Nome: André Machado Silva

Repositório no GitHub:
https://github.com/andrems1/onboarding-LIPAI/tree/main/Projeto%201/projeto_E

Código fonte:

- **src**

main.py

```
from models import Sala, Reserva
import repositorios as repo
import validacoes as val

def menu():
    print("\n=== SISTEMA DE RESERVAS DE SALAS ===")
    print("1. Cadastrar Sala")
    print("2. Listar Salas")
    print("3. Fazer Reserva")
    print("4. Listar Reservas (Geral)")
    print("5. Listar Reservas por Data")
    print("6. Excluir Sala")
    print("7. Excluir Reserva")
    print("0. Sair")
    return input("\nOpção: ")

def cadastrar_sala():
    print("\n--- Nova Sala ---")
    id_sala = str(len(repo.listar_salas()) + 1)
    nome = input("Nome da sala: ")
    cap = input("Capacidade: ")
    tipo = input("Tipo (Lab/Aula): ")

    try:
        nova_sala = Sala(id_sala, nome, cap, tipo)
        repo.salvar_sala(nova_sala)
        print("Sala cadastrada com sucesso!")
    except ValueError as e:
        print(f"Erro: {e}")

def fazer_reserva():
    print("\n--- Nova Reserva ---")
    salas = repo.listar_salas()
```

```
    if not salas:
        print("Nenhuma sala cadastrada.")
        return

    for s in salas:
        print(s)

    id_sala = input("Digite o ID da sala desejada: ")
    sala_selecionada = repo.buscar_sala_por_id(id_sala)

    if not sala_selecionada:
        print("Sala não encontrada.")
        return

    # Coletar dados
    resp = input("Responsável: ")
    data = input("Data (AAAA-MM-DD): ")
    inicio = input("Horário Início (HH:MM): ")
    fim = input("Horário Fim (HH:MM): ")

    id_res = str(len(repo.listar_reservas()) + 1)

    try:
        nova_reserva = Reserva(id_res, sala_selecionada, resp,
data, inicio, fim)

        # 3. Validar conflito
        todas_reservas = repo.listar_reservas()
        if val.verificar_conflito(nova_reserva, todas_reservas):
            print("ERRO: Conflito de horário! Já existe reserva
neste período.")
        else:
            repo.salvar_reserva(nova_reserva)
            print("Reserva realizada com sucesso!")

    except ValueError as e:
        print(f"Erro nos dados: {e}")

def listar_reservas_filtro():
    print("\n--- Consultar Reservas por Data ---")
```

```
    datas_disponiveis = repo.obter_datas_com_reservas()

    if not datas_disponiveis:
        print("Nenhuma reserva cadastrada.")
        return

    print("\nDatas com reservas feitas:")
    for i, data in enumerate(datas_disponiveis, 1):
        print(f"{i}. {data}")


    try:
        escolha = input("\nDigite o número da data desejada: ")
        indice = int(escolha) - 1

        if indice < 0 or indice >= len(datas_disponiveis):
            print("Opção inválida.")
            return

        data_selecionada = datas_disponiveis[indice]


        todas = repo.listar_reservas()
        reservas_data = [r for r in todas if r.data ==
data_selecionada]

        print(f"\nReservas para {data_selecionada}:")
        if reservas_data:
            for r in reservas_data:
                print(f"- {r}")
        else:
            print("Nenhuma reserva encontrada.")

    except ValueError:
        print("Opção inválida. Digite um número.")

def excluir_sala():
    print("\n--- Excluir Sala ---")
    salas = repo.listar_salas()
```

```
    if not salas:
        print("Nenhuma sala cadastrada.")
        return

    for s in salas:
        print(s)

    id_sala = input("\nDigite o ID da sala a excluir: ")

    if repo.excluir_sala(id_sala):
        print("Sala excluída com sucesso!")
    else:
        print("Sala não encontrada.")

def excluir_reserva():
    print("\n--- Excluir Reserva ---")
    reservas = repo.listar_reservas()

    if not reservas:
        print("Nenhuma reserva cadastrada.")
        return

    for r in reservas:
        print(r)
        print()

    id_reserva = input("Digite o ID da reserva a excluir: ")

    if repo.excluir_reserva(id_reserva):
        print("Reserva excluída com sucesso!")
    else:
        print("Reserva não encontrada.")

if __name__ == "__main__":
    while True:
        op = menu()
        if op == "1":
            cadastrar_sala()
        elif op == "2":
            salas = repo.listar_salas()
            for s in salas: print(s)
```

```
        elif op == "3":
            fazer_reserva()
        elif op == "4":
            reservas = repo.listar_reservas()
            for r in reservas: print(r)
        elif op == "5":
            listar_reservas_filtro()
        elif op == "6":
            excluir_sala()
        elif op == "7":
            excluir_reserva()
        elif op == "0":
            break
        else:
            print("Opção inválida.")
```

models.py

```
class Sala:
    def __init__(self, id_sala, nome, capacidade, tipo):
        self.id = id_sala
        self.nome = nome
        self.capacidade = capacidade
        self.tipo = tipo

    @property
    def capacidade(self):
        return self._capacidade

    @capacidade.setter
    def capacidade(self, valor):
        if int(valor) <= 0:
            raise ValueError("A capacidade deve ser maior que
zero.")
        self._capacidade = int(valor)

    def __str__(self):
        return f"[{self.id}] {self.nome} ({self.tipo}) - Cap:
{self.capacidade} pessoas"


class Reserva:
```

```
    def __init__(self, id_reserva, sala_obj, responsavel, data,
horario_inicio, horario_fim):
        self.id = id_reserva
        self.sala = sala_obj
        self.responsavel = responsavel
        self.data = data
        self.inicio = horario_inicio
        self.fim = horario_fim

    @property
    def sala(self):
        return self._sala

    @sala.setter
    def sala(self, valor):
        if not isinstance(valor, Sala):
            raise ValueError("O objeto associado deve ser do tipo
Sala.")
        self._sala = valor

    def __str__(self):
        return (f"Reserva [{self.id}] - {self.data} |
{self.inicio} às {self.fim}\n"
                f"   Local: {self.sala.nome} | Resp:
{self.responsavel}")
```

repositorios.py

```
import os
from models import Sala, Reserva

ARQUIVO_SALAS = os.path.join("data", "salas.csv")
ARQUIVO_RESERVAS = os.path.join("data", "reservas.csv")

# SALAS
def salvar_sala(sala):
    # Cria a pasta
    os.makedirs("data", exist_ok=True)
    with open(ARQUIVO_SALAS, 'a', encoding='utf-8') as f:
        linha =
f"{sala.id},{sala.nome},{sala.capacidade},{sala.tipo}\n"
        f.write(linha)
```

```
def listar_salas():
    salas = []
    if not os.path.exists(ARQUIVO_SALAS):
        return salas

    with open(ARQUIVO_SALAS, 'r', encoding='utf-8') as f:
        for linha in f:
            linha = linha.strip()
            if not linha: continue
            dados = linha.split(',')
            s = Sala(dados[0], dados[1], dados[2], dados[3])
            salas.append(s)
    return salas

def buscar_sala_por_id(id_busca):
    salas = listar_salas()
    for s in salas:
        if s.id == id_busca:
            return s
    return None

def excluir_sala(id_sala):
    salas = listar_salas()
    salas_atualizadas = [s for s in salas if s.id != id_sala]

    if len(salas_atualizadas) == len(salas):
        return False

    os.makedirs("data", exist_ok=True)
    with open(ARQUIVO_SALAS, 'w', encoding='utf-8') as f:
        for sala in salas_atualizadas:
            linha =
f"{sala.id},{sala.nome},{sala.capacidade},{sala.tipo}\n"
            f.write(linha)

    reservas = listar_reservas()
    reservas_atualizadas = [r for r in reservas if r.sala.id !=
id_sala]

    with open(ARQUIVO_RESERVAS, 'w', encoding='utf-8') as f:
```

```
        for reserva in reservas_atualizadas:
            linha =
f"{reserva.id},{reserva.sala.id},{reserva.responsavel},{reserva.d
ata},{reserva.inicio},{reserva.fim}\n"
            f.write(linha)
    return True

# RESERVAS
def salvar_reserva(reserva):
    os.makedirs("data", exist_ok=True)
    with open(ARQUIVO_RESERVAS, 'a', encoding='utf-8') as f:
        linha =
f"{reserva.id},{reserva.sala.id},{reserva.responsavel},{reserva.d
ata},{reserva.inicio},{reserva.fim}\n"
        f.write(linha)

def listar_reservas():
    reservas = []
    if not os.path.exists(ARQUIVO_RESERVAS):
        return reservas

    with open(ARQUIVO_RESERVAS, 'r', encoding='utf-8') as f:
        for linha in f:
            linha = linha.strip()
            if not linha: continue

            dados = linha.split(',')
            id_res = dados[0]
            id_sala = dados[1]
            resp = dados[2]
            data = dados[3]
            ini = dados[4]
            fim = dados[5]
            sala_obj = buscar_sala_por_id(id_sala)

            if sala_obj:
                r = Reserva(id_res, sala_obj, resp, data, ini,
fim)
                reservas.append(r)
    return reservas
```

```
def buscar_reserva_por_id(id_busca):
    reservas = listar_reservas()
    for r in reservas:
        if r.id == id_busca:
            return r
    return None

def excluir_reserva(id_reserva):
    reservas = listar_reservas()
    reservas_atualizadas = [r for r in reservas if r.id !=
id_reserva]

    if len(reservas_atualizadas) == len(reservas):
        return False
    os.makedirs("data", exist_ok=True)
    with open(ARQUIVO_RESERVAS, 'w', encoding='utf-8') as f:
        for reserva in reservas_atualizadas:
            linha =
f"{reserva.id},{reserva.sala.id},{reserva.responsavel},{reserva.d
ata},{reserva.inicio},{reserva.fim}\n"
            f.write(linha)

    return True

def obter_datas_com_reservas():
    reservas = listar_reservas()
    datas = set()
    for r in reservas:
        datas.add(r.data)
    return sorted(list(datas))
```

validacoes.py

```
def verificar_conflito(nova_reserva, lista_reservas_existentes):

    for r in lista_reservas_existentes:
        if r.sala.id != nova_reserva.sala.id:
            continue
        if r.data != nova_reserva.data:
            continue
```

```
        if (nova_reserva.inicio < r.fim) and (nova_reserva.fim >
r.inicio):
            return True # conflito

    return False # sem conflito
```

- **data**

Gerados em execução, exemplos:

salas.csv
1,Minas Gerais,10,aula
3,Estruturas de Dados,25,aula
4,Redes de Computadores,20,lab
5,Banco de Dados,28,aula
6,Engenharia de Software,35,aula
7,Inteligência Artificial,18,lab
8,Sistemas Operacionais,40,aula
9,Programação I,32,aula
10,Computação Gráfica,15,lab
11,Arquitetura de Computadores,22,aula
12,Segurança da Informação,16,lab

reservas.csv
1,1,André Machado,05/02/2023,18:00,23:00