IFRN

INFOWEB – POO EM PYTHON

Filtragem e Ordenação

Prof. Gilbert Azevedo

Conteúdo

- Filtragem e Ordenação de Listas
  - Filtragem de objetos
  - Ordenação de objetos
- Agendamento de serviço
  - Passo a passo da operação de agendamento

Filtragem de Objetos

- Em muitas aplicações, as operações de filtragem são realizadas para obter um objeto dentro de uma lista (cadastro)
- Por exemplo, a operação de *login* filtra o cadastro de usuários para encontrar o usuário que possue o e-mail e a senha informados

```
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
```

Filtragem de Objetos

- Algumas operações de filtragem retorna uma lista de objetos (ao invés de um único objeto) a partir de um cadastro da aplicação
- Por exemplo, encontrar os horários cadastrados para um profissional a partir de seu identificador (compare com *horario_listar*)

```
def horario_listar():
        return HorarioDAO.listar()
def horario_filtrar_profissional(id_profissional):
    r = []
    for h in View.horario_listar():
        if h.get_id_profissional() == id_profissional:
            r.append(h)
    return r
```

Ordenação de Objetos

- Quando vários objetos são inseridos em um cadastro, é importante definir um padrão de ordenação para melhorar a usabilidade da aplicação
- Por exemplo, para ordenar a listagem de clientes da aplicação, a operação de listagem deve ordenar a lista de clientes obtida do DAO
- O código abaixo usa o método *sort* e uma expressão *lambda* para ordenar os nomes dos clientes em ordem alfabética (independente dos *ids*)

```
def cliente_listar():
    r = ClienteDAO.listar()
    r.sort(key = lambda obj : obj.get_nome())
    return r
```

Ordenação de Objetos

- Ordenação do cadastro de clientes pelo nome

Passo 1. Ordenação dos Cadastros

- Atualize os métodos da classe *View* responsáveis pelas listagens de clientes, serviços, horários e profissionais para apresentar os dados em alguma ordem importante

```
def cliente_listar():
    r = ClienteDAO.listar()
    r.sort(key = lambda obj : obj.get_nome())
    return r

def servico_listar():
    r = ServicoDAO.listar()
    r.sort(key = lambda obj : obj.get_descricao())
    return r
```

Passo 1.1. Ordenação dos Cadastros

- Continuação...

```
def horario_listar():
    r = HorarioDAO.listar()
    r.sort(key = lambda obj : obj.get_data())
    return r

def profissional_listar():
    r = ProfissionalDAO.listar()
    r.sort(key = lambda obj : obj.get_nome())
    return r
```

Passo 2. Filtragem de Horários

- Insira um método na classe *View* para retornar os horários de um profissional específico que estão disponíveis para agendamento

```
def horario_agendar_horario(id_profissional):
    r = []
    agora = datetime.now()
    for h in View.horario_listar():
        if h.get_data() >= agora and h.get_confirmado() == False
            and h.get_id_cliente() == None
            and h.get_id_profissional() == id_profissional:
                r.append(h)
    r.sort(key = lambda h : h.get_data())
    return r
```

Passo 3. CDU Agendar Serviço

- Implemente o *CDU Agendar Serviço* para permitir ao cliente agendar um horário disponível com profissional específico para realizar um determinado serviço

Passo 3.1. CDU Agendar Serviço

- Insira a página agendarservicoUI na pasta templates

```
import streamlit as st
from views import View
import time
class AgendarServicoUI:
    def main():
        st.header("Agendar Serviço")
        profs = View.profissional_listar()
        if len(profs) == 0: st.write("Nenhum profissional cadastrado")
        else:
            profissional = st.selectbox("Informe o profissional", profs)
            horarios = View.horario_agendar_horario(profissional.get_id())
            if len(horarios) == 0: st.write("Nenhum horário disponível")
            else:
```

Passo 3.1. CDU Agendar Serviço

- Continuação...

```
# else:
    horario = st.selectbox("Informe o horário", horarios)
    servicos = View.servico_listar()
    servico = st.selectbox("Informe o serviço", servicos)
    if st.button("Agendar"):
        View.horario_atualizar(horario.get_id(),
            horario.get_data(), False,
            st.session_state["usuario_id"],
            servico.get_id(), profissional.get_id())
        st.success("Horário agendado com sucesso")
        time.sleep(2)
        st.rerun()
```

Passo 4. Atualização da Página *index*

- Atualize o menu do cliente na página *index*

```
from templates.agendarservicoUI import AgendarServicoUI

def menu_cliente():
    op = st.sidebar.selectbox("Menu", ["Meus Dados",
        "Agendar Serviço"])
    if op == "Meus Dados": PerfilClienteUI.main()
    if op == "Agendar Serviço": AgendarServicoUI.main()
```

Referências

- Documentação do Streamlit
  - https://docs.streamlit.io/