

ECM251 - Linguagens I

Interfaces Gráficas com o Usuário - Versão II
Prof. Murilo Zanini de Carvalho



Objetivos da aula de hoje

- Compreender o conceito do Streamlit
- Construir interfaces com o usuário utilizando o Streamlit
- Utilizar um padrão MVC
- Compreender o padrão MVC para contruir aplicações
- Iniciar o estudo de padrões de projeto

Um problema



Retirado de
(<https://images.wondershare.com/mockitt/ui-design/ui-design-examples-too-much-information.jpg>), em 13/09/2022

Um Problema

- Utilizando o tkinter, ou mesmo o ttkbootstrap, estamos tendo problemas com nossa interface. Ela não está ficando da forma como desejamos e ainda a documentação para desenvolver o projeto está bastante escassa.
- Exemplos de Bad UI:
<https://mattw.io/bad-ui/>



Retirado de
(https://pa1.narvii.com/6908/754577384a2301c5d135e6a607e7d75c75e46698r1-500-240_hq.gif), em 13/09/2022

Introdução ao StreamLit

Introdução ao StreamLit

A faster way to build and share data apps

Streamlit turns data scripts into shareable web apps in minutes.

All in pure Python. No front-end experience required.

Try Streamlit now

Sign up for **Streamlit Community Cloud**

<https://streamlit.io>

DESENVOLVER!

Hello World com o StreamLit

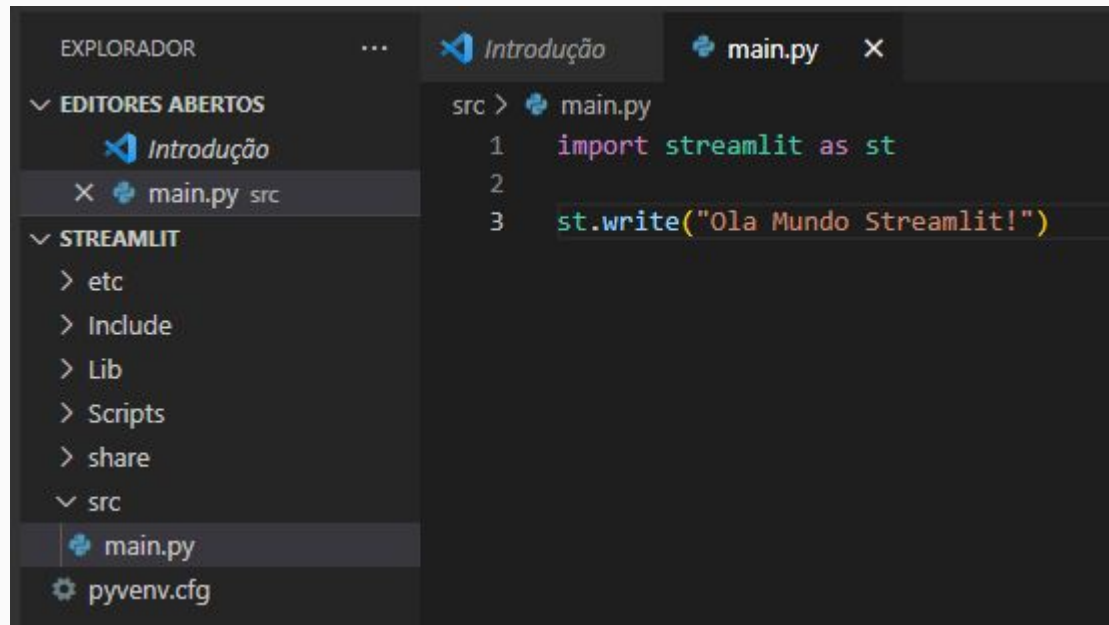
- Vamos iniciar o processo de implementar o StreamLit na aplicação.
- Criando o venv.
- Instalando a biblioteca:

```
$ pip install streamlit  
$ streamlit hello
```


IMPORTANTE:
Deixar a
documentação
aberta!

- <https://docs.streamlit.io/library/api-reference/widgets>
- <https://docs.streamlit.io/>

Hello World com o StreamLit



The image shows a screenshot of the Visual Studio Code (VS Code) editor interface. On the left, the 'EXPLORADOR' (Explorer) sidebar is visible, showing the project structure. Under 'EDITORES ABERTOS' (Open Editors), there are two tabs: 'Introdução' and 'main.py'. The 'main.py' file is selected and open in the editor. The file is located in the 'src' directory, as indicated by the path 'src > main.py' in the editor's title bar. The code in 'main.py' is as follows:

```
src > main.py
1 import streamlit as st
2
3 st.write("Ola Mundo Streamlit!")
```

The Explorer sidebar on the left shows the following structure:

- EDITORES ABERTOS
 - Introdução
 - main.py src
- STREAMLIT
 - > etc
 - > Include
 - > Lib
 - > Scripts
 - > share
 - src
 - main.py
 - pyenv.cfg

Hello World com o StreamLit

- Para executar a aplicação:

```
(streamlit) C:\Users\muril\Desktop\streamlit>streamlit run ./src/main.py
```

ATENÇÃO!

- Ao rodar uma aplicação Streamlit pela primeira vez, é necessário preencher os dados de uso da ferramenta.

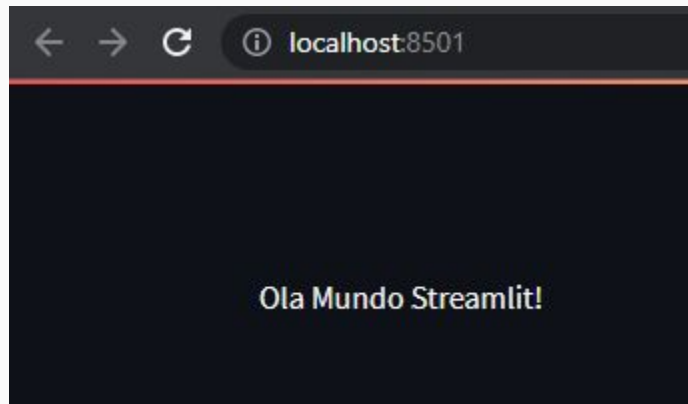
```
Welcome to Streamlit!
```

```
If you're one of our development partners or you're interested in getting  
personal technical support or Streamlit updates, please enter your email  
address below. Otherwise, you may leave the field blank.
```

```
Email: 
```

Hello World com o StreamLit

- Nossa aplicação sendo executada:



Adicionando mais alguns
componentes

Mais componentes do Streamlit

Markdown

```
1 import streamlit as st
2
3 #Utilizando um texto
4 st.write("Ola Mundo Streamlit!")
5
6 #Utilizando markdown
7 st.markdown("# Aqui tem um título!")
8 st.markdown("Ola **Mundo**!!")
```

Ola Mundo Streamlit!

Aqui tem um título!

Ola Mundo!!

Mais componentes do Streamlit

Título da Página

```
> main.py
1 import streamlit as st
2
3 #Utilizando um texto
4 st.write("Ola Mundo Streamlit!")
5
6 #Utilizando markdown
7 st.markdown("# Aqui tem um título!")
8 st.markdown("Ola **Mundo**!!")
9
10 #Define um título para a aplicação
11 st.title("Minha Página 🤖")
```

Ola Mundo Streamlit!

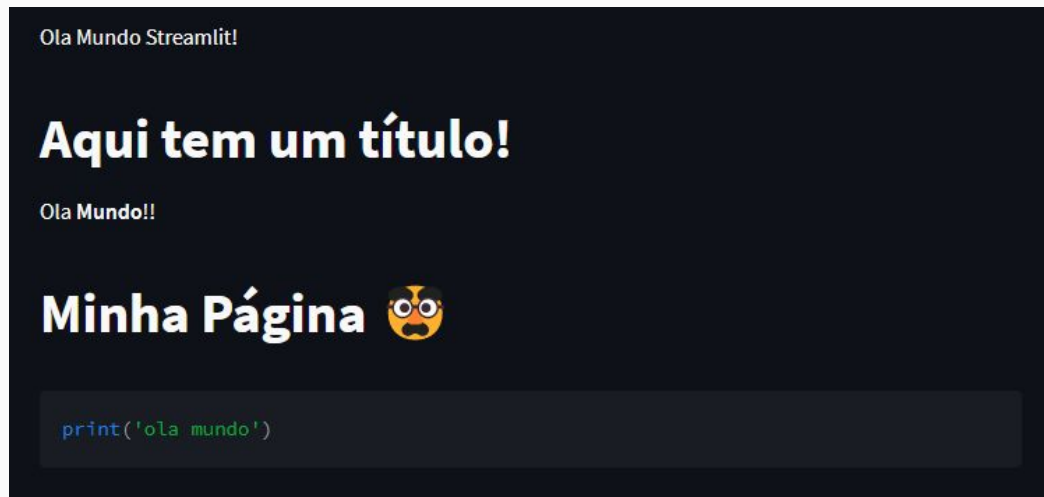
Aqui tem um título!

Ola Mundo!!

Minha Página 🤖

Mais componentes do Streamlit Código

```
1 import streamlit as st
2
3 #Utilizando um texto
4 st.write("Ola Mundo Streamlit!")
5
6 #Utilizando markdown
7 st.markdown("# Aqui tem um título!")
8 st.markdown("Ola **Mundo**!!")
9
10 #Define um título para a aplicação
11 st.title("Minha Página 🤖")
12
13 #Adiciona um bloco de código
14 st.code("print('ola mundo')", language='python')
```



Mais componentes do Streamlit

Botão

```
1 import streamlit as st
2
3 #Cria um botão na nossa aplicação
4 st.button("meu bot")
```



Mais componentes do Streamlit

Botão

```
1 import streamlit as st
2
3 #Define uma função para ser executada quando o botão for acionado
4 def ola():
5     print("Ola Mundo!")
6
7 #Cria um botão na nossa aplicação
8 st.button("meu bot", on_click=ola)
```



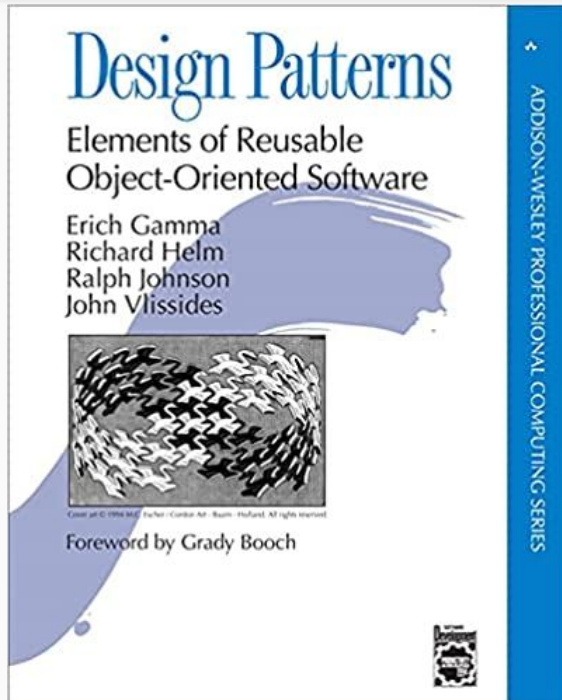
Implementar outros da
documentação!

Padrões de Projeto

Padrões de Projeto

“Padrões de projeto (design patterns) são soluções típicas para problemas comuns em projeto de software. Cada padrão é como uma planta de construção que você pode customizar para resolver um problema de projeto particular em seu código.” Retirado de [\(<https://refactoring.guru/pt-br/design-patterns/what-is-pattern>\)](https://refactoring.guru/pt-br/design-patterns/what-is-pattern), em 13/09/2022

Padrões de Projeto



Retirado de
(https://images-na.ssl-images-amazon.com/images/I/51YC9nPgbhS.X399_BO1,204,203,200.jpg), em
13/09/2022

Padrões de Projeto

English Español Français
日本語 Polski Русский
Українська 中文 Português-BR

El acabamos de reducir el precio de todos os produtos. Vamos capacitar nossas habilidades de programação para a era pós-COVID. Veja as of

Facebook Twitter

REFACTORING GURU

Ajude a Ucrânia a parar a Rússia

★ Conteúdo Premium

Padrões de Projeto

O que é um padrão

Catálogo

Padrões criacionais

Padrões estruturais

Padrões comportamentais

Exemplos de código

Refatoração em breve

Conectar Contate-nos

PADRÕES de PROJETO

Padrões de projeto (design patterns) são soluções típicas para problemas comuns em projeto de software. Cada padrão é como uma planta de construção que você pode customizar para resolver um problema de projeto particular em seu código.

O que é um padrão de projeto?

Benefícios dos padrões

Padrões são como um conjunto de ferramentas para soluções de problemas comuns em design de software. Eles definem uma linguagem comum que ajuda sua equipe a se comunicar mais eficientemente.

Mais sobre os benefícios »

Classificação

Padrões de projeto diferem por sua complexidade, nível de detalhe e grau de aplicabilidade. Além disso, eles podem ser categorizados por seu propósito e divididos em três grupos.

Mais sobre as categorias »

Catálogo de padrões

Lista de 22 padrões de projeto clássicos, agrupados por seu propósito.

Dê uma olhada no catálogo »

<https://refactoring.guru/pt-br/design-patterns>

Padrões de Projeto

- Os **padrões criacionais** fornecem mecanismos de criação de objetos que aumentam a flexibilidade e a reutilização de código.
- Os **padrões estruturais** explicam como montar objetos e classes em estruturas maiores, enquanto ainda mantém as estruturas flexíveis e eficientes.
- Os **padrões comportamentais** cuidam da comunicação eficiente e da assinalação de responsabilidades entre objetos.

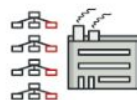
Retirado de (<https://refactoring.guru/pt-br/design-patterns/classification>), em 13/09/2022

Padrões de Projeto

Padrões criacionais



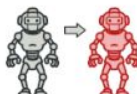
Factory Method



Abstract Factory



Builder



Prototype



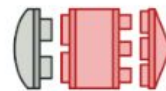
Singleton

Retirado de
(<https://refactoring.guru/pt-br/design-patterns/catalog>), em 13/09/2022

Padrões de Projeto

Padrões estruturais

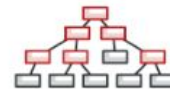
Retirado de
(<https://refactoring.guru/pt-br/design-patterns/catalog>), em 13/09/2022



Adapter



Bridge



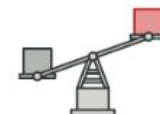
Composite



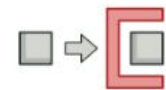
Decorator



Facade



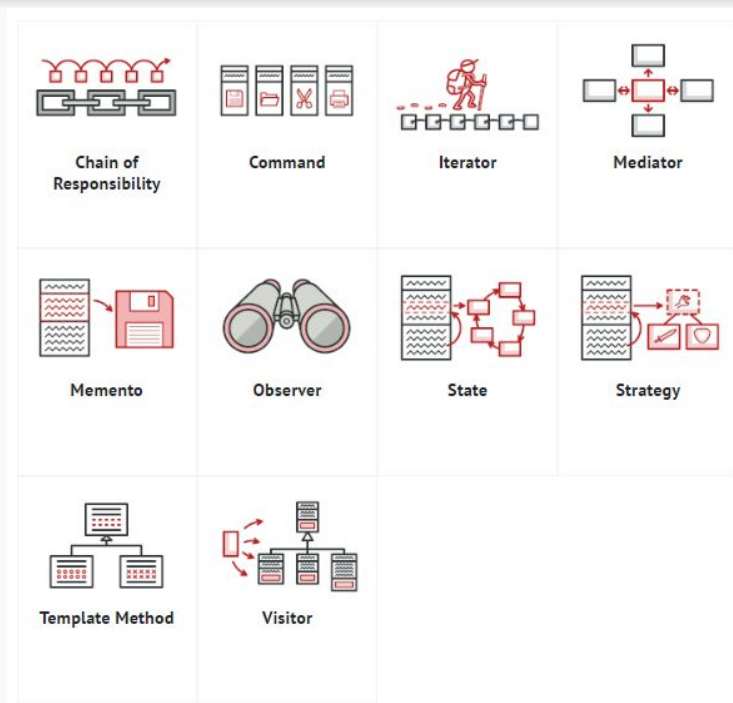
Flyweight



Proxy

Padrões de Projeto

Padrões comportamentais



Retirado de
(<https://refactoring.guru/pt-br/design-patterns/catalog>), em 13/09/2022

CUIDADO: Os padrões são GUIAS e não regras definitivas!

Padrão MVC

Padrão MVC

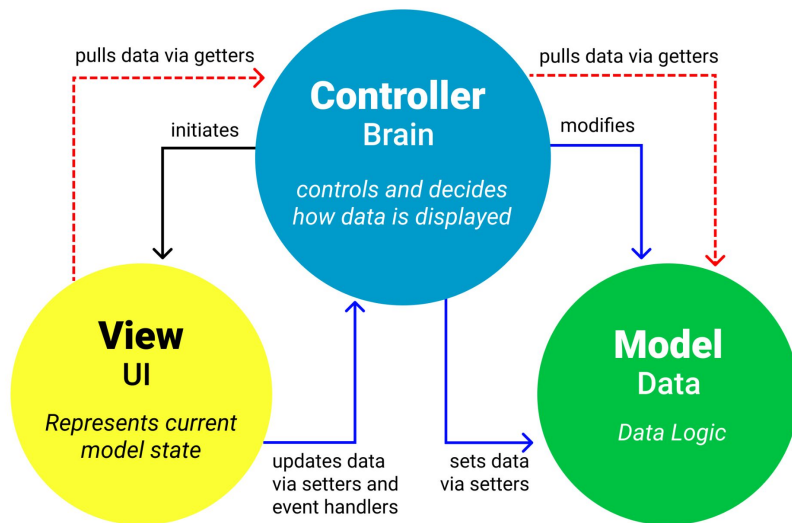
“MVC (Model-View-Controller) is a pattern in software design commonly used to implement user interfaces, data, and controlling logic. It emphasizes a separation between the software's business logic and display. This "separation of concerns" provides for a better division of labor and improved maintenance. Some other design patterns are based on MVC, such as MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter), and MVW (Model-View-Whatever). The three parts of the MVC software-design pattern can be described as follows:

- **Model:** Manages data and business logic.
- **View:** Handles layout and display.
- **Controller:** Routes commands to the model and view parts.”

Retirado de (<https://developer.mozilla.org/en-US/docs/Glossary/MVC>), em 13/09/2022

Padrão MVC

MVC Architecture Pattern



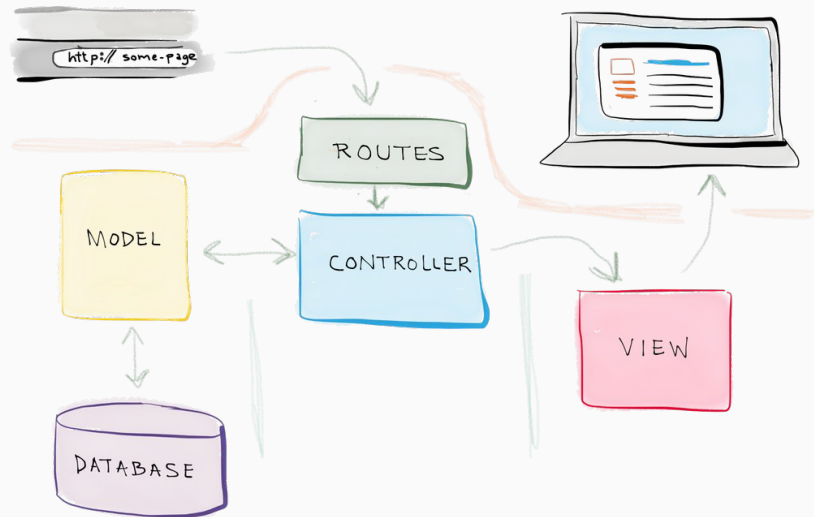
Retirado de
(<https://www.freecodecamp.org/news/content/images/2021/04/MVC3.png>), em 13/09/2022

Padrão MVC para Nossa Aplicação

Padrão MVC para Nossa Aplicação

- Para nosso projeto, vamos realizar a separação dos arquivos do nosso projeto.
- <https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos/>

Retirado de
(https://robocrop.realpython.net/?url=https%3A//files.realpython.com/media/mvc_diagram_with_routes.e12c5b982ac8.png&w=999&sig=92a7e18109cb47bc2a4e5119b522615e0a84b2d5), em
13/09/2022



Perguntas?



Retirado de
(<https://cdn-icons-png.flaticon.com/512/1268/1268705.png>), em 02/03/2022