivity.java. The code is as follows:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        initComponentes();

        btnCalcular.setOnClickListener( view->{
            resultado.setText("Voce clicou no botao calcular");
        });
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), ( View v, WindowInsetsCompat insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
    private void initComponentes() {
```

Fizemos os passos 1, 2 e 3, agora vamos criar o evento de click do botão calcular

Fazemos isso dentro do método onCreate.

Esse método tem que ficar depois do setContentView que é o método que desenha a tela.

The screenshot shows the Android Studio interface with the code editor open to `MainActivity.java`. The code is as follows:

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable($this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);

        initComponentes();

        btnCalcular.setOnClickListener(view -> {
            validaCampos();
        });
    }

    private void validaCampos() {
        getInsets(WindowInsetsCompat.Type.systemBars());
        return insets;
    }

    private void initComponentes() {
    }
}
```

A tooltip is displayed over the `validaCampos()` method, showing options: "Create method 'validaCampos' in 'MainActivity'", "Rename reference", "Extract to method reference", and "Press Ctrl+Q to toggle preview".

Quando a gente programa, temos que pensar em quem vai usar o app (usuário). E temos que prever que ele pode fazer algo errado, e contornar o erro.

Aqui a gente vai calcular a media, então todos os campos de nota tem que estar preenchidos.

Temos que garantir isso, criando um método para forçar o usuário digitar tudo.

Fazemos isso criando o `validaCampos` e pedindo para a IDE criar.

The screenshot shows the Android Studio interface. On the left is the code editor with `MainActivity.java` open. The code implements a `validateFields` method that checks if four text fields (`nota1`, `nota2`, `nota3`, `nota4`) are empty. If any field is empty, it sets an error message on the corresponding `EditText`. On the right is the Android emulator displaying a simple UI with four floating input fields labeled `Nota 1`, `Nota 2`, `Nota 3`, and `Nota 4`. Below them is a `Calcular` button and a `Número de Faltas` input field.

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, WindowInsets insets) {
            return insets;
        });
    }
}

private void validaCampos() {
    if (TextUtils.isEmpty(nota1.getText())) {
        // O campo está vazio ou contém apenas espaços em branco
        // Ex: Exiba uma mensagem de erro para o usuário!
        nota1.setError("Este campo não pode estar vazio.");
    } else if(TextUtils.isEmpty(nota2.getText())){
        nota2.setError("Este campo não pode estar vazio.");
    }else if(TextUtils.isEmpty(nota3.getText())){
        nota3.setError("Este campo não pode estar vazio.");
    }else if(TextUtils.isEmpty(nota3)){
        nota3.setError("Este campo não pode estar vazio.");
    }else if(TextUtils.isEmpty(nota4.getText())){
        nota4.setError("Este campo não pode estar vazio.");
    }
}
```

A classe `TextUtils.isEmpty` checa se o campo esta vazio ou com espaços.
Se estiver, ao clicar no botão calcular, ele não vai e fica esperando ate que todos os campos estejam preenchidos

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** Calculo de Média, main, Pixel_3a_API_35...
- MainActivity.java:** The code is as follows:

```
15     public class MainActivity extends AppCompatActivity {
16         protected void onCreate(Bundle savedInstanceState) {
17             super.onCreate(savedInstanceState);
18             setContentView(R.layout.activity_main);
19             initComponentes();
20
21             btnCalcular.setOnClickListener(view -> {
22                 validaCampos();
23                 calculaMedia();
24             });
25         }
26
27         private void calculaMedia() {
28             int nota1 = Integer.parseInt(editNota1.getText().toString());
29             int nota2 = Integer.parseInt(editNota2.getText().toString());
30             int nota3 = Integer.parseInt(editNota3.getText().toString());
31             int nota4 = Integer.parseInt(editNota4.getText().toString());
32
33             int numeroDeFaltas = Integer.parseInt(editFaltas.getText().toString());
34
35             float media = (nota1 + nota2 + nota3 + nota4) / 4;
36
37             String resultado = String.format("A média é: %.2f", media);
38
39             resultadoText.setText(resultado);
40
41             resultadoText.setMovementMethod(new ScrollingMovementMethod());
42
43             resultadoText.setEllipsize(TextUtils.TruncateAt.END);
44
45         }
46
47         private void validaCampos() {
48             if (TextUtils.isEmpty(editNota1.getText())) {
49                 // O campo está vazio ou contém apenas espaços em branco
50                 // Ex: Exiba uma mensagem de erro para o usuário
51                 editNota1.setError("Este campo não pode estar vazio.");
52             }
53         }
54     }
```

A context menu is open at the line 33, showing options like "Create method 'calculaMedia' in 'MainActivity'", "Rename reference", "Extract to method reference", and "Press Ctrl+Q to toggle preview".

Preview: A smartphone screen displays four input fields (each containing '1') and a button labeled 'Calcular'. Below the screen, there are navigation icons and a zoom level indicator.

Digitamos o nome do método para calcular a media e pedimos para o Android criar o metodo

The screenshot shows the Android Studio interface. The left pane displays the Java code for `MainActivity.java`. The code includes logic for handling window insets and calculating a media grade. The right pane shows a preview of the app running on a Pixel 3a API 35 emulator, displaying four input fields, a button labeled "Calcular", and a text input field labeled "Número de Faltas".

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        ...
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, Insets insets) -> {
            insets.setSystemBarsInset(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }

    private void calculaMedia() {
        double n1 = Double.parseDouble(notas1.getText().toString());
        double n2 = Double.parseDouble(notas2.getText().toString());
        double n3 = Double.parseDouble(notas3.getText().toString());
        double n4 = Double.parseDouble(notas4.getText().toString());
    }

    private void validaCampos() {
        if (TextUtils.isEmpty(notas1.getText())) {
            // O campo está vazio ou contém apenas espaços em branco
        }
    }
}
```

Toda vez que digitamos qualquer numero em uma caixa de texto, ele vem como texto, então temos que converter para o que queremos. E aqui queremos números do tipo double, pois podem ter decimais. Esse processo de converter se chama casting

Aqui eu calculo a media

The screenshot shows the Android Studio interface. On the left, the code editor displays `MainActivity.java` with the following code:

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        ...
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (View v, Insets insets) -> {
            insets.setSystemBars(Insets.Type.systemBars);
            v.setPadding(insets.left, insets.top, insets.right, insets.bottom);
            return insets;
        });
    }

    private void calculaMedia() {
        double n1 = Double.parseDouble(notas1.getText().toString());
        double n2 = Double.parseDouble(notas2.getText().toString());
        double n3 = Double.parseDouble(notas3.getText().toString());
        double n4 = Double.parseDouble(notas4.getText().toString());

        double media = (n1 + n2 + n3 + n4) / 4;
    }
}
```

The right side of the screen shows the application running on a virtual device (Pixel 3a API 35). The UI consists of four input fields containing the number 1, a text input field labeled "Número de Faltas" with the value 0, and a button labeled "Calcular".

The screenshot shows the Android Studio interface with the file `MainActivity.java` open. The code implements a method `calculaMedia()` to calculate the average of four input numbers and determine the student's status based on the average and the number of absences.

```
public class MainActivity extends AppCompatActivity {
    private void calculaMedia() {
        double n1 = Double.parseDouble(nota1.getText().toString());
        double n2 = Double.parseDouble(nota2.getText().toString());
        double n3 = Double.parseDouble(nota3.getText().toString());
        double n4 = Double.parseDouble(nota4.getText().toString());
        double media = (n1 + n2 + n3 + n4) / 4;
        double faltas = Double.parseDouble(numeroFaltas.getText().toString());

        if( media > 7 ){
            if( faltas < 20 ) {
                resultado.setTextColor(Color.parseColor( colorString: "#437845" ));
                resultado.setText("Aluno Aprovado com media " + media);
            }else{
                resultado.setTextColor(Color.parseColor( colorString: "#F44336" ));
                resultado.setText("Excesso de falta " + faltas);
            }
        }else{
            resultado.setTextColor(Color.parseColor( colorString: "#F44336" ));
            resultado.setText("Aluno Retido com media " + media);
        }
    }
}
```

Aqui é pura logica:
Se a media eh maior que 7 e o numero de falta menor que 20, aprovado.
Se a media eh maior que 7 mas o numero de falta eh maior que 20, reprovado
Se media eh menor que 7, aluno reprovado por media
E se o numero de falta for maior que 20, aluno retido por falta

The screenshot shows the Android Studio interface with the file `MainActivity.java` open. The code implements a validation method `validaCampos2` that checks if four text fields (`nota1`, `nota2`, `nota3`, `nota4`) are empty. If any field is empty, it sets an error message on the `nota4` field. The code then returns a boolean indicating if all fields were valid.

```
public class MainActivity extends AppCompatActivity {
    private void validaCampos() {
        nota4.setError("Este campo não pode estar vazio.");
    }

    private boolean validaCampos2(){
        boolean camposValidados = true;
        if( nota1.getText().toString().isEmpty() ){
            camposValidados = false;
        }else if( nota2.getText().toString().isEmpty() ){
            camposValidados = false;
        } else if( nota3.getText().toString().isEmpty() ){
            camposValidados = false;
        }else if( nota4.getText().toString().isEmpty() ){
            camposValidados = false;
        }

        return camposValidados;
    }
}
```

Outro jeito de validar campos, crio uma variável verdadeira e se algum campo for vazio eu torno essa variável falsa. E no final, retorno a variável. La em cima, basta checar se o `validaCampos2` eh verdadeiro ou falso.

The screenshot shows the Android Studio interface with the file `MainActivity.java` open. The code implements a method `validaCampos3()` that checks if four EditText fields (`nota1`, `nota2`, `nota3`, `nota4`) are empty. The code is highlighted with a blue selection bar.

```
public class MainActivity extends AppCompatActivity {  
    public boolean validaCampos3(){  
        return nota1.getText().toString().isEmpty()  
            && nota2.getText().toString().isEmpty()  
            && nota3.getText().toString().isEmpty()  
            && nota4.getText().toString().isEmpty();  
    }  
}
```

Aqui otimizei a logica para não precisar de usar o if

Atividade

Otimize esse app para pedir o nome do aluno.

Apresentar lo nome do aluno e se passou ou não e ainda media e faltas

Altere o layout para deixar mais atrativo, mudando cores, formatos, etc

Apostila para o curso de PAMII

Professores:
Fabio Claret
Cleiton Silva