Exemplo 1 – Cria uma janela vazia

O ciclo de vida da aplicação é controlado pela classe MeuApp, responsável por controlar toda a execução do aplicativo.

```
from kivymd.app import MDApp
```

MDApp é a classe base para todas as aplicações KivyMD. Ela gerencia o ciclo de vida da aplicação.

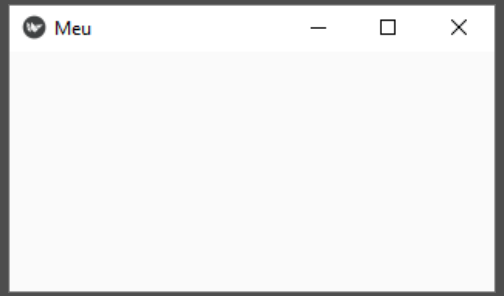
```
class MeuApp(MDApp):
    pass
```

A classe MeuApp herda de MDApp

```
MeuApp().run()
```

Instancia a classe MeuApp e, ao mesmo tempo, chama o método run(), que está definido na classe base MDApp.

O KivyMD, por padrão, utiliza o nome da classe para gerar o título da janela. Ele pega o nome da classe, remove o sufixo App e usa o restante como o título da janela. Como a classe se chama MeuApp, o título da janela acaba sendo "Meu".



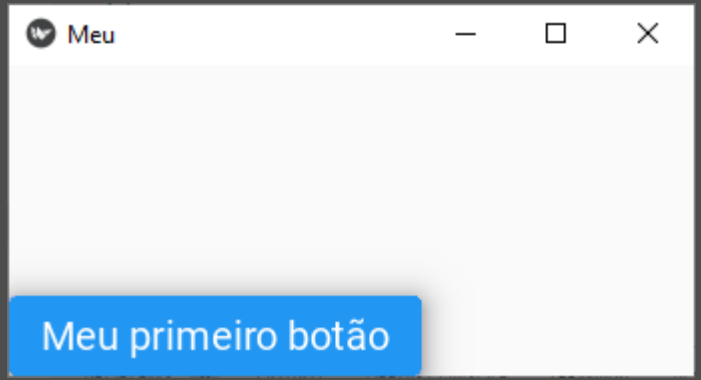
Exemplo 2 – Instancia um objeto da classe Button.

Além de gerenciar o ciclo de vida da aplicação, o método run() é responsável por chamar o método build() para construir a interface do usuário.

```
from kivymd.app import MDApp
from kivymd.uix.button import MDRaisedButton

class MeuApp(MDApp):
    def build(self):
        objetoBotao = MDRaisedButton()
        objetoBotao.text = 'Meu primeiro botão'
        objetoBotao.color = 'yellow'
        objetoBotao.font_size = 20
        #objetoBotao.disabled = True
        return objetoBotao

MeuApp().run()
```

Instancia um objeto da Classe MDRaisedButton.

O método build() retorna o botão criado, que será exibido na interface do aplicativo.



Exemplo 3 – Inserindo outro widget (label).

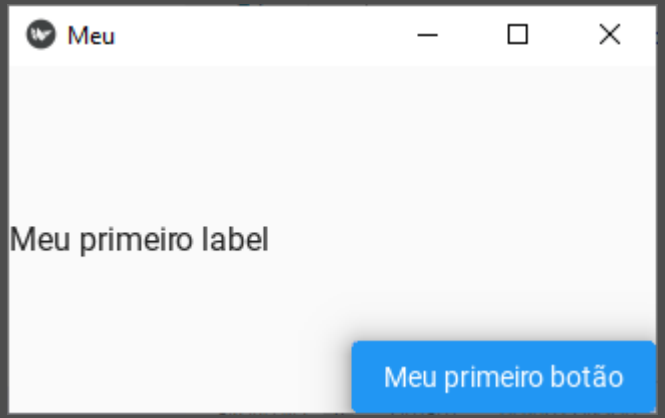
```
from kivymd.app import MDApp
from kivymd.uix.button import MDRaisedButton
from kivymd.uix.label import MDLabel

class MeuApp(MDApp):
    def build(self):
        objetoBotao = MDRaisedButton(text='Meu primeiro botão')
        objetoLabel = MDLabel(text='Meu primeiro label')
        return objetoLabel

MeuApp().run()
```

O problema é que não conseguimos retornar o objetoBotao e o objetoLabel na janela.

Mas antes de corrigir esse problema, observe como o objetoBotao foi instanciado.

```
from kivymd.app import MDApp
from kivymd.uix.button import MDRaisedButton
from kivymd.uix.label import MDLabel
from kivymd.uix.boxlayout import MDBoxLayout

class MeuApp(MDApp):
    def build(self):
        objetoBotao = MDRaisedButton(text='Meu primeiro botão')
        objetoLabel = MDLabel(text = 'Meu primeiro label')

        objetoBoxLayout = MDBoxLayout()
        objetoBoxLayout.add_widget(objetoLabel)
        objetoBoxLayout.add_widget(objetoBotao)

        return objetoBoxLayout

MeuApp().run()
```

Exemplo 4 – Para inserir vários *widgets* dentro de uma mesma janela, precisamos trabalhar com uma espécie de *container* (não visível) que irá abrigar esse *widgets*. No final, retornamos o *container*.



```
from kivymd.app import MDApp
from kivymd.uix.button import MDRaisedButton
from kivymd.uix.label import MDLabel
from kivymd.uix.textfield import MDTextField
from kivymd.uix.boxlayout import MDBoxLayout
from kivy.core.window import Window

class ManipulaJanela:
    def __init__(self, largura, altura):
        self.largura = largura
        self.altura = altura

    def ajustar_tamanho_janela(self):
        Window.size = (self.largura, self.altura)
```

Criei a classe ManipulaJanela() para permitir a alteração das dimensões da janela do aplicativo.

É necessário importar a classe Window para efetuar o redimensionamento.

```
class MeuApp(MDApp):
    def build(self):
        manipulador = ManipulaJanela(400, 600)
        manipulador.ajustar_tamanho_janela()

        layout = MDBoxLayout(orientation='vertical', padding=20, spacing=20 )
        self.numero1  = MDTextField( hint_text="Digite o primeiro número", input_filter="int" )

        self.numero2 = MDTextField( hint_text="Digite o segundo número",input_filter="float" )

        botao_somar = MDRaisedButton( text="Somar", on_release=self.somar_numeros )
        self.resultado = MDLabel( text="Resultado: ", halign="center" )

        layout.add_widget(self.numero1 )
        layout.add_widget(self.numero2)
        layout.add_widget(botao_somar)
        layout.add_widget(self.resultado)

        return layout
```

```
    def somar_numeros(self, instance):
        try:
            num1 = int(self.numero1.text)
            num2 = float(self.numero2.text)
            soma = num1 + num2
            self.resultado.text = f"Resultado: {soma}"
        except ValueError:
            self.resultado.text = "Por favor, insira números válidos"

MeuApp().run()
```