

TRABALHO DE SISTEMAS OPERACIONAIS

ALUNO: Henrique José Ferreira dos Santos

TURMA: 2º período - ADS

```
alunoifro@LAPTOP-D2SFCDB7:~$ mkdir -p ~/aero70/{dados,logs,relatorios,scripts,torre,docs}
alunoifro@LAPTOP-D2SFCDB7:~$ cd ~/aero70
alunoifro@LAPTOP-D2SFCDB7:~/aero70$ python3 -m venv .venv
The virtual environment was not created successfully because ensurepip is not
available. On Debian/Ubuntu systems, you need to install the python3-venv
package using the following command.
```

```
apt install python3.12-venv
```

You may need to use sudo with that command. After installing the python3-venv package, recreate your virtual environment.

Failing command: /home/alunoifro/aero70/.venv/bin/python3

```
alunoifro@LAPTOP-D2SFCDB7:~/aero70$ sudo apt install python3.13-venv
[sudo] password for alunoifro:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package python3.13-venv
E: Couldn't find any package by glob 'python3.13-venv'
alunoifro@LAPTOP-D2SFCDB7:~/aero70$ sudo apt install python3.12-venv
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpython3.12-minimal libpython3.12-stdlib libpython3.12t64
  python3-pip-whl python3-setuptools-whl python3.12
  python3.12-minimal
Suggested packages:
  python3.12-doc binfmt-support
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.12-venv
The following packages will be upgraded:
```

```
Setting up python3.12-venv (3.12.3-1ubuntu0.8) ...
Processing triggers for systemd (255.4-1ubuntu8.10) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.5) ...
alunoifro@LAPTOP-D2SFCDB7:~/aero70$ python3 -m venv .venv
alunoifro@LAPTOP-D2SFCDB7:~/aero70$ source .venv/bin/activate
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd dados
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch planos_voo.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch pistas.txt
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch frota.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch pilotos.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch metar.txt
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch notam.txt
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cd..
cd..: command not found
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd logs
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/logs$ touch torre.log
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/logs$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd relatorios
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/relatorios$ touch operacao_YYYYMMDD.txt
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/relatorios$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd torre
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ touch torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd docs
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/docs$ touch README.md
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/docs$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd dados
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "voo,origem,destino,etd,eta,aeronave,tipo,prioridade,pista_pref" > planos_voo.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "ALT123,PVH,MAO,13:20,14:45,B727,COMERCIAL,1,10/28" >> planos_voo.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "ALT901,STM,PVH,14:05,15:20,EMB-110,EMERGENCIA,3,01/19" >> planos_voo.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat planos_voo.csv
voo,origem,destino,etd,eta,aeronave,tipo,prioridade,pista_pref
```



```

CSV
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "ALT123,PVH,MAO,13:20,14:45,B727,COMERCIAL,1,10/28" >> planos_voo.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "ALT901,STM,PVH,14:05,15:20,EMB-110,EMERGENCIA,3,01/19" >> planos_voo.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat planos_voo.csv
voo,origem,destino,etd,eta,aeronave,tipo,prioridade,pista_pref
ALT123,PVH,MAO,13:20,14:45,B727,COMERCIAL,1,10/28
ALT901,STM,PVH,14:05,15:20,EMB-110,EMERGENCIA,3,01/19
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "10/28,ABERTA" > pistas.txt
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "01/19,ABERTA" >> pistas.txt
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat pistas.txt
10/28,ABERTA
01/19,ABERTA
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "aeronave,comprimento_min_pista,obs" > frota.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "B727,2200,desempenho normal" >> frota.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "EMB-110,1200,leve" >> frota.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "DC-10,3000,pista longa" >> frota.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat frota.csv
aeronave,comprimento_min_pista,obs
B727,2200,desempenho normal
EMB-110,1200,leve
DC-10,3000,pista longa
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "matricula,nome,licenca,habilitacao,validade" > pilotos.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "P001,João Silva,ATPL,B727,1979-06-30" >> pilotos.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "P002,Maria Souza,CPL,EMB-110,1978-12-15" >> pilotos.csv
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat pilotos.csv
matricula,nome,licenca,habilitacao,validade
P001,João Silva,ATPL,B727,1979-06-30
P002,Maria Souza,CPL,EMB-110,1978-12-15
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "13:00 VENTO 090/12KT VIS 7KM CHUVA LEVE" > metar.txt
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "14:00 VENTO 120/15KT VIS 5KM CHUVA MOD" >> metar.txt
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat metar.txt
13:00 VENTO 090/12KT VIS 7KM CHUVA LEVE
14:00 VENTO 120/15KT VIS 5KM CHUVA MOD
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "PISTA 01/19 FECHADA 14:00-16:00 MANUTENCAO" > notam.txt
(.env) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ echo "RADIO VHF SETOR NORTE INTERMITENTE 15:00-15:30" >> notam.txt

```

```
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cat notam.txt
PISTA 01/19 FECHADA 14:00-16:00 MANUTENCAO
RADIO VHF SETOR NORTE INTERMITENTE 15:00-15:30
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ touch logs/torre.log
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd logs
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/logs$ touch torre.log
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/logs$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd dados
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch fila_decolagem.txt
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ touch fila_pouso.txt
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/dados$ cd ..
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70$ cd torre
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py importar-dados
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py listar [--por=voo|etd|tipo|prioridade]
Command 'tipo' not found, did you mean:
  command 'tips' from snap tips (0.0.9)
  command 'tio' from snap tio (3.9)
  command 'tipc' from deb iproute2 (6.1.0-1ubuntu6.2)
  command 'tio' from deb tio (2.7-1)
See 'snap info <snapname>' for additional versions.
Command 'etd' not found, did you mean:
  command 'std' from snap std (1.0.1)
  command 'rtd' from deb skycat (3.1.2+starlink1~b+dfsg-7build1)
  command 'eid' from deb id-utils (4.6.28-20200521ss15dab)
  command 'atd' from deb at (3.2.5-1ubuntu1)
  command 'emd' from deb emd (1.0.1-3ubuntu0.24.04.3)
  command 'etm' from deb etm (3.2.39-1)
  command 'etcd' from deb etcd-server (3.4.30-1ubuntu0.24.04.3)
  command 'ed' from deb ed (1.19-1)
  command 'eta' from deb eta (1.0.1-1)
  command 'etr' from deb extremetuxracer (0.8.3-1)
  command 'etw' from deb etw (3.6+svn162-6)
  command 'td' from deb textdraw (0.2+ds-0+nmu1build3)
See 'snap info <snapname>' for additional versions.
```



```
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py importar-dados
Erro: arquivo obrigatório não encontrado: dados/planos_voo.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ ls ~/aero70/dados
fila_decolagem.txt  fila_pouso.txt  frota.csv  metar.txt  notam.txt  pilotos.csv  pistas.txt  planos_voo.csv
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ ls dados/
ls: cannot access 'dados/': No such file or directory
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py importar-dados
Dados importados com sucesso!
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py listar --por=prioridade
Nenhum voo importado. Execute importar-dados primeiro.
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py importar-dados
Dados importados com sucesso!
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py listar --por=prioridade
Nenhum voo importado. Execute importar-dados primeiro.
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py listar --por=prioridade
voo      origem  destino  etd      eta      aeronave      tipo      prioridade  pista_pref
-----
ALT901   STM     PVH      14:05    15:20    EMB-110      EMERGENCIA    3          01/19
ALT123   PVH     MAO      13:20    14:45    B727         COMERCIAL     1          10/28
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py enfileirar decolagem --voo ALT123
Voo ALT123 adicionado à fila de decolagem
```

```
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py enfileirar decolagem --voo ALT123
Voo ALT123 adicionado à fila de decolagem
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py enfileirar pouso --voo ALT901
Voo ALT901 adicionado à fila de pouso
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py autorizar decolagem --pista 10/28
decolagem ALT123 - AUTORIZADO
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py autorizar pouso --pista 01/19
NEGADO: NOTAM ativo - PISTA 01/19 FECHADA 14:00-16:00 MANUTENCAO
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py status
=== STATUS DA TORRE ===
```

Pistas:
10/28: ABERTA
01/19: ABERTA

Tamanho da fila de decolagem: 0
Próximos 3 voos de decolagem:

Tamanho da fila de pouso: 1
Próximos 3 voos de pouso:
ALT901;14:05;3;01/19

NOTAM ativos:
PISTA 01/19 FECHADA 14:00-16:00 MANUTENCAO
RADIO VHF SETOR NORTE INTERMITENTE 15:00-15:30

METAR ativos:
13:00 VENTO 090/12KT VIS 7KM CHUVA LEVE
14:00 VENTO 120/15KT VIS 5KM CHUVA MOD

```
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ nano torre.py
(.venv) alunoifro@LAPTOP-D2SFCDB7:~/aero70/torre$ python3 torre.py relatorio
Relatório gerado: /home/alunoifro/aero70/torre/./relatorios/operacao_20250918.txt
```

GNU nano 7.2

torre.py

```
import os
import csv
from datetime import datetime
import argparse

# Pastas e arquivos
DADOS_DIR = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados")
LOG_FILE = "logs/torre.log"

arquivos_obrigatorios = {
    "planos_voo": "planos_voo.csv",
    "pistas": "pistas.txt",
    "metar": "metar.txt",
    "notam": "notam.txt",
    "frota": "frota.csv",
    "pilotos": "pilotos.csv"
}

# Estruturas de dados globais
voos = []
pistas = {}
metar = []
notams = []
frota = {}
pilotos = {}

def log(mensagem):
    os.makedirs("logs", exist_ok=True)
    with open(LOG_FILE, "a") as f:
        f.write(f"{datetime.now().isoformat()} - {mensagem}\n")
```

[Read 457 lines]

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy


```
def importar_dados():
    #❑ Verifica se todos os arquivos existem
    for key, file in arquivos_obrigatorios.items():
        path = os.path.join(DADOS_DIR, file)
        if not os.path.exists(path):
            msg = f"Erro: arquivo obrigatório não encontrado: {path}"
            print(msg)
            log(msg)
            return False

    #❑ Ler planos_voo.csv
    global voos
    path_voos = os.path.join(DADOS_DIR, arquivos_obrigatorios["planos_voo"])
    with open(path_voos, newline="") as f:
        reader = csv.DictReader(f)
        voos = [row for row in reader]

    #❑ Ler pistas.txt
    global pistas
    path_pistas = os.path.join(DADOS_DIR, arquivos_obrigatorios["pistas"])
    pistas = {}
    with open(path_pistas) as f:
        for line in f:
            if line.strip():
                pista, status = line.strip().split(",")
                pistas[pista] = status

    #❑ Ler metar.txt
    global metar
    path_metar = os.path.join(DADOS_DIR, arquivos_obrigatorios["metar"])
    with open(path_metar) as f:
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy


```
metar = [line.strip() for line in f if line.strip()]

#5 Ler notam.txt
global notams
path_notam = os.path.join(DADOS_DIR, arquivos_obrigatorios["notam"])
with open(path_notam) as f:
    notams = [line.strip() for line in f if line.strip()]

#6 Ler frota.csv
global frota
path_frota = os.path.join(DADOS_DIR, arquivos_obrigatorios["frota"])
with open(path_frota, newline="") as f:
    reader = csv.DictReader(f)
    frota = {row["aeronave"]: row for row in reader}

#7 Ler pilotos.csv
global pilotos
path_pilotos = os.path.join(DADOS_DIR, arquivos_obrigatorios["pilotos"])
with open(path_pilotos, newline="") as f:
    reader = csv.DictReader(f)
    pilotos = {row["matricula"]: row for row in reader}

#8 Criar filas iniciais (vazias)
os.makedirs(DADOS_DIR, exist_ok=True)
with open(os.path.join(DADOS_DIR, "fila_decolagem.txt"), "w") as f:
    pass
with open(os.path.join(DADOS_DIR, "fila_pouso.txt"), "w") as f:
    pass

print("Dados importados com sucesso!")
log("Dados importados com sucesso")
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy

```
print("Dados importados com sucesso!")
log("Dados importados com sucesso")
return True
```

```
def listar(ordenar_por="voo"):
    path_voos = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "planos_voo.csv")
    if not os.path.exists(path_voos):
        print(f"Arquivo {path_voos} não encontrado. Execute importar-dados primeiro.")
        return

    # Ler voos do CSV
    with open(path_voos, newline="") as f:
        reader = csv.DictReader(f)
        voos = [row for row in reader]

    if not voos:
        print("Nenhum voo encontrado no CSV.")
        return

    # Ordenar
    if ordenar_por in ["voo", "etd", "tipo", "prioridade"]:
        if ordenar_por == "prioridade":
            lista_ordenada = sorted(voos, key=lambda x: int(x["prioridade"]), reverse=True)
        else:
            lista_ordenada = sorted(voos, key=lambda x: x[ordenar_por])
    else:
        lista_ordenada = voos

    # Imprimir tabela
    colunas = ["voo", "origem", "destino", "etd", "eta", "aeronave", "tipo", "prioridade", "pista_pref"]
    print("\t".join(colunas))
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy

GNU nano 7.2

torre.py

```
colunas = ["voo", "origem", "destino", "etd", "eta", "aeronave", "tipo", "prioridade", "pista_pref"]
print("\t".join(colunas))
print("-"*80)
for voo in lista_ordenada:
    print("\t".join([voo[c] for c in colunas]))
```

Função auxiliar para log

```
def escrever_log(mensagem):
    log_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "logs")
    os.makedirs(log_dir, exist_ok=True)
    log_path = os.path.join(log_dir, "torre.log")
    with open(log_path, "a") as f:
        f.write(f"{datetime.now().strftime('%Y-%m-%d %H:%M:%S')} - {mensagem}\n")
```

Função para ler os voos

```
def ler_voos():
    path_voos = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "planos_voo.csv")
    if not os.path.exists(path_voos):
        print(f"Arquivo {path_voos} não encontrado. Execute importar-dados primeiro.")
        return []
    with open(path_voos, newline="") as f:
        reader = csv.DictReader(f)
        return [row for row in reader]
```

Função para ler pilotos

```
def ler_pilotos():
    path_pilotos = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "pilotos.csv")
    with open(path_pilotos, newline="") as f:
        reader = csv.DictReader(f)
        return {row["matricula"]: row for row in reader}
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy

```
# Função para ler frota
def ler_frota():
    path_frota = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "frota.csv")
    with open(path_frota, newline="") as f:
        reader = csv.DictReader(f)
        return {row["aeronave"]: row for row in reader}

# Função principal de enfileirar
def enfileirar(tipo, codigo_voo):
    voos = ler_voos()
    pilotos = ler_pilotos()
    frota = ler_frota()

    # Encontrar o voo
    voo = next((v for v in voos if v["voo"] == codigo_voo), None)
    if not voo:
        print(f"Voo {codigo_voo} não encontrado no CSV")
        escrever_log(f"Falha ao enfileirar: voo {codigo_voo} não encontrado")
        return

    # Validar piloto
    matricula_piloto = voo.get("piloto", "") # Ajuste se você tiver a coluna piloto
    if matricula_piloto and matricula_piloto in pilotos:
        piloto = pilotos[matricula_piloto]
        # Aqui você poderia validar licença e habilitação
        # Exemplo: checar validade
        # if piloto["licenca_vencida"]:
        #     print("Piloto inválido")
    else:
        piloto = None
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy


```
# Validar aeronave
aeronave = voo["aeronave"]
if aeronave not in frota:
    print(f"Aeronave {aeronave} não cadastrada na frota")
    escrever_log(f"Falha ao enfileirar: aeronave {aeronave} não cadastrada")
    return

# Escolher arquivo da fila
fila_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", f"fila_{tipo}.txt")
os.makedirs(os.path.dirname(fila_file), exist_ok=True)

# Adicionar voo à fila
with open(fila_file, "a") as f:
    linha = f"{voo['voo']};{voo['etd']};{voo['prioridade']};{voo['pista_pref']}\n"
    f.write(linha)

print(f"Voo {codigo_voo} adicionado à fila de {tipo}")
escrever_log(f"Voo {codigo_voo} enfileirado em {tipo}")

def ler_pistas():
    path = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "pistas.txt")
    with open(path, newline="") as f:
        reader = csv.reader(f)
        return [row[0]: row[1] for row in reader] # {'10/28': 'ABERTA', ...}

def ler_notam():
    path = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "notam.txt")
    if not os.path.exists(path):
        return []
    with open(path) as f:
        return [line.strip() for line in f]
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy

```
def ler_metar():
    path = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", "metar.txt")
    if not os.path.exists(path):
        return []
    with open(path) as f:
        return [line.strip() for line in f]

def autorizar(tipo, pista):
    fila_file = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados", f"fila_{tipo}.txt")
    if not os.path.exists(fila_file):
        print(f"Fila {tipo} está vazia")
        return

    # Ler filas
    with open(fila_file) as f:
        voos_fila = [line.strip() for line in f if line.strip()]

    if not voos_fila:
        print(f"Fila {tipo} está vazia")
        return

    pistas_status = ler_pistas()
    notams = ler_notam()
    metars = ler_metar()

    # Pegando o primeiro voo elegivel
    voo_info = voos_fila[0].split(";") # voo;hora;prioridade;pista_atribuida
    codigo_voo, hora, prioridade, pista_atribuida = voo_info

    # Verifica pista aberta
```

```
^G Help
^X Exit
```

```
^O Write Out
^R Read File
```

```
^W Where Is
^_ Replace
```

```
^K Cut
^U Paste
```

```
^T Execute
^J Justify
```

```
^C Location
^_ Go To Line
```

```
M-U Undo
M-E Redo
```

```
M-A Set Mark
M-6 Copy
```



```
# Verifica pista aberta
if pista not in pistas_status or pistas_status[pista] != "ABERTA":
    resultado = f"NEGADO: pista {pista} não está ABERTA"
    print(resultado)
    escrever_log(f"{tipo} {codigo_voo} - {resultado}")
    return

# Verifica NOTAM
for n in notams:
    if pista in n and "FECHADA" in n:
        resultado = f"NEGADO: NOTAM ativo - {n}"
        print(resultado)
        escrever_log(f"{tipo} {codigo_voo} - {resultado}")
        return

# Verifica clima (simples: VIS < 6KM bloqueia mais de 1 operação)
vis_restrita = False
for m in metars:
    if "VIS" in m:
        vis = int(m.split("VIS")[1].split("KM")[0].strip())
        if vis < 6:
            vis_restrita = True
            break

if vis_restrita and len(voos_fila) > 1:
    resultado = f"NEGADO: visibilidade restrita, apenas 1 operação permitida"
    print(resultado)
    escrever_log(f"{tipo} {codigo_voo} - {resultado}")
    return

# Se passou todas as checagens → AUTORIZADO
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy

```
# Se passou todas as checagens → AUTORIZADO
resultado = "AUTORIZADO"
print(f"{tipo} {codigo_voo} - {resultado}")
escrever_log(f"{tipo} {codigo_voo} - {resultado}")
```

```
# Remove voo da fila
voos_fila.pop(0)
with open(fila_file, "w") as f:
    for v in voos_fila:
        f.write(v + "\n")
```

```
def status():
    # Pastas e arquivos
    dados_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "dados")
    fila_decolagem = os.path.join(dados_dir, "fila_decolagem.txt")
    fila_pouso = os.path.join(dados_dir, "fila_pouso.txt")
    pistas_file = os.path.join(dados_dir, "pistas.txt")
    notam_file = os.path.join(dados_dir, "notam.txt")
    metar_file = os.path.join(dados_dir, "metar.txt")
```

```
# Ler filas
def ler_fila(path):
    if os.path.exists(path):
        with open(path) as f:
            return [line.strip() for line in f if line.strip()]
    return []
```

```
fila_d = ler_fila(fila_decolagem)
fila_p = ler_fila(fila_pouso)
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy


```
# Ler pistas
pistas = {}
if os.path.exists(pistas_file):
    with open(pistas_file) as f:
        for line in f:
            p, status = line.strip().split(",")
            pistas[p] = status

# Ler NOTAM
notams = []
if os.path.exists(notam_file):
    with open(notam_file) as f:
        notams = [line.strip() for line in f]

# Ler METAR
metars = []
if os.path.exists(metar_file):
    with open(metar_file) as f:
        metars = [line.strip() for line in f]

# Imprimir status
print("=== STATUS DA TORRE ===\n")

print("Pistas:")
for p, st in pistas.items():
    print(f"{p}: {st}")
print()

print(f"Tamanho da fila de decolagem: {len(fila_d)}")
print(f"Próximos 3 voos de decolagem:")
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy

```
print(f"Tamanho da fila de decolagem: {len(fila_d)}")
print(f"Próximos 3 voos de decolagem:")
for v in fila_d[:3]:
    print(f"  {v}")
print()
```

```
print(f"Tamanho da fila de pouso: {len(fila_p)}")
print(f"Próximos 3 voos de pouso:")
for v in fila_p[:3]:
    print(f"  {v}")
print()
```

```
print("NOTAM ativos:")
for n in notams:
    print(f"  {n}")
print()
```

```
print("METAR ativos:")
for m in metars:
    print(f"  {m}")
print()
```

```
def relatorio():
    import re
    from datetime import datetime
```

```
    logs_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "logs")
    log_file = os.path.join(logs_dir, "torre.log")
```

```
^G Help
^X Exit
```

```
^O Write Out
^R Read File
```

```
^W Where Is
^_ Replace
```

```
^K Cut
^U Paste
```

```
^T Execute
^J Justify
```

```
^C Location
^_ Go To Line
```

```
M-U Undo
M-E Redo
```

```
M-A Set Mark
M-6 Copy
```

```
if not os.path.exists(log_file):
    print("Nenhum log encontrado para gerar relatório.")
    return

# Contadores
autorizados = 0
negados = 0
motivos_negado = {}
emergencias = 0

with open(log_file) as f:
    for line in f:
        line = line.strip()
        if not line:
            continue
        # AUTORIZADO
        if "AUTORIZADO" in line:
            autorizados += 1
            if "EMERGENCIA" in line.upper():
                emergencias += 1
        # NEGADO
        elif "NEGADO" in line.upper():
            negados += 1
            # Captura motivo
            match = re.search(r"NEGADO: (.+)$", line)
            if match:
                motivo = match.group(1)
                motivos_negado[motivo] = motivos_negado.get(motivo, 0) + 1
```

```
# Criar pasta relatorios se não existir
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/_ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy


```
# Criar pasta relatorios se não existir
rel_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), "..", "relatorios")
os.makedirs(rel_dir, exist_ok=True)

# Nome do arquivo: operacao_YYYYMMDD.txt
hoje = datetime.now().strftime("%Y%m%d")
rel_file = os.path.join(rel_dir, f"operacao_{hoje}.txt")

with open(rel_file, "w") as f:
    f.write(f"RELATÓRIO DO TURNO - {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")
    f.write(f"Total de voos autorizados: {autorizados}\n")
    f.write(f"Total de voos negados: {negados}\n")
    f.write(f"Emergências atendidas: {emergencias}\n\n")
    f.write(f"Motivos mais comuns de negação:\n")
    for motivo, count in motivos_negado.items():
        f.write(f"  {motivo}: {count}\n")



print(f"Relatório gerado: {rel_file}")

# Exemplo de teste
if __name__ == "__main__":
    parser = argparse.ArgumentParser(prog="torre")
    subparsers = parser.add_subparsers(dest="comando")

    # Comando importar-dados
    subparsers.add_parser("importar-dados")
    # Comando para listar
    listar_parser = subparsers.add_parser("listar")
```

 ^G Help	 ^O Write Out	 ^W Where Is	 ^K Cut
 ^X Exit	 ^R Read File	 ^\ Replace	 ^U Paste

 ^T Execute
 ^J Justify

 ^C Location
 ^_ Go To Line

 M-U Undo
 M-E Redo

 M-A Set Mark
 M-6 Copy

```
# Comando autorizar
auth_parser = subparsers.add_parser("autorizar")
auth_parser.add_argument("tipo", choices=["decolagem", "pouso"])
auth_parser.add_argument("--pista", required=True)

# Comando status
status_parser = subparsers.add_parser("status")

# Comando relatorio
rel_parser = subparsers.add_parser("relatorio")

args = parser.parse_args()

if args.comando == "importar-dados":
    importar_dados()

elif args.comando == "listar":
    listar(args.por)

elif args.comando == "enfileirar":
    enfileirar(args.tipo, args.voo)

elif args.comando == "autorizar":
    autorizar(args.tipo, args.pista)

elif args.comando == "status":
    status()

elif args.comando == "relatorio":
    relatorio()
```

^G Help
^X Exit

^O Write Out
^R Read File

^W Where Is
^_ Replace

^K Cut
^U Paste

^T Execute
^J Justify

^C Location
^/ Go To Line

M-U Undo
M-E Redo

M-A Set Mark
M-6 Copy