IFRN

INFOWEB – POO EM PYTHON

Associações entre classes

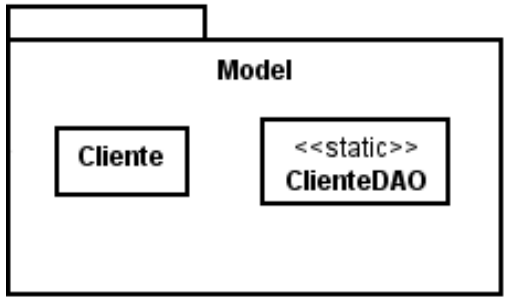
Prof. Gilbert Azevedo

Conteúdo

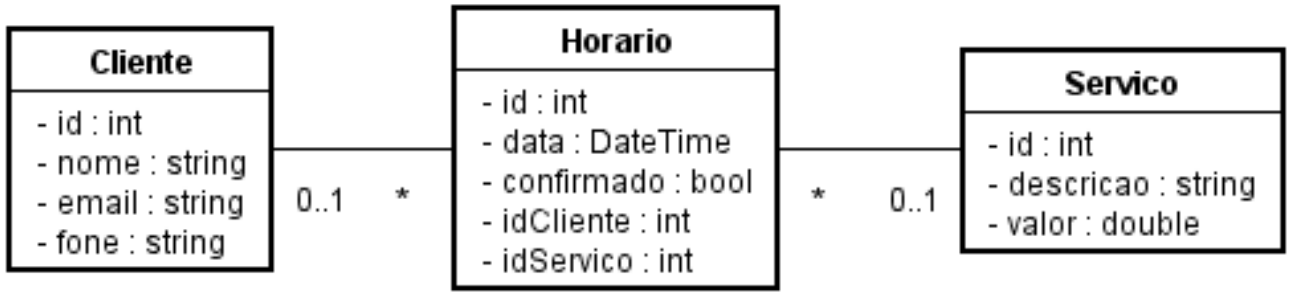
- Classes e Entidades
  - Tipos de relacionamentos
  - Associações entre classes
  - Multiplicidade
- Sistema de Agendamento
  - Passo a passo do cadastro de horários

Entidades e Relacionamentos

- As classes que representam as entidades em um sistema, na camada Model, normalmente estão relacionadas entre si



- No Sistema de Agendamento, por exemplo: Clientes, Horários e Serviços estão relacionados



Tipos de Relacionamentos

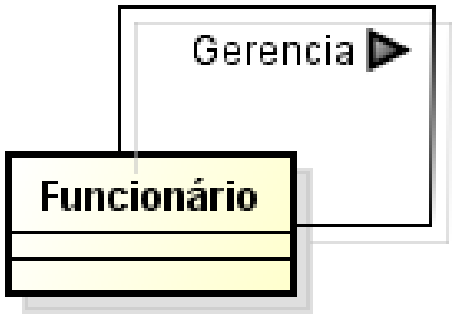
- Existem diversos tipos de relacionamentos na POO que definem como as classes de um sistema se relacionam:
  - Associação Unária
  - Associação Binária
  - Agregação
  - Composição
  - Especialização e Generalização
  - Dependência
  - Realização

Associação

- O relacionamento de associação ocorre quando uma instância de uma classe está ligada a um ou mais instâncias de outra classe ou de sua própria classes
- Por exemplo, um fornecedor (objeto da classe Fornecedor) fornece um ou mais produtos (objetos da classe Produto)
- Associações são representadas por retas ligando as classes envolvidas

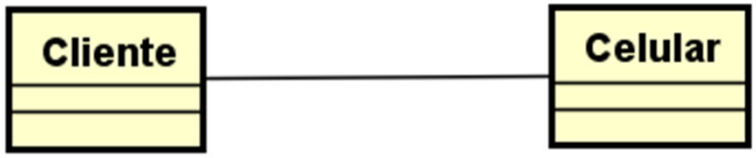
Associação Unária

- Ocorre quando instâncias de uma classe se associa com outras instância da própria classe
- Ex: O gerente, que é um funcionário, gerencia funcionários em uma empresa



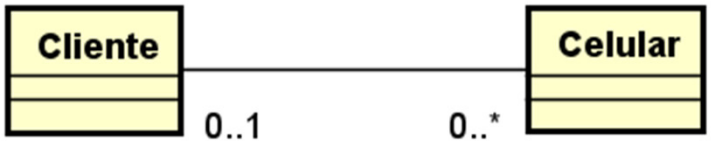
Associação Binária

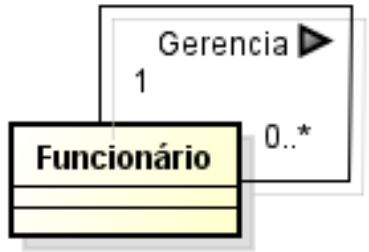
- Ocorre quando instâncias de uma classe se associa com instâncias da outra classe
- Ex: Um cliente pode possuir um ou mais celulares

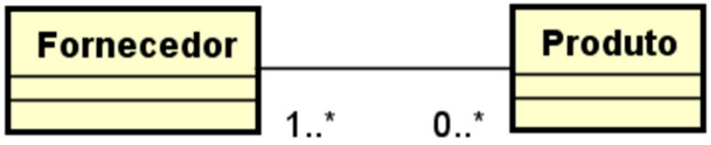


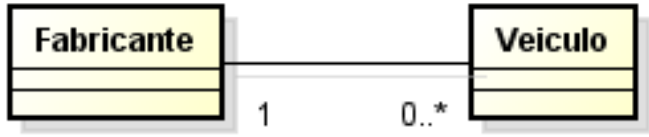
Multiplicidade

- A multiplicidade informa o número de instâncias envolvidas em cada uma das classes do relacionamento
  - 0..1: Mínimo zero e máximo um objeto envolvido no relacionamento
  - 0..*: Mínimo zero e máximo muitos objetos
  - 1..1: Um e somente um objeto
  - 1..*: Mínimo um e máximo muitos objetos
  - 1: Um objeto (equivalente a 1..1)
  - *: Muitos objetos (equivalente a 0..*)



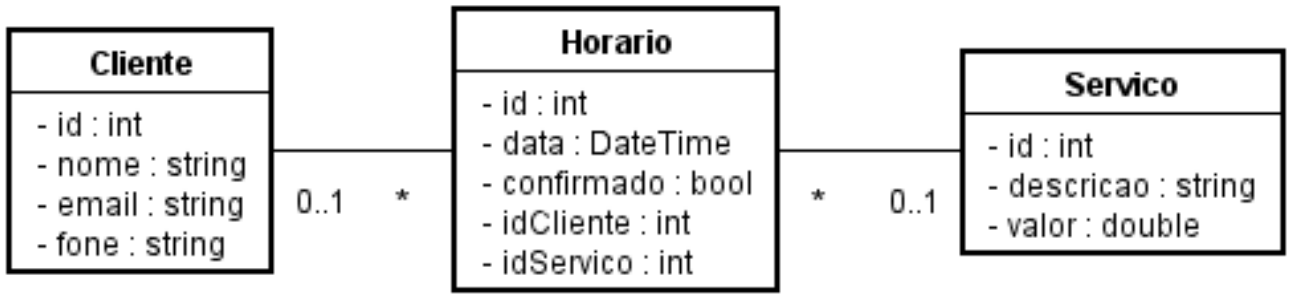






Modelo do Sistema de Agendamento

- As entidades Cliente, Serviço e Horário representam as entidades do sistema e estão relacionadas entre si
- Um cliente pode agendar vários horários de atendimento e, em cada horário, é realizado um serviço
- Um serviço pode ser agendado várias vezes em horários distintos sendo que, em cada horário, um cliente é atendido
- Um determinado horário pode estar disponível para ser agendado por um cliente e para ser realizado um serviço específico



Relacionamento com Ids

- A *priori*, *ids* não aparecem em um diagrama de classes. No exemplo, estes estão representados para facilitar o entendimento.
- A classe Horario poderia definir atributos cliente e serviço, ao invés de *idCliente* e *idServico*
- A utilização de *ids* facilita a serialização dos objetos em *json* e evita o aninhamento de objetos

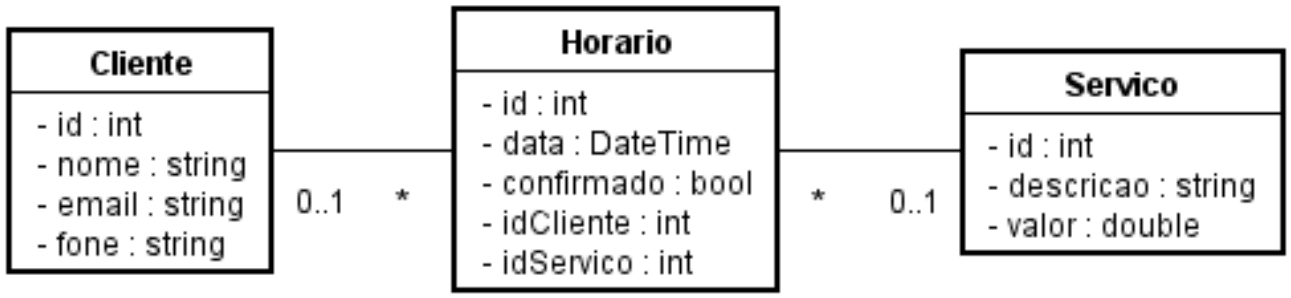


Diagrama de Classes do Sistema

- Atualização das camadas *Template* e *Model*
  - Camada *Template* com as páginas de cadastros além de *IndexUI*
  - Camada *Model* com as entidades e os DAOs

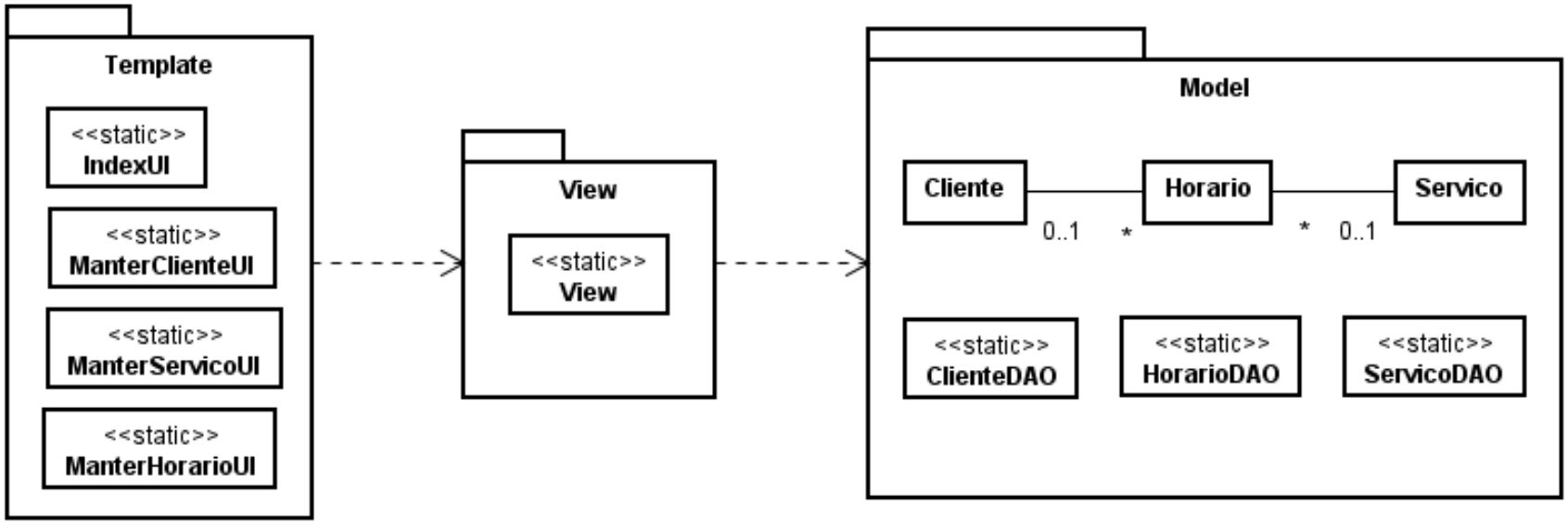
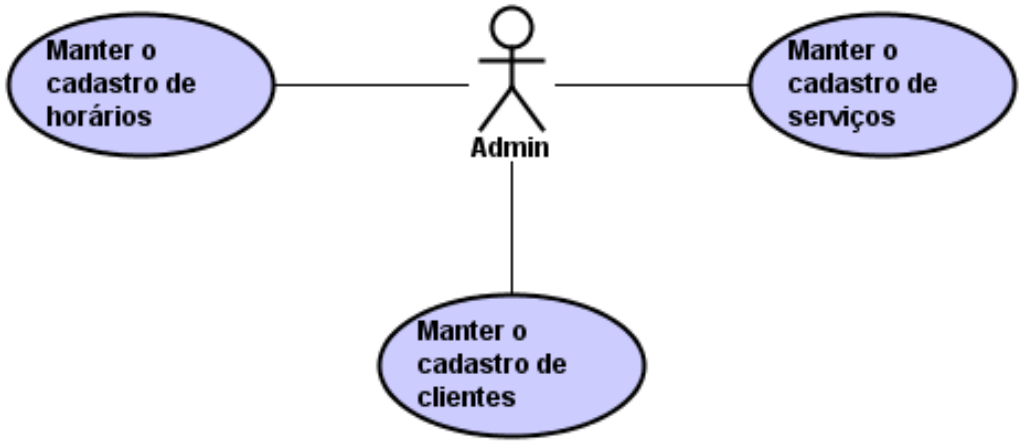


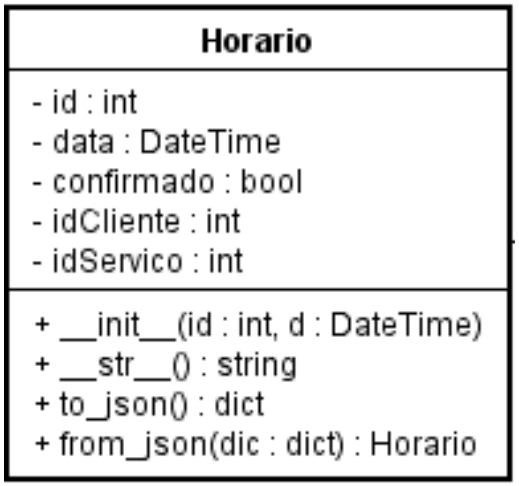
Diagrama de Casos de Uso

- Atualização do diagrama de casos de uso para mostrar as três operações de cadastro implementadas no sistema, todas realizadas pelo ator Admin



Passo 1. Definição da classe Horario

- Programe a classe Horario de acordo com o diagrama abaixo
- Insira os métodos *get* e *set* não listados no diagrama
- O método *to_json* deve retornar um dicionário com os dados de um horário
- O método *from_json* deve retornar um Horário com base no dicionário informado



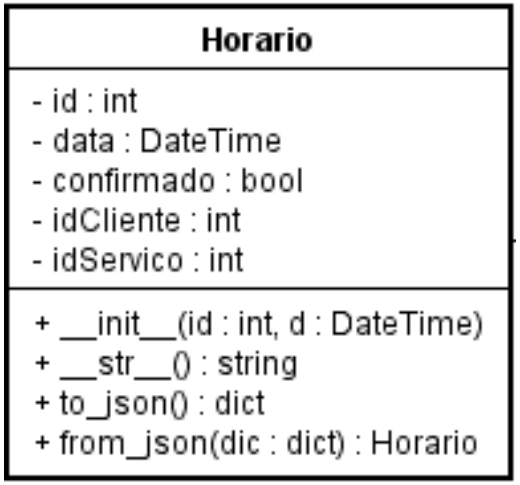
Passo 1.1. __init__ e __str__

- O *__init__* recebe apenas o *id* e a *data* do serviço

```
from datetime import datetime

class Horario:
    def __init__(self, id, data):
        self.set_id(id)
        self.set_data(data)
        self.set_confirmado(False)
        self.set_id_cliente(0)
        self.set_id_servico(0)

    def __str__(self):
        return f"{self.__id} - self.__data.strftime('%d/%m/%Y %H:%M')}
            - {self.__confirmado}"
```



Passo 2.2. *gets* e *sets*

- Métodos para definir e recuperar os atributos

```
def get_id(self): return self.__id
def get_data(self): return self.__data
def get_confirmado(self): return self.__confirmado
def get_id_cliente(self): return self.__id_cliente
def get_id_servico(self): return self.__id_servico

def set_id(self, id): self.__id = id
def set_data(self, data): self.__data = data
def set_confirmado(self, confirmado): self.__confirmado = confirmado
def set_id_cliente(self, id_cliente): self.__id_cliente = id_cliente
def set_id_servico(self, id_servico): self.__id_servico = id_servico
```

Passo 2.3. *O*perações com *json*

- Métodos para manipulação do arquivo *json*

```
def to_json(self):
    dic = {"id":self.__id, "data":self.__data.strftime("%d/%m/%Y %H:%M"),
        "confirmado":self.__confirmado, "id_cliente":self.__id_cliente,
        "id_servico":self.__id_servico}
    return dic

@staticmethod
def from_json(dic):
    horario = Horario(dic["id"], datetime.strptime(dic["data"], "%d/%m/%Y
        %H:%M"))
    horario.set_confirmado(dic["confirmado"])
    horario.set_id_cliente(dic["id_cliente"])
    horario.set_id_servico(dic["id_servico"])
    return horario
```

Passo 3. HorarioDAO

- Programe a classe HorarioDAO de acordo com o diagrama abaixo
- A classe mantém a lista de objetos horários do cadastro
- Inserir, Listar, Listar_Id, Atualizar e Excluir realizam as operações do CRUD
- Abrir e Salvar realizam as operações com o arquivo *json*

Passo 3.1. HorarioDAO.inserir

```
import json

class HorarioDAO:
    __objetos = []

    @classmethod
    def inserir(cls, obj):
        cls.abrir()
        id = 0
        for aux in cls.__objetos:
            if aux.get_id() > id: id = aux.get_id()
        obj.set_id(id + 1)
        cls.__objetos.append(obj)
        cls.salvar()
```

Passo 3.2. HorarioDAO.listar

```
@classmethod
def listar(cls):
    cls.abrir()
    return cls.__objetos

@classmethod
def listar_id(cls, id):
    cls.abrir()
    for obj in cls.__objetos:
        if obj.get_id() == id: return obj
    return None
```

Passo 3.3. atualizar e excluir

```
@classmethod
def atualizar(cls, obj):
    aux = cls.listar_id(obj.get_id())
    if aux != None:
        cls.__objetos.remove(aux)
        cls.__objetos.append(obj)
        cls.salvar()

@classmethod
def excluir(cls, obj):
    aux = cls.listar_id(obj.get_id())
    if aux != None:
        cls.__objetos.remove(aux)
        cls.salvar()
```

Passo 3.4. abrir e salvar

```
@classmethod
def abrir(cls):
    cls.__objetos = []
    try:
        with open("horarios.json", mode="r") as arquivo:
            list_dic = json.load(arquivo)
            for dic in list_dic:
                obj = Horario.from_json(dic)
                cls.__objetos.append(obj)
    except FileNotFoundError:
        pass

@classmethod
def salvar(cls):
    with open("horarios.json", mode="w") as arquivo:
        json.dump(cls.__objetos, arquivo, default = Horario.to_json)
```

Passo 4. View

- Programe a classe View para incluir as operações relacionadas ao cadastro de horários (Não remova os métodos de serviços e clientes já existentes)

```
from models.horario import Horario, HorarioDAO
class View:
    def horario_inserir(data, confirmado, id_cliente, id_servico):
        c = Horario(0, data)
        c.set_confirmado(confirmado)
        c.set_id_cliente(id_cliente)
        c.set_id_servico(id_servico)
        HorarioDAO.inserir(c)

    def horario_listar():
        return HorarioDAO.listar()
```

Passo 4.1. Continuação da classe View

- Observe que o construtor da classe Horario não recebe todos os atributos do objeto, por isso os *sets* são utilizados

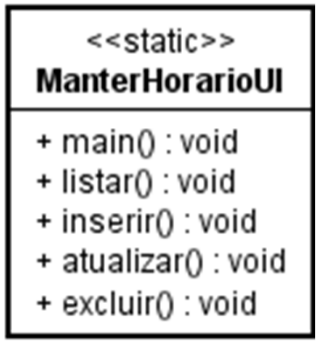
```
def horario_atualizar(id, data, confirmado, id_cliente, id_servico):
    c = Horario(id, data)
    c.set_confirmado(confirmado)
    c.set_id_cliente(id_cliente)
    c.set_id_servico(id_servico)
    HorarioDAO.atualizar(c)

def horario_excluir(id):
    c = Horario(id, None)
    HorarioDAO.excluir(c)
```

Passo 5. ManterHorarioUI

- A classe ManterHorarioUI vai ser usada para montar uma interface com o usuário para o cadastro de horários usando o Streamlit
- Cada uma das operações do CRUD: inserir, listar, atualizar e excluir, será realizada em um método diferente da classe, controlado pelo *main*
- Importe os módulos necessários à classe ManterHorarioUI

```
import streamlit as st
import pandas as pd
from views import View
import time
from datetime import datetime
```



Passo 5.1. main

- Main utiliza a função *tabs* do Streamlit para controlar abas na página

```
class ManterHorarioUI:
    def main():
        st.header("Cadastro de Horários")
        tab1, tab2, tab3, tab4 = st.tabs(["Listar", "Inserir",
            "Atualizar", "Excluir"])
        with tab1: ManterHorarioUI.listar()
        with tab2: ManterHorarioUI.inserir()
        with tab3: ManterHorarioUI.atualizar()
        with tab4: ManterHorarioUI.excluir()
```

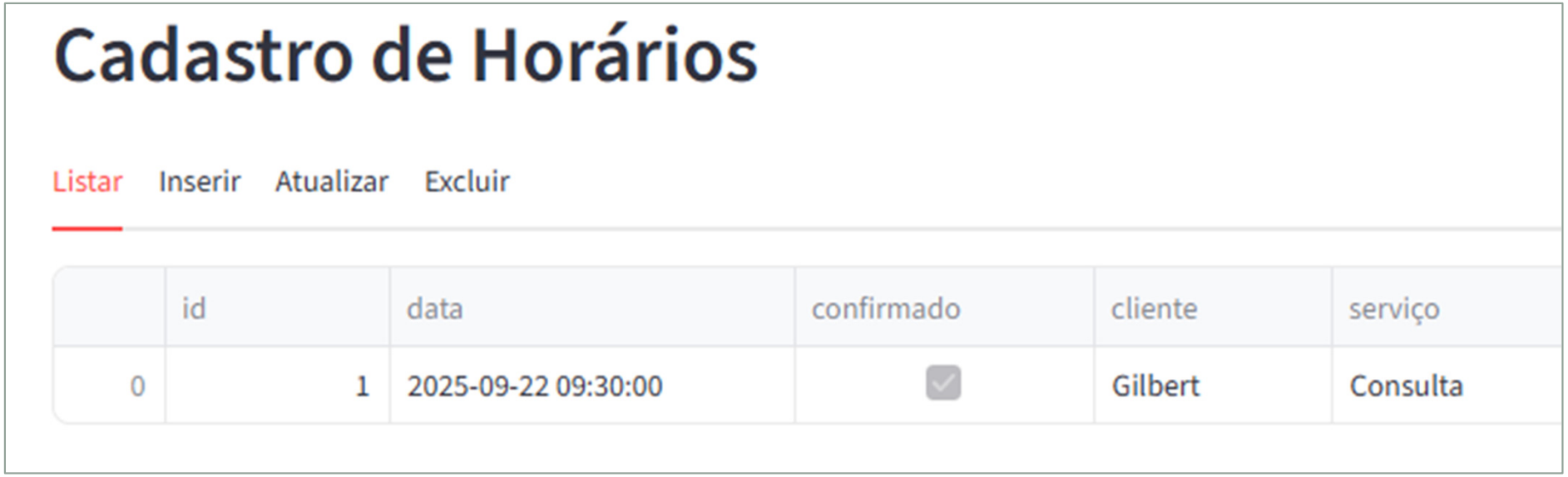
Passo 5.2. listar

- O método listar busca o nome do cliente e a descrição do serviço a partir dos *ids* de cliente e serviço para melhorar a experiência do usuário

```
def listar():
    horarios = View.horario_listar()
    if len(horarios) == 0: st.write("Nenhum horário cadastrado")
    else:
        dic = []
        for obj in horarios:
            cliente = View.cliente_listar_id(obj.get_id_cliente())
            servico = View.servico_listar_id(obj.get_id_servico())
            if cliente != None: cliente = cliente.get_nome()
            if servico != None: servico = servico.get_descricao()
            dic.append({"id" : obj.get_id(), "data" : obj.get_data(),
                "confirmado" : obj.get_confirmado(), "cliente" : cliente,
                "serviço" : servico})
        df = pd.DataFrame(dic)
        st.dataframe(df)
```

Passo 5.2. listar

- Observe que ao invés dos *ids*, a página mostra o nome do cliente e a descrição do serviço agendada



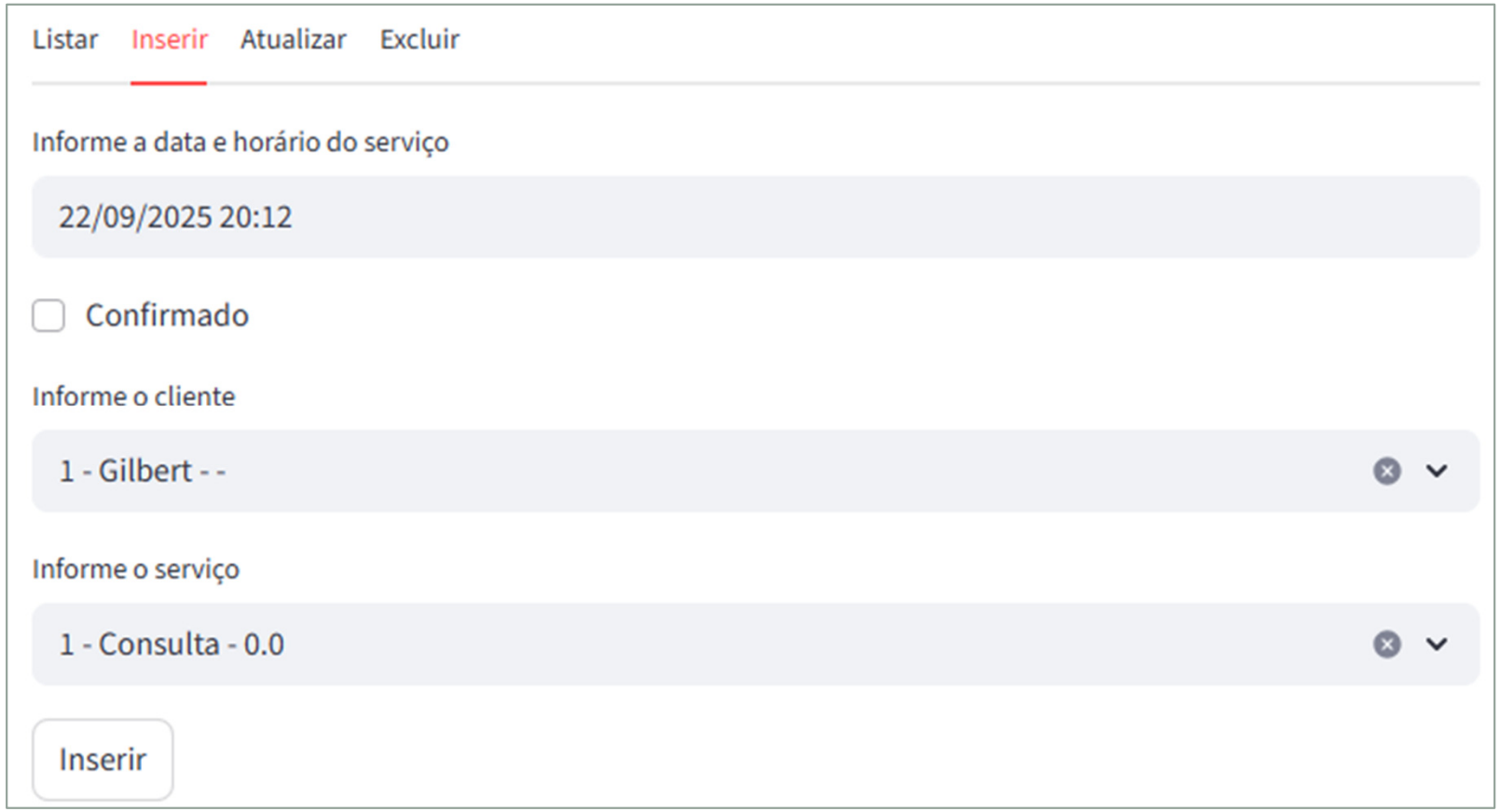
Passo 5.2. inserir

- Inserir usa *checkbox* para a confirmação do horário e *selectbox* para buscar os clientes e serviços. Insira o *sleep* e o *rerun* no final do código

```
def inserir():
    clientes = View.cliente_listar()
    servicos = View.servico_listar()
    data = st.text_input("Informe a data e horário do serviço",
        datetime.now().strftime("%d/%m/%Y %H:%M"))
    confirmado = st.checkbox("Confirmado")
    cliente = st.selectbox("Informe o cliente", clientes, index = None)
    servico = st.selectbox("Informe o serviço", servicos, index = None)
    if st.button("Inserir"):
        id_cliente = None
        id_servico = None
        if cliente != None: id_cliente = cliente.get_id()
        if servico != None: id_servico = servico.get_id()
        View.horario_inserir(datetime.strptime(data, "%d/%m/%Y %H:%M"),
            confirmado, id_cliente, id_servico)
        st.success("Horário inserido com sucesso")
```

Passo 5.2. inserir

- Observe o uso do *checkbox* e dos *selectboxes*



Passo 5.3. atualizar

- No atualizar, é preciso *linkar* os *selectboxes* com os valores atuais dos ids do cliente e do serviço

```
def atualizar():
    horarios = View.horario_listar()
    if len(horarios) == 0: st.write("Nenhum horário cadastrado")
    else:
        clientes = View.cliente_listar()
        servicos = View.servico_listar()
        op = st.selectbox("Atualização de Horários", horarios)
        data = st.text_input("Informe a nova data e horário do serviço",
            op.get_data().strftime("%d/%m/%Y %H:%M"))
        confirmado = st.checkbox("Nova confirmação", op.get_confirmado())
        id_cliente = None if op.get_id_cliente() in [0, None] else
            op.get_id_cliente()
        id_servico = None if op.get_id_servico() in [0, None] else
            op.get_id_servico()
```

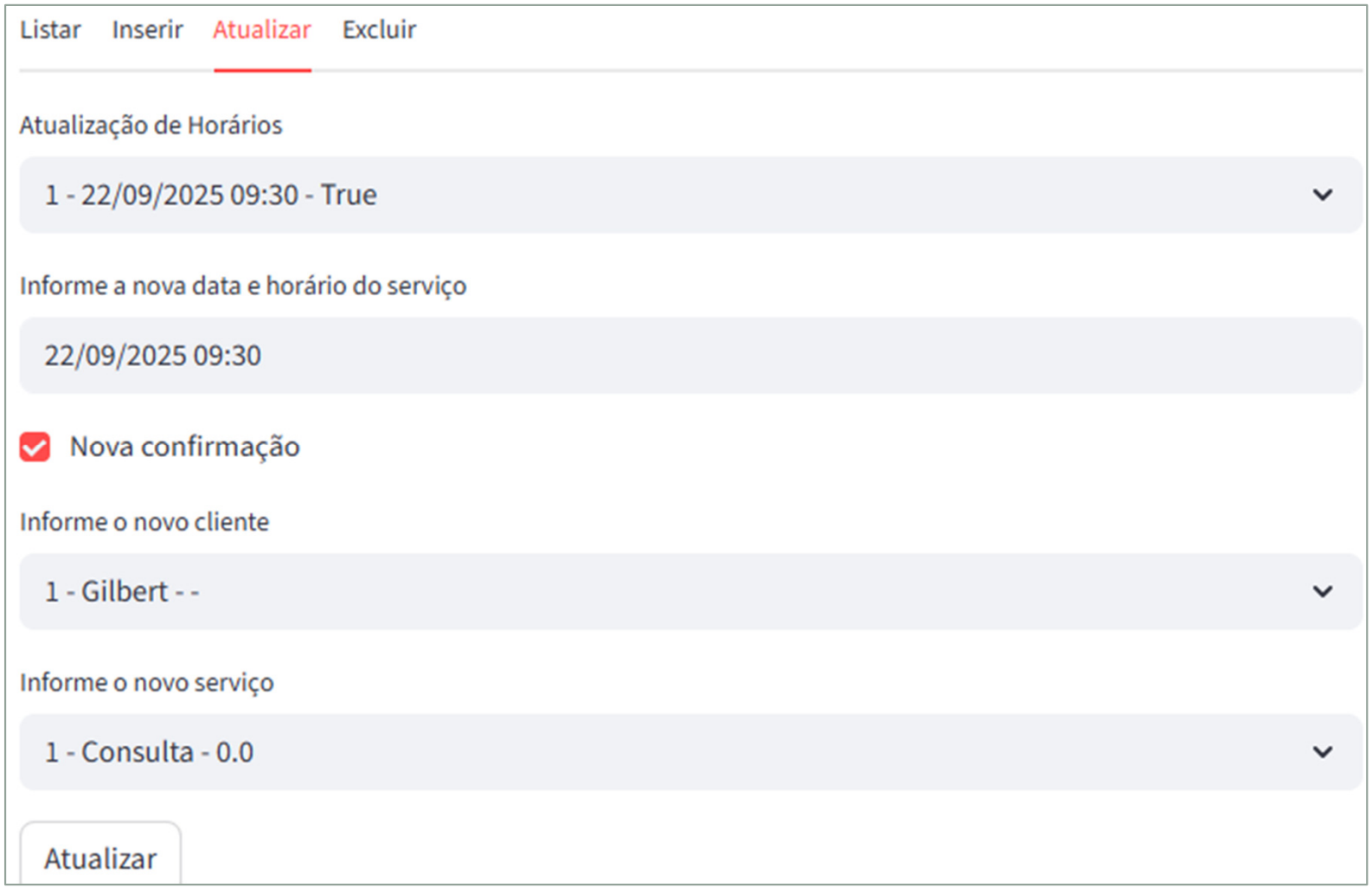
Passo 5.3. atualizar - continuação

- No atualizar, é preciso *linkar* os *selectboxes* com os valores atuais dos ids do cliente e do serviço

```
#id_serviço ...
cliente = st.selectbox("Informe o novo cliente", clientes, next((i for i,
    c in enumerate(clientes) if c.get_id() == id_cliente), None))
servico = st.selectbox("Informe o novo serviço", servicos, next((i for i,
    s in enumerate(servicos) if s.get_id() == id_servico), None))
if st.button("Atualizar"):
    id_cliente = None
    id_servico = None
    if cliente != None: id_cliente = cliente.get_id()
    if servico != None: id_servico = servico.get_id()
    View.horario_atualizar(op.get_id(), datetime.strptime(data, "%d/%m/%Y
        %H:%M"), confirmado, id_cliente, id_servico)
    st.success("Horário atualizado com sucesso")
```

Passo 5.3. atualizar

- Observe o uso do *checkbox* e dos *selectboxes*



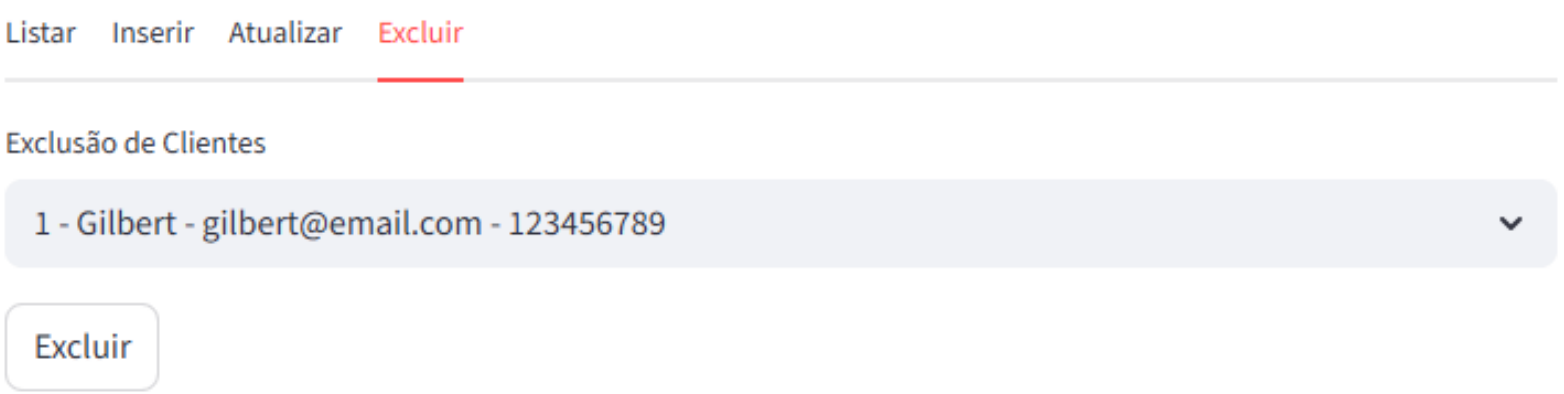
Passo 5.4. excluir

- Inserir usa *select_box* para excluir um horário

```
def excluir():
    horarios = View.horario_listar()
    if len(horarios) == 0: st.write("Nenhum horário cadastrado")
    else:
        op = st.selectbox("Exclusão de Horários", horarios)
        if st.button("Excluir"):
            View.horario_excluir(op.get_id())
            st.success("Horário excluído com sucesso")
            time.sleep(2)
            st.rerun()
```

Passo 5.4. ManterClienteUI.excluir

- Inserir usa *select_box* para excluir um cliente



Passo 6. IndexUI

- Altere a classe IndexUI para permitir a seleção das três páginas de cadastros através do menu lateral (*sidebar*)

- `from templates.manterclienteUI import ManterClienteUI`
- `from templates.manterservicoUI import ManterServicoUI`
- `from templates.manterhorarioUI import ManterHorarioUI`
- `import streamlit as st`

Passo 6. IndexUI - continuação

- Altere a classe IndexUI para permitir a seleção das três páginas de cadastro através do menu lateral (*sidebar*)

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

Passo 6. IndexUI - continuação

- Visualização do menu lateral

Referências

- Documentação do Streamlit
  - https://docs.streamlit.io/
- Código completo da aplicação
  - https://github.com/Gilbert-Silva/Agenda