

# IFRN

## INFOWEB – POO EM PYTHON

---

### Streamlit – Controle de Sessão

Prof. Gilbert Azevedo

# Conteúdo

- Controle de Sessão
  - Variável *session\_state* do Streamlit
  - Usando o controle de sessão
  - Fazendo a autenticação no sistema
- Sistema de Agendamento
  - Atualização da classe Cliente
  - Novos usuários: visitante e cliente
  - Validação da senha
  - Páginas de Login, Abrir Conta e Perfil
  - Novos menus de operações

# Controle de Sessão

- O controle de sessão em uma aplicação *web* permite compartilhar variáveis entre as execuções da aplicação
- Aplicações *desktop* mantêm as variáveis na memória até que sejam encerradas
- Aplicações *web* são re-executadas quando uma página é carregada, exigindo o controle explícito dos dados a serem mantidos
- Por exemplo, para controlar o *login* em uma aplicação, é necessário manter as informações do usuário em uma sessão

# Inserindo valores na sessão

- O *streamlit* mantém os dados da sessão na variável *session\_state* usando uma sintaxe similar a um dicionário ou a um atributo
- Teste e atribuição usando sintaxe de dicionário:
  - `if 'key' not in st.session_state:`
  - `st.session_state['key'] = 'value'`
- Teste e atribuição usando sintaxe de atributo:
  - `if 'key' not in st.session_state:`
  - `st.session_state.key = 'value'`

# Recuperando e atualizando valores na sessão

- Para recuperar um valor da sessão, é só acessar a chave do item:
  - `st.write(st.session_state['key'])`
  - `st.write(st.session_state.key)`
- Para atualizar o valor, basta atribuir um novo valor a uma chave existente:
  - `st.session_state['key'] = 'new_value'`
  - `st.session_state.key = 'new_value'`
- Também é possível recuperar todos os dados contidos na sessão:
  - `st.write(st.session_state)`

# Exemplo

- No código abaixo, dois valores são inseridos na sessão e apresentados na página
  - `if 'key1' not in st.session_state:`
  - `st.session_state['key1'] = 'value1'`
  - `if 'key2' not in st.session_state:`
  - `st.session_state.key2 = 'value2'`
  - `st.write(st.session_state['key1'])`
  - `st.write(st.session_state.key2)`
  - `st.write(st.session_state)`

value1

value2

```
{  
  "key2" : "value2"  
  "key1" : "value1"  
}
```

# Removendo valores na sessão

- Para remover um valor, basta utilizar a instrução *del* do Python:
  - `del st.session_state['key']`
- Removendo todos os valores da sessão:
  - `for key in st.session_state.keys():`
  - `del st.session_state[key]`

# Autenticação no Sistema

- O controle de sessão será usado na autenticação do usuário, inserindo *id* e *nome* do usuário na *session\_state*
- Se algumas das chaves, *id* ou *nome* do usuário, não estiver na sessão, significa que nenhum usuário está logado: visitante
- Quando um usuário entrar no sistema (*login*), após sua senha ser validada, as chaves *usuario\_id* e *usuario\_nome* serão inseridas na sessão
- Quando um usuário sair do sistema (*logout*), essas chaves serão removidas



# Passo 1. Atualização da Classe Cliente

- A autenticação do sistema de agendamento vai ser realizada, inicialmente, com o e-mail e a senha do cliente
- Para isso, o atributo *senha* deve ser incluído na classe *Cliente*

```
class Cliente:
    def __init__(self, id, nome, email, fone, senha):
        self.set_id(id)
        self.set_nome(nome)
        self.set_email(email)
        self.set_fone(fone)
        self.set_senha(senha)
```

## Passo 1.1. Atualização da Classe Cliente

- Insira os métodos *get* e *set* para o novo atributo

```
def get_senha(self): return self.__senha
```

```
def set_senha(self, senha): self.__senha = senha
```

- Altere os métodos *to\_json* e *from\_json* para suportar o novo atributo

```
def to_json(self):
```

```
    dic = {"id":self.__id, "nome":self.__nome, "email":self.__email,  
          "fone":self.__fone, "senha":self.__senha}
```

```
    return dic
```

```
@staticmethod
```

```
def from_json(dic):
```

```
    return Cliente(dic["id"], dic["nome"], dic["email"], dic["fone"],  
                  dic["senha"])
```

## Passo 1.2. Atualização da View

- Atualize as operações da *View* relacionadas ao cadastro de clientes
  - Listar e listar\_id não são alterados

```
def cliente_inserir(nome, email, fone, senha):
```

```
    cliente = Cliente(0, nome, email, fone, senha)
```

```
    ClienteDAO.inserir(cliente)
```

```
def cliente_atualizar(id, nome, email, fone, senha):
```

```
    cliente = Cliente(id, nome, email, fone, senha)
```

```
    ClienteDAO.atualizar(cliente)
```

```
def cliente_excluir(id):
```

```
    cliente = Cliente(id, "", "", "", "")
```

```
    ClienteDAO.excluir(cliente)
```

## Passo 1.3. Atualização da UI de clientes

- Modifique o método *inserir* de *ManterClienteUI* para solicitar a senha do usuário

```
def inserir():  
    nome = st.text_input("Informe o nome")  
    email = st.text_input("Informe o e-mail")  
    fone = st.text_input("Informe o fone")  
    senha = st.text_input("Informe a senha", type="password")  
    if st.button("Inserir"):  
        View.cliente_inserir(nome, email, fone, senha)  
        st.success("Cliente inserido com sucesso")  
        time.sleep(2)  
        st.rerun()
```

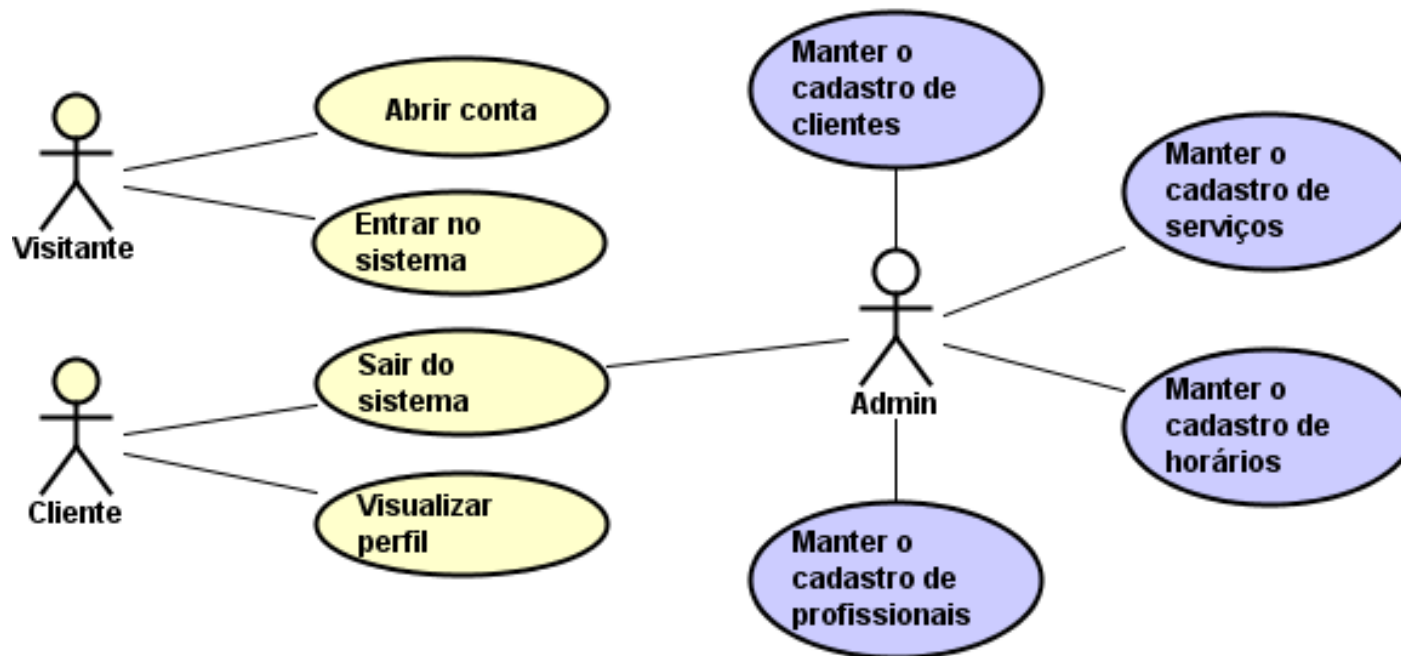
## Passo 1.4. Atualização da UI de clientes

- Modifique o método *atualizar* para permitir a alteração da senha

```
def atualizar():
    clientes = View.cliente_listar()
    if len(clientes) == 0: st.write("Nenhum cliente cadastrado")
    else:
        op = st.selectbox("Atualização de Clientes", clientes)
        nome = st.text_input("Novo nome", op.get_nome())
        email = st.text_input("Novo e-mail", op.get_email())
        fone = st.text_input("Novo fone", op.get_fone())
        senha = st.text_input("Nova senha", op.get_senha(),
                               type="password")
        if st.button("Atualizar"):
            id = op.get_id()
            View.cliente_atualizar(id, nome, email, fone, senha)
            st.success("Cliente atualizado com sucesso")
```

## Passo 2. Novos Usuários

- O sistema de agendamento terá novos atores e funcionalidades
  - Visitante: é o usuário que abre conta no sistema (apenas como Cliente) e que entra no sistema, autenticando-se como Admin ou como Cliente
  - Cliente: é o usuário que, por enquanto, pode apenas visualizar seu perfil



## Passo 2.1. Usuário Admin

- Para facilitar, o usuário *Admin* vai ser um cliente com e-mail = Admin e senha = 1234
- A operação *cliente\_criar\_admin* da classe *View* será responsável por cadastrar o usuário *Admin* no sistema
- Essa operação será chamada na página *index.py*

```
def cliente_criar_admin():  
    for c in View.cliente_listar():  
        if c.get_email() == "admin": return  
    View.cliente_inserir("admin", "admin", "fone", "1234")
```

## Passo 3. Validação da Senha

- A autenticação no sistema, validando a senha do usuário, será realizada pelo método *cliente\_autenticar* da classe *View*
- Essa operação será chamada na página de *login*

```
def cliente_autenticar(email, senha):  
    for c in View.cliente_listar():  
        if c.get_email() == email and c.get_senha() == senha:  
            return {"id": c.get_id(), "nome": c.get_nome()}  
    return None
```




## Passo 4. Página de Login

- A Página de *Login* realiza a autenticação do usuário, inserindo (ou não) os dados do cliente na sessão
- Insira o arquivo *loginUI.py* na pasta *templates*

### Entrar no Sistema

Informe o e-mail

Informe a senha



Entrar

## Passo 4.1. Código da Página de Login

```
import streamlit as st
from views import View

class LoginUI:
    def main():
        st.header("Entrar no Sistema")
        email = st.text_input("Informe o e-mail")
        senha = st.text_input("Informe a senha", type="password")
        if st.button("Entrar"):
            c = View.cliente_autenticar(email, senha)
            if c == None: st.write("E-mail ou senha inválidos")
            else:
                st.session_state["usuario_id"] = c["id"]
                st.session_state["usuario_nome"] = c["nome"]
                st.rerun()
```

## Passo 5. Página de Abrir Conta no Sistema

- A Página de *Abrir Conta no Sistema* permite ao cliente se cadastrar
- Insira o arquivo *abrircontaUI.py* na pasta *templates*

### Abrir Conta no Sistema

Informe o nome

Informe o e-mail

Informe o fone

Informe a senha



Inserir

## Passo 5.1. Código da página de Abrir Conta

```
import streamlit as st
from views import View
import time

class AbrirContaUI:
    def main():
        st.header("Abrir Conta no Sistema")
        nome = st.text_input("Informe o nome")
        email = st.text_input("Informe o e-mail")
        fone = st.text_input("Informe o fone")
        senha = st.text_input("Informe a senha", type="password")
        if st.button("Inserir"):
            View.cliente_inserir(nome, email, fone, senha)
            st.success("Conta criada com sucesso")
            time.sleep(2)
            st.rerun()
```

## Passo 6. Página de Perfil do Cliente

- A Página de *Perfil do Cliente* permite ao cliente alterar seus dados
- Insira o arquivo *perfilclienteUI.py* na pasta *templates*

### Meus Dados

Informe o novo nome

Informe o novo e-mail

Informe o novo fone

Informe a nova senha

Atualizar

## Passo 6.1. Código da Página de Perfil

```
import streamlit as st
from views import View
import time
class PerfilClienteUI:
    def main():
        st.header("Meus Dados")
        op = View.cliente_listar_id(st.session_state["usuario_id"])
        nome = st.text_input("Informe o novo nome", op.get_nome())
        email = st.text_input("Informe o novo e-mail", op.get_email())
        fone = st.text_input("Informe o novo fone", op.get_fone())
        senha = st.text_input("Informe a nova senha", op.get_senha(),
                               type="password")
        if st.button("Atualizar"):
            id = op.get_id()
            View.cliente_atualizar(id, nome, email, fone, senha)
            st.success("Cliente atualizado com sucesso")
```

## Passo 7. Atualização da Página *index*

- A Página *index* passa a controlar o menu dos usuários *visitante*, *cliente* e *Admin*
- Importe os novos módulos na página

```
from templates.abrircontaUI import AbrirContaUI
```

```
from templates.loginUI import LoginUI
```

```
from templates.perfilclienteUI import PerfilClienteUI
```

```
from views import View
```

## Passo 7.1. Menus de visitante e cliente

- Insira os menus de visitante e cliente

```
def menu_visitante():  
    op = st.sidebar.selectbox("Menu", ["Entrar no Sistema",  
                                        "Abrir Conta"])  
    if op == "Entrar no Sistema": LoginUI.main()  
    if op == "Abrir Conta": AbrirContaUI.main()  
  
def menu_cliente():  
    op = st.sidebar.selectbox("Menu", ["Meus Dados"])  
    if op == "Meus Dados": PerfilClienteUI.main()
```



## Passo 7.2. Menu do Admin

- Atualize o menu do Admin, se necessário

```
def menu_admin():  
    op = st.sidebar.selectbox("Menu", ["Cadastro de Clientes",  
        "Cadastro de Serviços", "Cadastro de Horários",  
        "Cadastro de Profissionais"])  
    if op == "Cadastro de Clientes": ManterClienteUI.main()  
    if op == "Cadastro de Serviços": ManterServicoUI.main()  
    if op == "Cadastro de Horários": ManterHorarioUI.main()  
    if op == "Cadastro de Profissionais": ManterProfissionalUI.main()
```

## Passo 7.2. Operação de Sair do Sistema

- Insira a operação de sair do sistema

```
def sair_do_sistema():  
    if st.sidebar.button("Sair"):  
        del st.session_state["usuario_id"]  
        del st.session_state["usuario_nome"]  
        st.rerun()
```

## Passo 7.2. Operação do Menu Lateral

- Atualize a operação do menu lateral

```
def sidebar():  
    if "usuario_id" not in st.session_state:  
        IndexUI.menu_visitante()  
    else:  
        admin = st.session_state["usuario_nome"] == "admin"  
        st.sidebar.write("Bem-vindo(a), " +  
            st.session_state["usuario_nome"])  
        if admin: IndexUI.menu_admin()  
        else: IndexUI.menu_cliente()  
        IndexUI.sair_do_sistema()
```

## Passo 7.3. Operação Main de Index

- Atualize a operação *main* da página *index* e a aplicação estará concluída com as novas funcionalidades

```
def main():  
    # verifica a existe o usuário admin  
    View.cliente_criar_admin()  
    # monta o sidebar  
    IndexUI.sidebar()
```

# Referências

- Documentação do Streamlit – Session State
  - [https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session\\_state](https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session_state)