**Aula XVIII e XIX - DevAppII**

**Componentes de Formulários**

**TextField**

**TextFormField**

**TextEditingController**

**Checkbox**

**CheckBoxTile**

**RadioButton**

**Validação**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Fontes:***

<https://flutter.dev/>

<https://flutterando.com.br/>

<https://www.woolha.com/tutorials/flutter-using-checkboxlisttile-widget-examples>

<https://api.flutter.dev/flutter/material/CheckboxListTile-class.html>

<https://www.woolha.com/tutorials/flutter-creating-custom-radio-button-examples>

<https://imasters.com.br/mobile/flutter-parte-01-apresentando-widgets>

<https://www.woolha.com/tutorials/flutter-using-texteditingcontroller-examples>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Componentes de Formulário**

Os aplicativos geralmente exigem que os usuários insiram informações em um campo de texto. Por exemplo, podemos exigir que os usuários façam login com uma combinação de endereço de e-mail e senha.

Para tornar os aplicativos seguros e fáceis de usar, sempre verificamos se as informações fornecidas pelo usuário são válidas. Se o usuário preencheu corretamente o formulário, processe as informações. Se o usuário enviar informações incorretas, exiba uma mensagem de erro amigável informando o que deu errado.

Primeiro Código possui os seguintes objetivos:

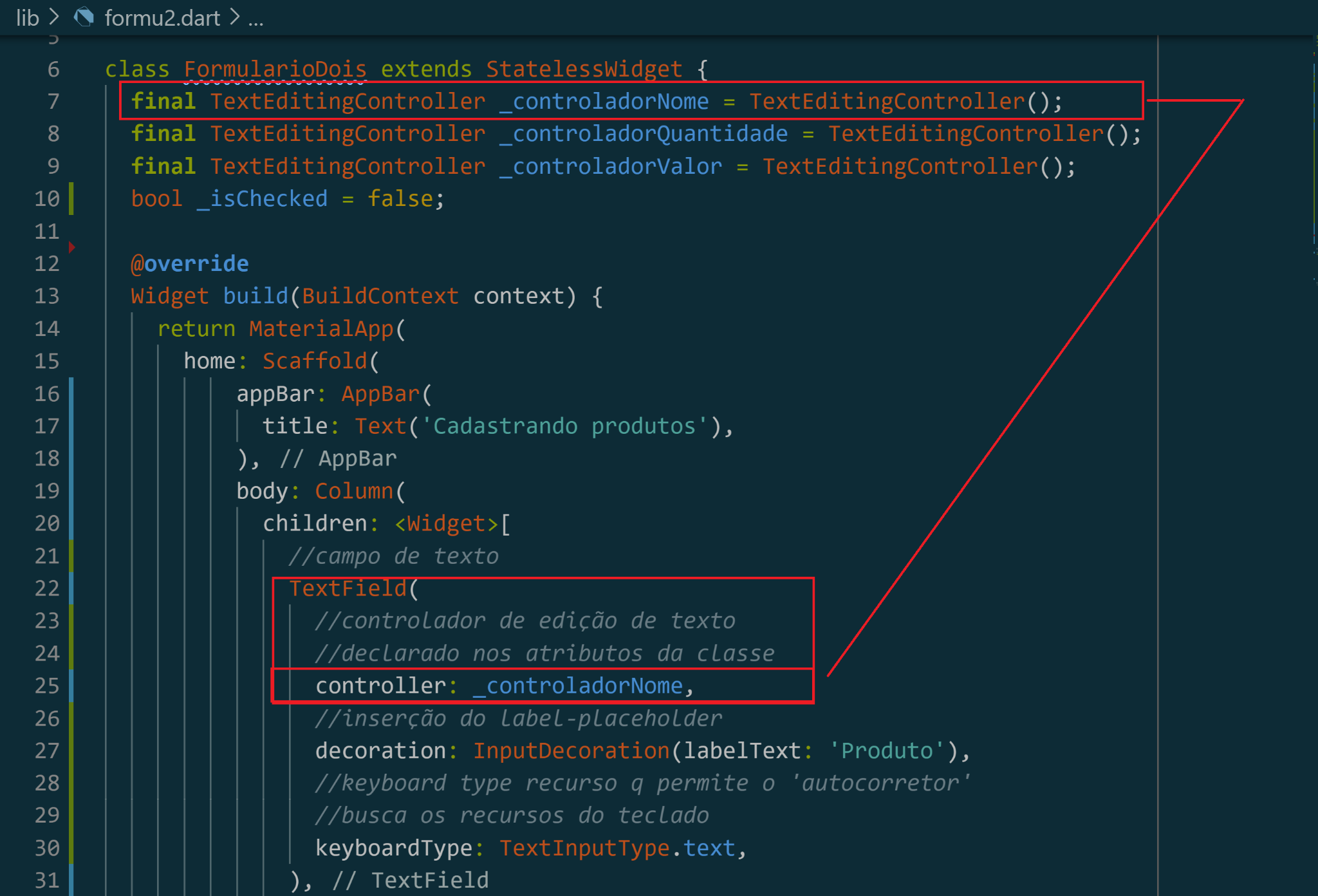
1. **Criar um Form com um GlobalKey.**
2. **Adicionar um TextFormField com lógica de validação.**
3. **Criar um botão para validar e enviar o formulário.**

Prática:

**TextEditingController**

É o widget responsável por controlar as edições em um TextField ou TextFormField.

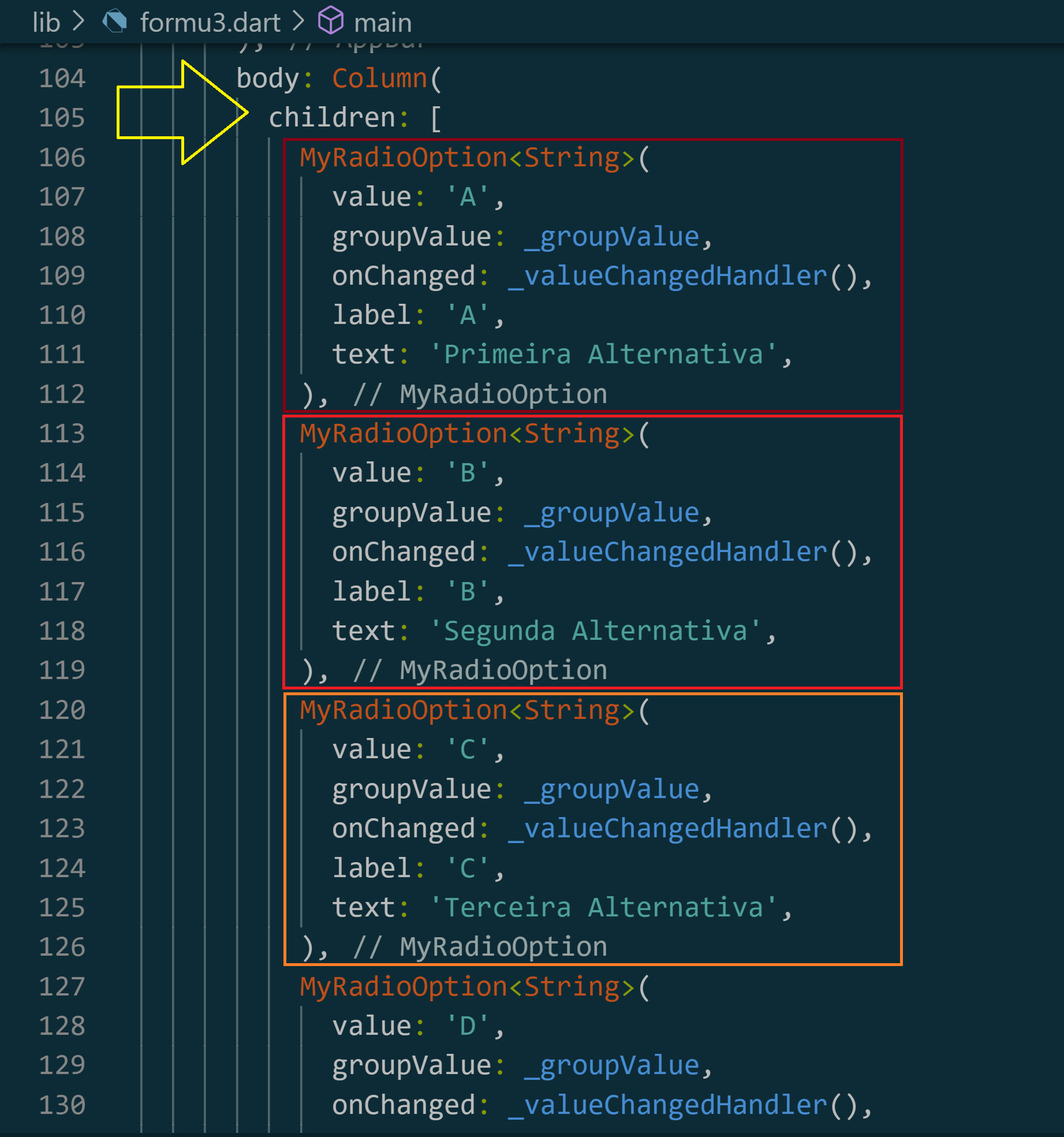
**Exemplo:**



**RadioButton**

**Radiowidget** para RadioButton é um botão de opção criado com esse widget que consiste em um círculo externo vazio e um círculo interno sólido, sendo o último mostrado apenas no estado selecionado. Em alguns casos, podemos criar um grupo de rádio cujas opções usem um design personalizado, não o botão de rádio convencional.

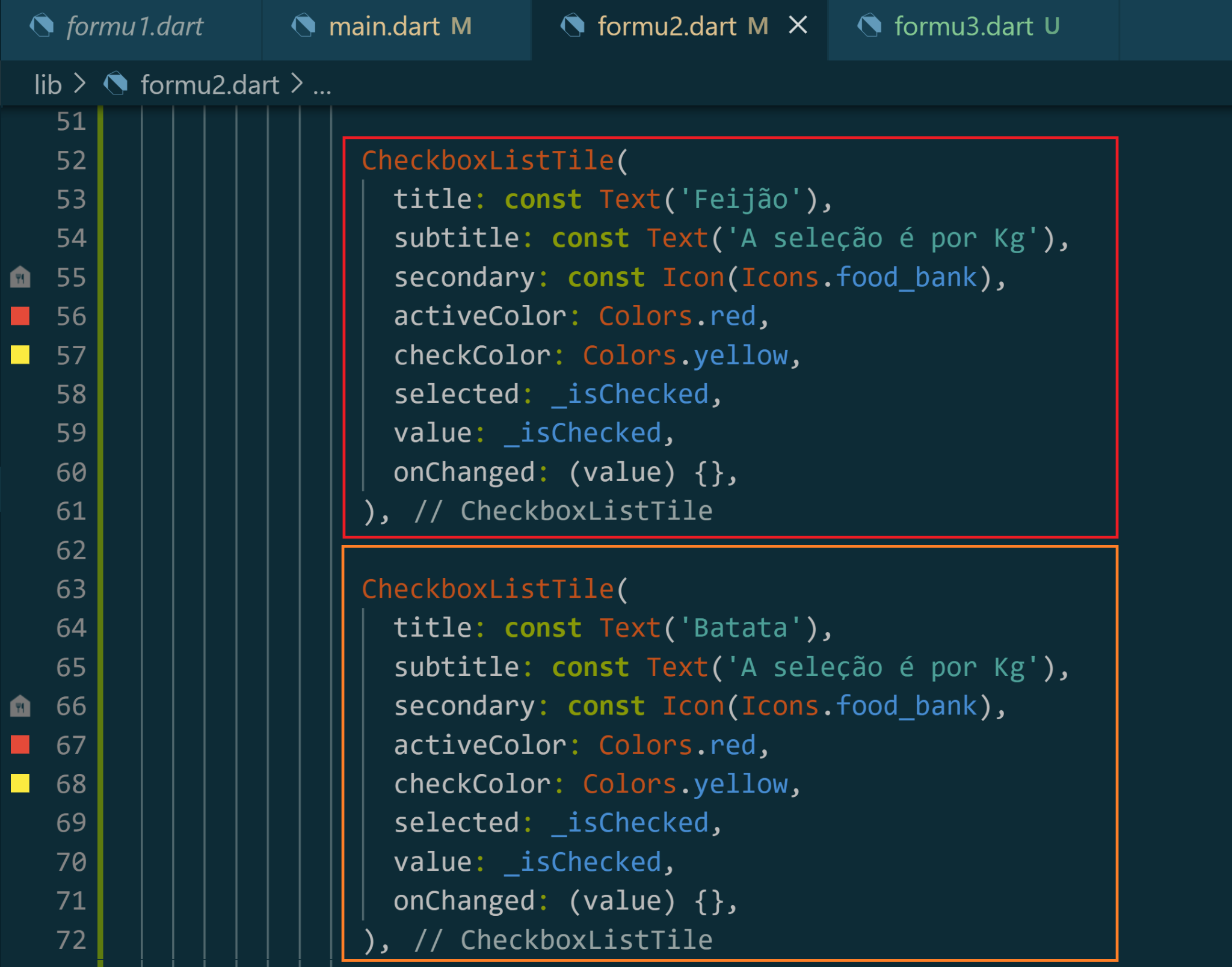
**Exemplo:**



**CheckBox e CheckboxListTile:**

O CheckBox já conhecemos, ele não muda de acordo com a linguagem utilizada e o Flutter tem um **widget chamado Checkbox** para esse fim. No entanto, em muitos casos, também queremos adicionar texto ou ícone à caixa de seleção. Se quisermos usar o Checkbox com ícones e textos, então temos o **CheckboxListTile,** ele é um widget que combina uma caixa de seleção e um bloco de lista. Ele permite a criação de uma caixa de seleção junto com o título, subtítulo e ícone sem a necessidade de criar um widget separado para cada parte.

**Exemplo:**



**Propriedades**

Aqui está a lista de propriedades que CheckboxListTile que podemos passar no construtor.

1. **Key key:** A chave do widget, usada para controlar se deve ser substituída.
2. **bool value\*:** Se true, a caixa de seleção está marcada. Se false, a caixa de seleção está desmarcada.
3. **void Function(bool) onChanged \*:** Uma função de retorno de chamada que será chamada quando o usuário marcar ou desmarcar a caixa de seleção.
4. **Color activeColor :** A cor quando a caixa de seleção está marcada.
5. **Color checkColor :** A cor do ícone de seleção quando a caixa de seleção está marcada.
6. **Widget title :** O conteúdo principal do bloco da lista.
7. **Widget subtitle :** Conteúdo adicional abaixo do título.
8. **Widget secondary :** Um widget a ser exibido no lado oposto da caixa de seleção.
9. **bool isThreeLine:** Se este bloco de lista se destina a exibir três linhas de texto. O padrão é false.
10. **bool dense :** Se este bloco de lista faz parte de uma lista verticalmente densa.
11. **bool selected:** Se deseja renderizar ícones e texto no activeColor. O padrão é false.
12. **ListTileControlAffinity controlAffinity:** Onde colocar o controle em relação ao texto. O padrão é **ListTileControlAffinity.platform.**

**\*: requeridos**