TRABALHO SEMESTRAL

PROF: RICARDO BLACK

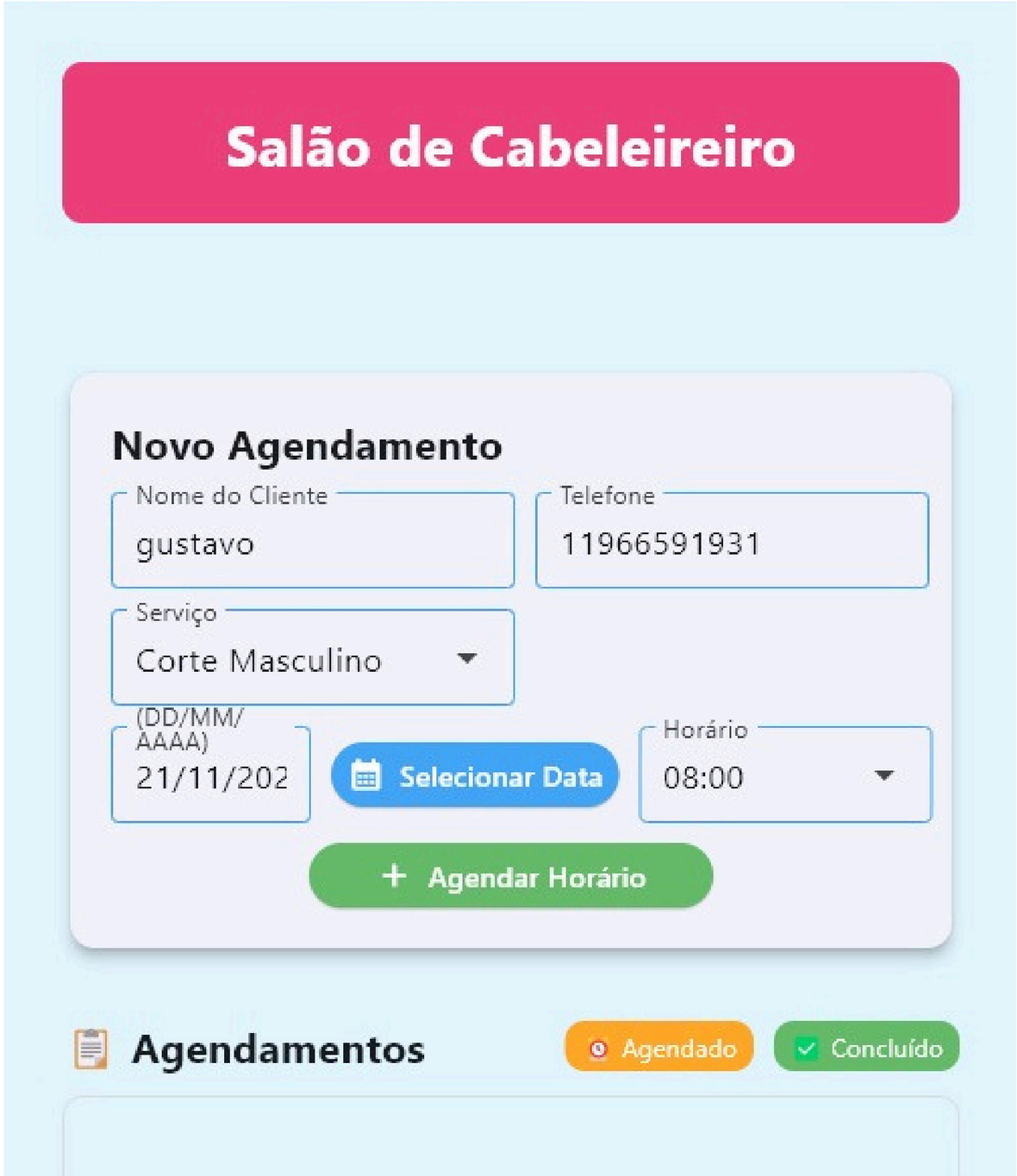
Integrantes do Grupo

- Gustavo de Sales
- Eduardo de Sales
- Ryan Honorato
- Stefany Coelho

Contexto do app

Nossa equipe, produziu um aplicativo voltado à planejamento e gerenciamento de clientes para um salão de beleza. O app é voltado para a profissional de beleza, sendo de uso próprio para a organização de clientes, horarios e custos.

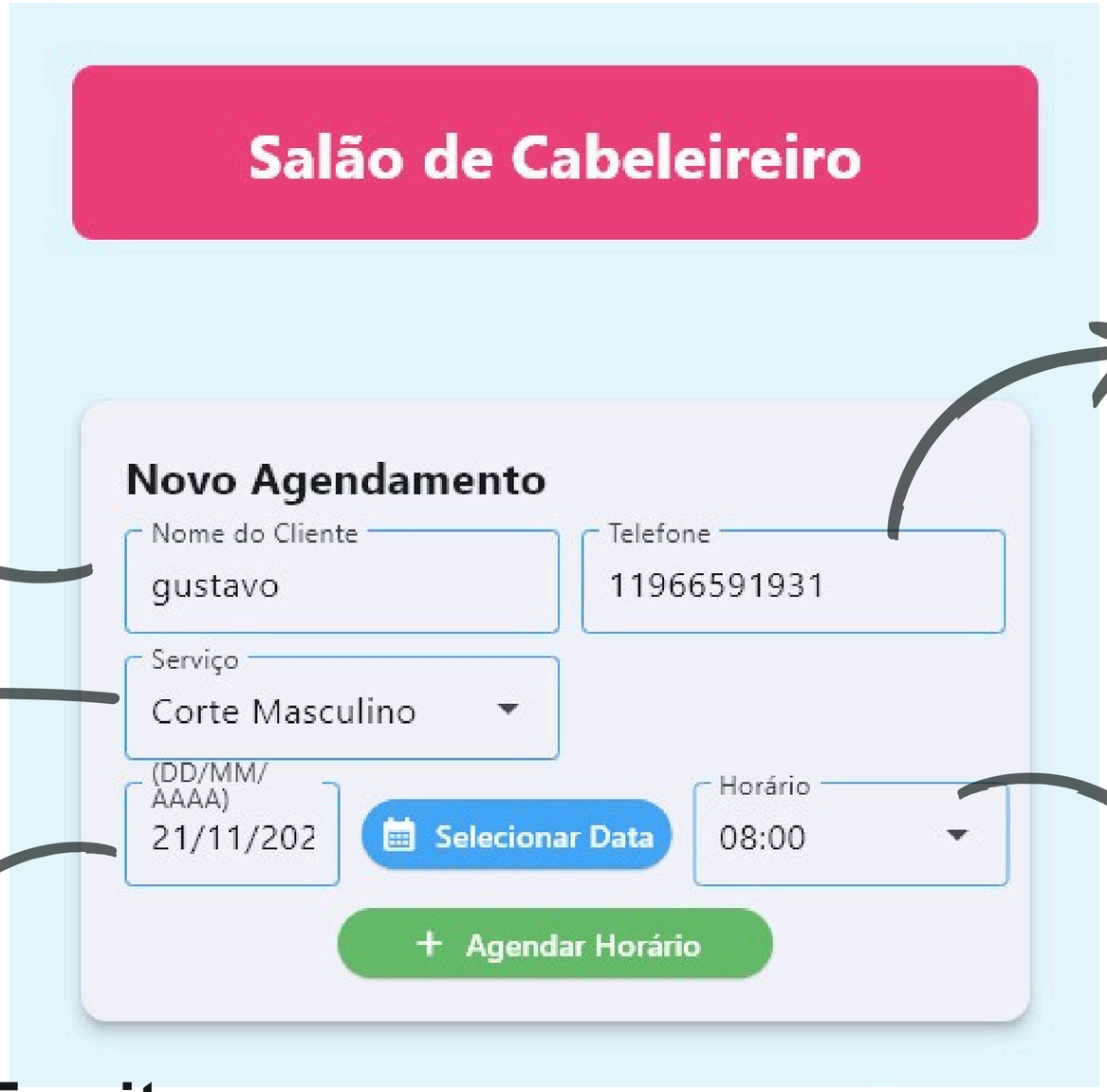
Interface


Salão de Cabeleireiro
Novo Agendamento
Nome do Cliente
gustavo
Telefone
11966591931
Serviço
Corte Masculino
(DD/MM/AAAA)
21/11/202
Selecionar Data
Horário
08:00
+ Agendar Horário
Agendamentos
Agendado
Concluído

Nome do Cliente

Número de Telefone do Cliente

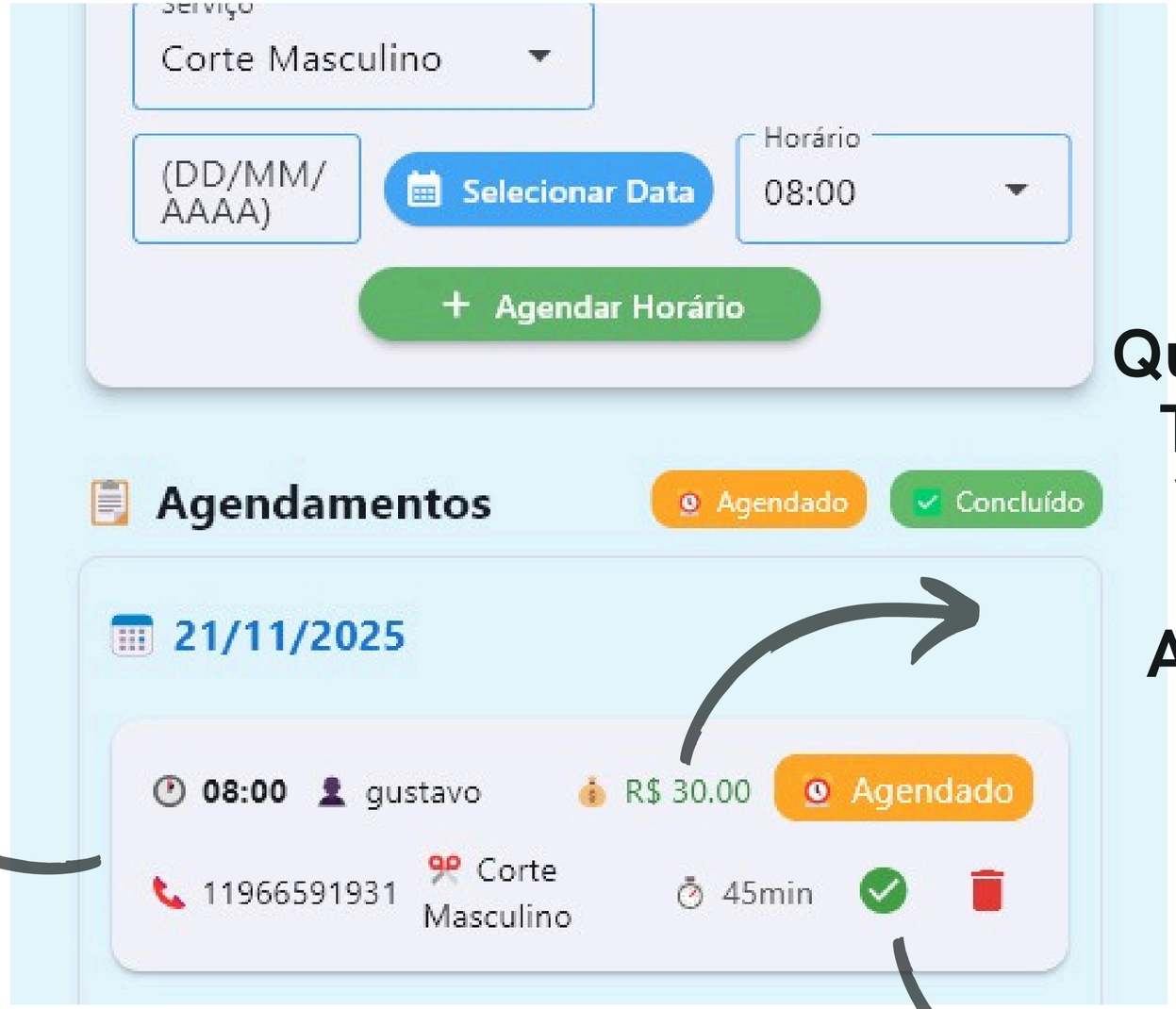
Selecionar Tipo de Serviço a Ser Feito



Escolha do Horário

A Data Pode ser Escrita ou Selecionada por um Calenáario

Horario Agendado com Todas as Informaçoes Necessarias

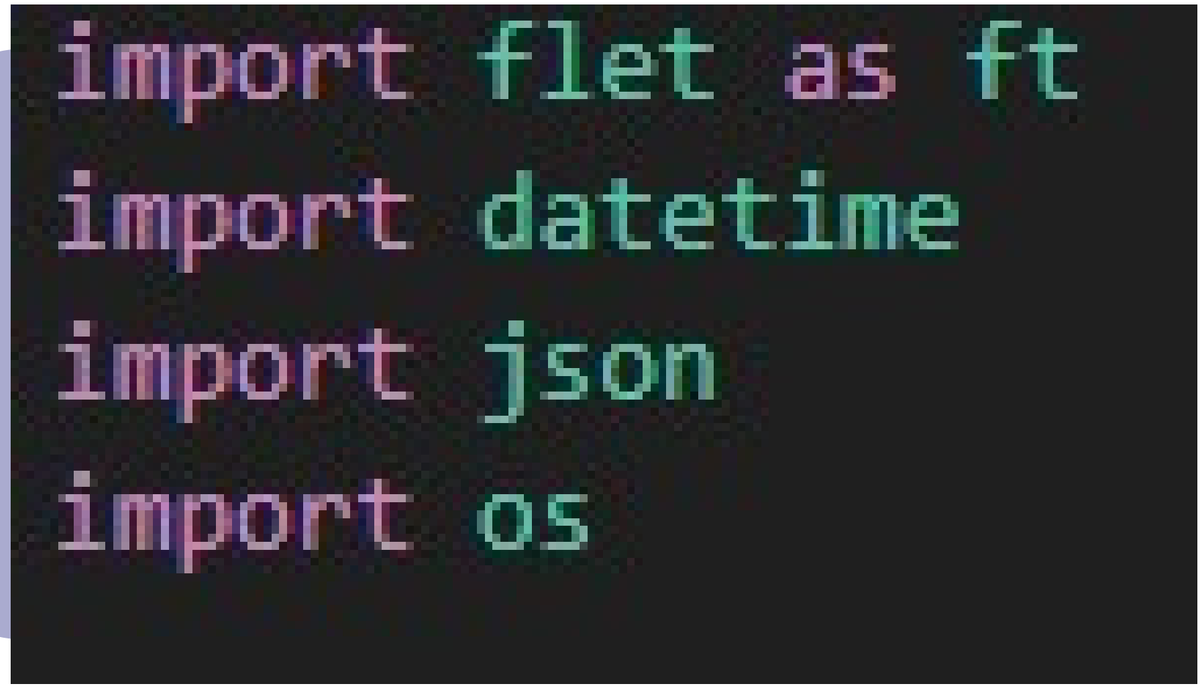


Quando Celeciona o Tipo de Serviço o Valor a ser Pago Aparece Automaticamente

Após Concluido ou Rejeitado a Informação Fica Guardada no Banco de Dados

Destaques de Codificaçao

Bibliotecas Utilizadas



```
lass SalaoAgendamento:
    def __init__(self):
        self.agendamentos = []
        self.arquivo_dados = "agendamentos.json"
        self.carregar_agendamentos()

        # Serviços disponíveis
        self.servicos = [
            {"nome": "Corte Feminino", "preco": 50.00, "duracao": 60},
            {"nome": "Corte Masculino", "preco": 30.00, "duracao": 45},
            {"nome": "Escova", "preco": 40.00, "duracao": 45},
            {"nome": "Coloração", "preco": 120.00, "duracao": 120},
            {"nome": "Hidratação", "preco": 60.00, "duracao": 60},
            {"nome": "Progressiva", "preco": 200.00, "duracao": 180},
            {"nome": "Manicure", "preco": 25.00, "duracao": 30},
            {"nome": "Pedicure", "preco": 30.00, "duracao": 45}
        ]

        # Horários disponíveis
        self.horarios_disponiveis = [
            "  ","08:00", "08:30", "09:00", "09:30", "10:00", "10:30",
            "11:00", "11:30", "12:00", "12:30", "13:00", "13:30",
            "14:00", "14:30", "15:00", "15:30", "16:00", "16:30",
            "17:00", "17:30", "18:00", "18:30"
        ]
```

- Inicia e salva o carregamento JSON
- Verifica se a um horario disponível

- Adiciona ou remove ajendamentos
- Marca como concluido

```
lass SalaoAgendamento:

    def adicionar_agendamento(self, nome, telefone, servico, data, horario):
        """Adiciona um novo agendamento"""
        servico_info = next((s for s in self.servicos if s['nome'] == servico), None)

        agendamento = {
            'id': len(self.agendamentos) + 1,
            'nome': nome,
            'telefone': telefone,
            'servico': servico,
            'preco': servico_info['preco'] if servico_info else 0,
            'duracao': servico_info['duracao'] if servico_info else 60,
            'data': data,
            'horario': horario,
            'status': 'Agendado'
        }

        self.agendamentos.append(agendamento)
        self.salvar_agendamentos()
        return True

    def remover_agendamento(self, agendamento_id):
        """Remove um agendamento"""
        for agendamento in self.agendamentos:
            if agendamento['id'] == agendamento_id:
                agendamento['status'] = 'Removido'
                self.salvar_agendamentos()
                return True
        return False

    def marcar_como_concluido(self, agendamento_id):
        """Marca um agendamento como concluído"""
        for agendamento in self.agendamentos:
            if agendamento['id'] == agendamento_id:
                agendamento['status'] = 'Concluído'
                self.salvar_agendamentos()
                return True
        return False
```

Desing

```python
def main(page: ft.Page):
    # Componentes da interface
    nome_field = ft.TextField(
        label="Nome do Cliente",
        width=200,
        border_color=ft.Colors.BLUE_400
    )

    telefone_field = ft.TextField(
        label="Telefone",
        width=195,
        border_color=ft.Colors.BLUE_400
    )

    servico_dropdown = ft.Dropdown(
        label="Serviço",
        width=200,
        options=[ft.dropdown.Option(s['nome']) for s in salao.servicos],
        border_color=ft.Colors.BLUE_400
    )

    data_picker = ft.DatePicker(
        first_date=datetime.datetime.now(),
        last_date=datetime.datetime.now() + datetime.timedelta(days=365)
    )

    data_field = ft.TextField(
        label="(DD/MM/AAAA)",
        width=99,
        read_only=False,
        border_color=ft.Colors.BLUE_400
    )

    horario_dropdown = ft.Dropdown(
        label="Horário",
        width=145,
        options=[ft.dropdown.Option(h) for h in salao.horarios_disponiveis],
        border_color=ft.Colors.BLUE_400
    )
```

```
et-app > src > salao_agendamento (3).py > SalaoAgendamento > adicionar_agendamento
```

```python
def main(page: ft.Page):
    page.title = "Salão de Cabeleireiro - Sistema de Agendamento"
    page.theme_mode = ft.ThemeMode.LIGHT
    page.window.width = 520
    page.window.height = 900
    page.window.resizable = False
    #trocar cor de fundo
    page.bgcolor = ft.Colors.LIGHT_BLUE_50
    #trocar cor de borda da janela
    page.window.bgcolor = ft.Colors.BLUE_200

    # Instância da classe de agendamento
    salao = SalaoAgendamento()

    # Componente de mensagem de status
    status_message = ft.Text(
        "",
        size=14,
        weight=ft.FontWeight.BOLD,
        text_align=ft.TextAlign.CENTER
    )

    def mostrar_mensagem(texto, cor=ft.Colors.BLUE_600):
        """Exibe uma mensagem de status"""
        status_message.value = texto
        status_message.color = cor
        page.update()

        # Remove a mensagem após 3 segundos
        import threading
        def limpar_mensagem():
            import time
            time.sleep(3)
            status_message.value = ""
            page.update()

        threading.Thread(target=limpar_mensagem, daemon=True).start()
```

- Arquivo JSON como banco de dados

Amostra de clintes :

- agendados
- conlcuidos
- removidos