Turma	01	**Semestre**	2024.1

Equipe	NeverMind

Nome	Matrícula
Elias Faria de Oliveira	221007706
João Eduardo Pereira Rabelo	180053299
Yan Lucas Souza Guimarães	222006220
Daniel dos Santos Barros de Sousa	211030980

Realizando a atividade AAE-04:

- Repositório da atividade: NeverMind

Ciclo 1: Teste Funcionalidade “Sair”.

- Código do teste:

```python
def test_sair():
    agenda = Agenda()
    assert agenda.running is True
    agenda.sair()
    assert agenda.running is False
```

- Código da funcionalidade:

```python
def sair(self):
    self.running = False
```

- Teste em Execução:

```
PS C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind> python -m pytest .\Test\test_sair.py
============================= test session starts =============================
platform win32 -- Python 3.12.3, pytest-8.2.2, pluggy-1.5.0
rootdir: C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind
plugins: cov-5.0.0
collected 1 item

Test\test_sair.py .                                                      [100%]

============================== 1 passed in 0.01s ==============================
```

Ciclo 2: Teste Funcionalidade “Adicionar” um evento .

- Código do teste:

```python
import pytest
from datetime import datetime
from Source.Agenda import Agenda

@pytest.fixture
def agenda():
    return Agenda()

def test_add_Eventos_sucesso(agenda,capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert out.strip() == 'Evento adicionado com sucesso.'
    assert len(agenda.eventos) == 1
    assert agenda.eventos[0]['nome'] == "Final dos 100m"
    assert agenda.eventos[0]['inicio'] == datetime(2024, 7, 24, 10, 0)
    assert agenda.eventos[0]['fim'] == datetime(2024, 7, 24, 11, 0)
```

- Código da funcionalidade:

```python
    def add_Eventos(self, nome, inicio, fim):
        inicio_dt = datetime.datetime.strptime(inicio, "%Y-%m-%d %H:%M")
        fim_dt = datetime.datetime.strptime(fim, "%Y-%m-%d %H:%M")
        self.eventos.append({
        'nome': nome,
        'inicio': inicio_dt,
        'fim': fim_dt
        })
        self.qtd_eventos += 1
        print("Evento adicionado com sucesso.")
```

- Teste em Execução:

```
============================= test session starts =============================
platform win32 -- Python 3.12.3, pytest-8.2.2, pluggy-1.5.0
rootdir: C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind
plugins: cov-5.0.0
collected 1 item

Test\test_adicionar.py .                                                 [100%]

============================== 1 passed in 0.02s ==============================
```

Ciclo 3: Teste Funcionalidade “Adicionar” com data inválida .

- Código do teste:

```python

def test_add_Eventos_data_invalida(agenda,capsys):
    agenda.add_Eventos("Final dos 100m", "2024-13-35 11:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert out.strip() == "Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'."
    assert len(agenda.eventos) == 0
```

- Código da funcionalidade:

```python
    def add_Eventos(self, nome, inicio, fim):
        inicio_dt = 0
        fim_dt = 0
        erro = 0

        try:
            inicio_dt = datetime.datetime.strptime(inicio, "%Y-%m-%d %H:%M")
            fim_dt = datetime.datetime.strptime(fim, "%Y-%m-%d %H:%M")
        except ValueError:
            print("Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'.")
            erro = 1

        if erro == 0:
            self.eventos.append({
            'nome': nome,
            'inicio': inicio_dt,
            'fim': fim_dt
            })
            self.qtd_eventos += 1
            print("Evento adicionado com sucesso.")
```

- Teste em Execução:

```
PS C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind> python -m pytest .\Test\test_adicionar.py
============================= test session starts =============================
platform win32 -- Python 3.12.3, pytest-8.2.2, pluggy-1.5.0
rootdir: C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind
plugins: cov-5.0.0
collected 2 items

Test\test_adicionar.py ..                                                [100%]

============================== 2 passed in 0.04s ==============================
```

Ciclo 4: Teste Funcionalidade "Adicionar" com data inicial superior a final.

- Código do teste:

```python
def test_add_Eventos_hora_inicio_maior_que_fim(agenda,capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 12:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert out.strip() == "A hora de inicio deve ser anterior a hora de termino."
    assert len(agenda.eventos) == 0
```

- Código da funcionalidade:

```python
    def add_Eventos(self, nome, inicio, fim):
        inicio_dt = 0
        fim_dt = 0
        erro = 0

        try:
            inicio_dt = datetime.datetime.strptime(inicio, "%Y-%m-%d %H:%M")
            fim_dt = datetime.datetime.strptime(fim, "%Y-%m-%d %H:%M")
        except ValueError:
            print("Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'.")
            erro = 1

        if erro == 0:
            if inicio_dt >= fim_dt:
                    print("A hora de inicio deve ser anterior a hora de termino.")
                    erro = 1
            else:
                self.eventos.append({
                'nome': nome,
                'inicio': inicio_dt,
                'fim': fim_dt
                })
                self.qtd_eventos += 1
                print("Evento adicionado com sucesso.")
```

- Teste em Execução:

```
PS C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind> python -m pytest .\Test\test_adicionar.py
============================================================ test session starts ============================================================
platform win32 -- Python 3.12.3, pytest-8.2.2, pluggy-1.5.0
rootdir: C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind
plugins: cov-5.0.0
collected 3 items

Test\test_adicionar.py ...                                                                                                          [100%]

============================================================= 3 passed in 0.02s =============================================================
```

Ciclo 5: Teste Funcionalidade “Adicionar” eventos com conflito.

- Código do teste:

```python

def test_add_Eventos_conflito(agenda):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    agenda.add_Eventos("Final dos 200m", "2024-07-24 10:30", "2024-07-24 11:30")
    assert len(agenda.eventos) == 1
```

- Código da funcionalidade:

```python
    def add_Eventos(self, nome, inicio, fim):
        inicio_dt = 0
        fim_dt = 0
        erro = 0

        try:
            inicio_dt = datetime.datetime.strptime(inicio, "%Y-%m-%d %H:%M")
            fim_dt = datetime.datetime.strptime(fim, "%Y-%m-%d %H:%M")
        except ValueError:
            print("Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'.")
            erro = 1

        if erro == 0:
            for evento in self.eventos:
                if not (fim_dt <= evento['inicio'] or inicio_dt >= evento['fim']):
                    print("Conflito de agendamento detectado.")
                    erro = 1
                    break
            if erro == 0:
                if inicio_dt >= fim_dt:
                        print("A hora de inicio deve ser anterior a hora de termino.")
                        erro = 1
                else:
                    self.eventos.append({
                    'nome': nome,
                    'inicio': inicio_dt,
                    'fim': fim_dt
                    })
                    self.qtd_eventos += 1
                    print("Evento adicionado com sucesso.")

```

- Teste em Execução:

```
PS C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind> python -m pytest .\Test\test_adicionar.py
============================================== test session starts ==============================================
platform win32 -- Python 3.12.3, pytest-8.2.2, pluggy-1.5.0
rootdir: C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind
plugins: cov-5.0.0
collected 4 items

Test\test_adicionar.py ....                                                                              [100%]

=============================================== 4 passed in 0.02s ===============================================
```

Ciclo 6: Teste Funcionalidade "Adicionar" com mais de um evento sem conflito.

- Código do teste:

```
34
35  def test_add_Eventos_sem_conflito(agenda):
36      agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
37      agenda.add_Eventos("Final dos 200m", "2024-07-24 11:30", "2024-07-24 12:30")
38      assert len(agenda.eventos) == 2
```

- Código da funcionalidade:

```
 9      def add_Eventos(self, nome, inicio, fim):
10          inicio_dt = 0
11          fim_dt = 0
12          erro = 0
13
14          try:
15              inicio_dt = datetime.datetime.strptime(inicio, "%Y-%m-%d %H:%M")
16              fim_dt = datetime.datetime.strptime(fim, "%Y-%m-%d %H:%M")
17          except ValueError:
18              print("Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'.")
19              erro = 1
20
21          if erro == 0:
22              for evento in self.eventos:
23                  if not (fim_dt <= evento['inicio'] or inicio_dt >= evento['fim']):
24                      print("Conflito de agendamento detectado.")
25                      erro = 1
26                      break
27              if erro == 0:
28                  if inicio_dt >= fim_dt:
29                          print("A hora de inicio deve ser anterior a hora de termino.")
30                          erro = 1
31                  else:
32                      self.eventos.append({
33                      'nome': nome,
34                      'inicio': inicio_dt,
35                      'fim': fim_dt
36                      })
37                      self.qtd_eventos += 1
38                      print("Evento adicionado com sucesso.")
```

- Teste em Execução:

```
PS C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind> python -m pytest .\Test\test_adicionar.py
============================================= test session starts =============================================
platform win32 -- Python 3.12.3, pytest-8.2.2, pluggy-1.5.0
rootdir: C:\Programacao\JoaoE\JoaoEduardoP\UNB_FGA\TESTES DE SOFTWARE\Modulo-4\NeverMind
plugins: cov-5.0.0
collected 5 items

Test\test_adicionar.py .....                                                                          [100%]

============================================== 5 passed in 0.02s ==============================================
```

Ciclo 7: Teste Funcionalidade "Mostrar" com nenhum evento.

- Código do teste:

```python
import pytest
from io import StringIO
from Source.Agenda import Agenda

def test_mostrar_sem_evento(capsys):
    agenda = Agenda()
    agenda.get_Eventos()
    out, _ = capsys.readouterr()
    assert out.strip() == 'Nenhum evento agendado.'
```

- Código da funcionalidade:

```python
    def get_Eventos(self):
        print('Nenhum evento agendado.')
```

Ciclo 8: Teste Funcionalidade "Mostrar" com apenas um evento.

- Código do teste:

```python
def test_mostrar_um_Evento(agenda, capsys):
    agenda.add_Eventos( nome: "Final dos 100m", inicio: "2024-07-24 10:00", fim: "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    agenda.get_Eventos()
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 1
    assert out.strip() == "Final dos 100m: 2024-07-24 10:00 a 2024-07-24 11:00"
```

- Código da funcionalidade:

```python
def get_Eventos(self):
    formato = "%Y-%m-%d %H:%M"
    if len(self.eventos) >= 1:
        for evento in self.eventos:
            print(f"{evento['nome']}: {evento['inicio'].strftime(formato)} a {evento['fim'].strftime(formato)}")
    else:
        print("Nenhum evento agendado.")

    return
```

Ciclo 9: Teste Funcionalidade "Mostrar" com dois ou mais eventos.

- Código do teste:

```python
def test_mostrar_um_Evento(agenda, capsys):
    agenda.add_Eventos( nome: "Final dos 100m", inicio: "2024-07-24 10:00", fim: "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    agenda.get_Eventos()
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 1
    assert out.strip() == "Final dos 100m: 2024-07-24 10:00 a 2024-07-24 11:00"
```

- Código da funcionalidade:

```python
def get_Eventos(self):
    formato = "%Y-%m-%d %H:%M"
    if len(self.eventos) >= 1:
        for evento in self.eventos:
            print(f"{evento['nome']}: {evento['inicio'].strftime(formato)} a {evento['fim'].strftime(formato)}")
    else:
        print("Nenhum evento agendado.")

    return
```

- Teste em Execução:

```
yanlucas@DiaNoite NeverMind % python3 -m pytest ./Test/test_mostrar.py
============================= test session starts ==============================
platform darwin -- Python 3.12.0, pytest-7.4.3, pluggy-1.3.0
rootdir: /Users/yanlucas/NeverMind
collected 3 items

Test/test_mostrar.py ...                                                 [100%]

============================== 3 passed in 0.02s ===============================
```

Ciclo 10: Teste Funcionalidade "Remover" evento existente.

- Código do teste:

```python
def test_remover_Evento_existente(agenda, capsys):
    agenda.add_Eventos( nome: "Final dos 100m", inicio: "2024-07-24 10:00", fim: "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 1
    agenda.remove_Evento("Final dos 100m")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 0
    assert out.strip() == "Evento removido com sucesso."
```

- Código da funcionalidade:

```python
def remove_Evento(self, nome):
    if len(self.eventos) <= 0:
        print("Não existem eventos cadastrados")
        return

    try:
        self.eventos.remove(next(evento for evento in self.eventos if evento['nome'] == nome))
        print("Evento removido com sucesso.")
    except StopIteration:
        print("Não existem evento com esse nome")
```

Ciclo 11: Teste Funcionalidade "Remover" evento sem ter eventos existentes.

- Código do teste:

```
def test_remover_Evento_sem_ter_Eventos(agenda, capsys):
    agenda.remove_Evento("Final dos 100m")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 0
    assert out.strip() == "Não existem eventos cadastrados"
```

- Código da funcionalidade:

```
def remove_Evento(self, nome):
    if len(self.eventos) <= 0:
        print("Não existem eventos cadastrados")
        return

    try:
        self.eventos.remove(next(evento for evento in self.eventos if evento['nome'] == nome))
        print("Evento removido com sucesso.")
    except StopIteration:
        print("Não existem evento com esse nome")
```

Ciclo 12: Teste Funcionalidade "Remover" evento que não existe.

- Código do teste:

```python
def test_remover_Evento_que_nao_existe(agenda, capsys):
    agenda.add_Eventos( nome: "Final dos 100m",  inicio: "2024-07-24 10:00",  fim: "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    qtd = len(agenda.eventos)
    agenda.remove_Evento("Evento")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == qtd
    assert out.strip() == "Não existem evento com esse nome"
```

- Código da funcionalidade:

```python
def remove_Evento(self, nome):
    if len(self.eventos) <= 0:
        print("Não existem eventos cadastrados")
        return

    try:
        self.eventos.remove(next(evento for evento in self.eventos if evento['nome'] == nome))
        print("Evento removido com sucesso.")
    except StopIteration:
        print("Não existem evento com esse nome")
```

- Teste em Execução:

```
yanlucas@DiaNoite NeverMind % python3 -m pytest ./Test/test_remover.py
============================= test session starts ==============================
platform darwin -- Python 3.12.0, pytest-7.4.3, pluggy-1.3.0
rootdir: /Users/yanlucas/NeverMind
collected 3 items

Test/test_remover.py ...                                                 [100%]

============================== 3 passed in 0.02s ===============================
```

Código completo:

- Agenda.py:

```python
import datetime

class Agenda:
    def __init__(self):
        self.qtd_eventos = 0
        self.eventos = []
        self.running = True
        pass

    def add_Eventos(self, nome, inicio, fim):
        inicio_dt = 0
        fim_dt = 0
        erro = 0

        try:
            inicio_dt = datetime.datetime.strptime(inicio, "%Y-%m-%d %H:%M")
            fim_dt = datetime.datetime.strptime(fim, "%Y-%m-%d %H:%M")
        except ValueError:
            print("Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'.")
            erro = 1

        if erro == 0:
            for evento in self.eventos:
                if not (fim_dt <= evento['inicio'] or inicio_dt >= evento['fim']):
                    print("Conflito de agendamento detectado.")
                    erro = 1
                    break
            if erro == 0:
                if inicio_dt >= fim_dt:
                        print("A hora de inicio deve ser anterior a hora de termino.")
                        erro = 1
                else:
                    self.eventos.append({
                    'nome': nome,
                    'inicio': inicio_dt,
                    'fim': fim_dt
                    })
                    self.qtd_eventos += 1
                    print("Evento adicionado com sucesso.")

    def get_Eventos(self):
        formato = "%Y-%m-%d %H:%M"
        if len(self.eventos) >= 1:
            for evento in self.eventos:
                print(f"{evento['nome']}: {evento['inicio'].strftime(formato)} a {evento['fim'].strftime(formato)}")
        else:
            print("Nenhum evento agendado.")

        return

    def remove_Evento(self, nome):
        if len(self.eventos) <= 0:
            print("Não existem eventos cadastrados")
            return

        try:
            self.eventos.remove(next(evento for evento in self.eventos if evento['nome'] == nome))
            print("Evento removido com sucesso.")
        except StopIteration:
            print("Não existem evento com esse nome")
            return

        return


    def sair(self):
        self.running = False
```

- Main.py:

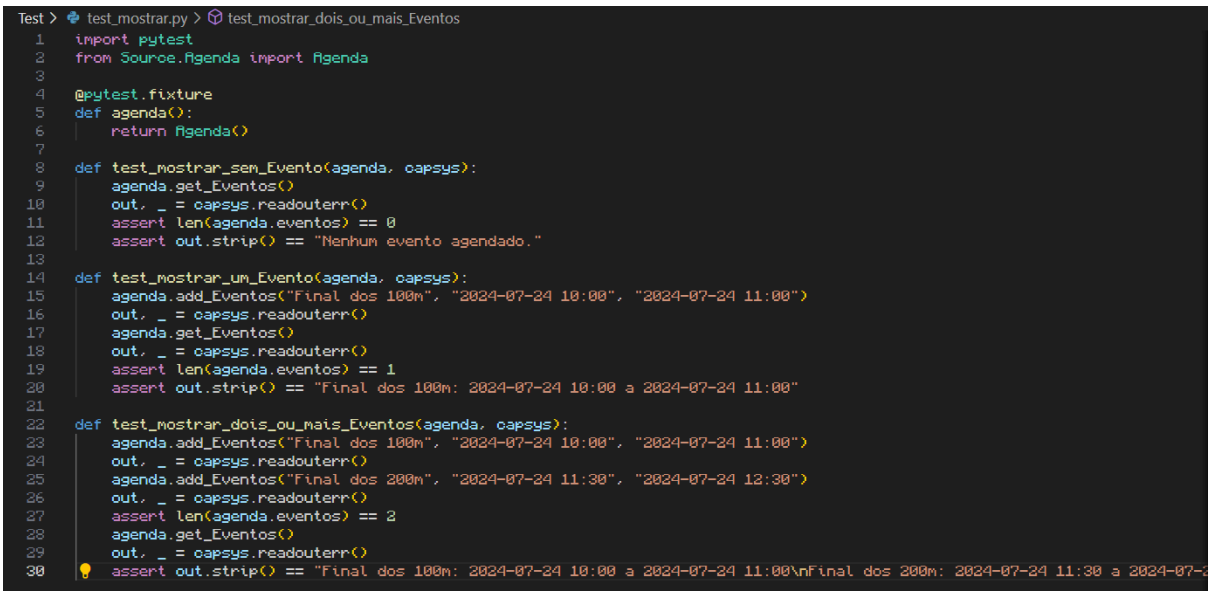
```python
from Agenda import Agenda

def exibir_menu():
    print("\nDigite a ação (adicionar, remover, mostrar, sair):")

def main():
    agenda = Agenda()
    while agenda.running:
        exibir_menu()
        escolha = input("Digite o comando: ").lower()

        if escolha == 'adicionar':
            nome = input("Digite o nome do evento: ")
            inicio = input("Digite a hora de início (YYYY-MM-DD HH:MM): ")
            fim = input("Digite a hora de término (YYYY-MM-DD HH:MM): ")
            agenda.add_Eventos(nome, inicio, fim)
        elif escolha == 'mostrar':
            agenda.get_Eventos()
        elif escolha == 'remover':
            nome = input("Digite o nome do evento para remover: ")
            agenda.remove_Evento(nome)
        elif escolha == 'sair':
            agenda.sair()
            print("Saindo...")
        else:
            print("Comando inválido! Tente novamente.")

if __name__ == "__main__":
    main()
```

- test_adicionar.py:

Test > test_adicionar.py > test_add_Eventos_hora_inicio_maior_que_fim

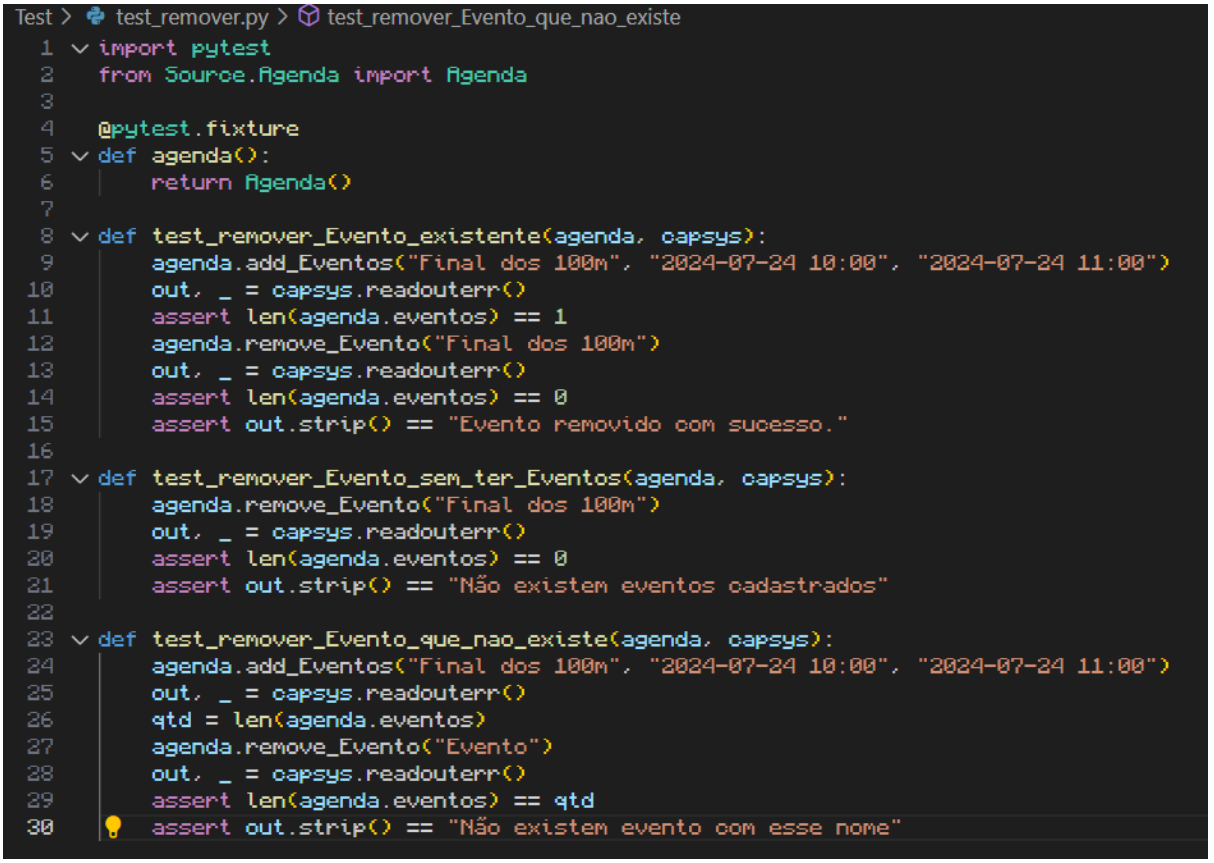
```python
import pytest
from datetime import datetime
from Source.Agenda import Agenda

@pytest.fixture
def agenda():
    return Agenda()

def test_add_Eventos_sucesso(agenda,capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert out.strip() == 'Evento adicionado com sucesso.'
    assert len(agenda.eventos) == 1
    assert agenda.eventos[0]['nome'] == "Final dos 100m"
    assert agenda.eventos[0]['inicio'] == datetime(2024, 7, 24, 10, 0)
    assert agenda.eventos[0]['fim'] == datetime(2024, 7, 24, 11, 0)

def test_add_Eventos_data_invalida(agenda,capsys):
    agenda.add_Eventos("Final dos 100m", "2024-13-35 11:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert out.strip() == "Formato de data e hora invalido. Use 'YYYY-MM-DD HH:MM'.
    assert len(agenda.eventos) == 0

def test_add_Eventos_hora_inicio_maior_que_fim(agenda,capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 12:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert out.strip() == "A hora de inicio deve ser anterior a hora de termino."
    assert len(agenda.eventos) == 0

def test_add_Eventos_conflito(agenda):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    agenda.add_Eventos("Final dos 200m", "2024-07-24 10:30", "2024-07-24 11:30")
    assert len(agenda.eventos) == 1

def test_add_Eventos_sem_conflito(agenda):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    agenda.add_Eventos("Final dos 200m", "2024-07-24 11:30", "2024-07-24 12:30")
    assert len(agenda.eventos) == 2
```

- test_mostrar.py:



```python
import pytest
from Source.Agenda import Agenda

@pytest.fixture
def agenda():
    return Agenda()

def test_mostrar_sem_Evento(agenda, capsys):
    agenda.get_Eventos()
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 0
    assert out.strip() == "Nenhum evento agendado."

def test_mostrar_um_Evento(agenda, capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    agenda.get_Eventos()
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 1
    assert out.strip() == "Final dos 100m: 2024-07-24 10:00 a 2024-07-24 11:00"

def test_mostrar_dois_ou_mais_Eventos(agenda, capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    agenda.add_Eventos("Final dos 200m", "2024-07-24 11:30", "2024-07-24 12:30")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 2
    agenda.get_Eventos()
    out, _ = capsys.readouterr()
    assert out.strip() == "Final dos 100m: 2024-07-24 10:00 a 2024-07-24 11:00\nFinal dos 200m: 2024-07-24 11:30 a 2024-07-
```

- test_remover.py:



```python
import pytest
from Source.Agenda import Agenda

@pytest.fixture
def agenda():
    return Agenda()

def test_remover_Evento_existente(agenda, capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 1
    agenda.remove_Evento("Final dos 100m")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 0
    assert out.strip() == "Evento removido com sucesso."

def test_remover_Evento_sem_ter_Eventos(agenda, capsys):
    agenda.remove_Evento("Final dos 100m")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == 0
    assert out.strip() == "Não existem eventos cadastrados"

def test_remover_Evento_que_nao_existe(agenda, capsys):
    agenda.add_Eventos("Final dos 100m", "2024-07-24 10:00", "2024-07-24 11:00")
    out, _ = capsys.readouterr()
    qtd = len(agenda.eventos)
    agenda.remove_Evento("Evento")
    out, _ = capsys.readouterr()
    assert len(agenda.eventos) == qtd
    assert out.strip() == "Não existem evento com esse nome"
```

- test_sair.py:

```python
from Source.Agenda import Agenda

def test_sair():
    agenda = Agenda()
    assert agenda.running is True
    agenda.sair()
    assert agenda.running is False
```

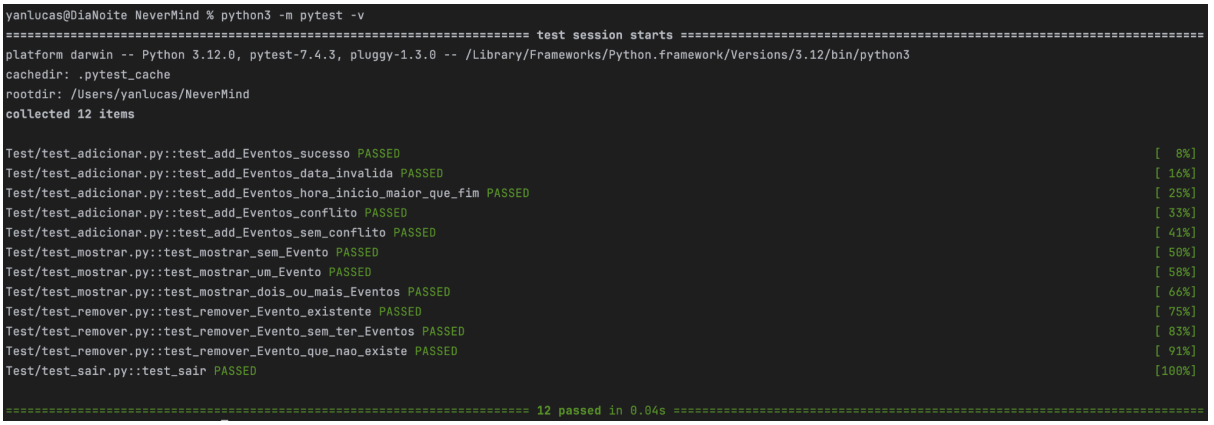
Exemplo de interação do Usuário:

```
yanlucas@DiaNoite NeverMind % python3 ./Source/Main.py

Digite a ação (adicionar, remover, mostrar, sair):
Digite o comando: adicionar
Digite o nome do evento: Meu aniversário
Digite a hora de início (YYYY-MM-DD HH:MM): 2025-03-03 18:00
Digite a hora de término (YYYY-MM-DD HH:MM): 2025-03-04 03:00
Evento adicionado com sucesso.

Digite a ação (adicionar, remover, mostrar, sair):
Digite o comando: adicionar
Digite o nome do evento: Aniversário do joao
Digite a hora de início (YYYY-MM-DD HH:MM): 2025-03-03 18:30
Digite a hora de término (YYYY-MM-DD HH:MM): 2025-03-03 20:00
Conflito de agendamento detectado.

Digite a ação (adicionar, remover, mostrar, sair):
Digite o comando: mostrar
Meu aniversário: 2025-03-03 18:00 a 2025-03-04 03:00

Digite a ação (adicionar, remover, mostrar, sair):
Digite o comando: remover
Digite o nome do evento para remover: Meu aniversário
Evento removido com sucesso.

Digite a ação (adicionar, remover, mostrar, sair):
Digite o comando: mostrar
Nenhum evento agendado.

Digite a ação (adicionar, remover, mostrar, sair):
Digite o comando: sair
Saindo...
yanlucas@DiaNoite NeverMind %
```

Suíte de testes completa:

```
yanlucas@DiaNoite NeverMind % python3 -m pytest
============================= test session starts ==============================
platform darwin -- Python 3.12.0, pytest-7.4.3, pluggy-1.3.0
rootdir: /Users/yanlucas/NeverMind
collected 12 items

Test/test_adicionar.py .....                                             [ 41%]
Test/test_mostrar.py ...                                                 [ 66%]
Test/test_remover.py ...                                                 [ 91%]
Test/test_sair.py .                                                      [100%]

============================== 12 passed in 0.05s ==============================
```

Print da IDE com todos os testes da suíte executados.



```
yanlucas@DiaNoite NeverMind % python3 -m pytest -v
============================= test session starts ==============================
platform darwin -- Python 3.12.0, pytest-7.4.3, pluggy-1.3.0 -- /Library/Frameworks/Python.framework/Versions/3.12/bin/python3
cachedir: .pytest_cache
rootdir: /Users/yanlucas/NeverMind
collected 12 items

Test/test_adicionar.py::test_add_Eventos_sucesso PASSED                  [  8%]
Test/test_adicionar.py::test_add_Eventos_data_invalida PASSED            [ 16%]
Test/test_adicionar.py::test_add_Eventos_hora_inicio_maior_que_fim PASSED [ 25%]
Test/test_adicionar.py::test_add_Eventos_conflito PASSED                 [ 33%]
Test/test_adicionar.py::test_add_Eventos_sem_conflito PASSED             [ 41%]
Test/test_mostrar.py::test_mostrar_sem_Evento PASSED                     [ 50%]
Test/test_mostrar.py::test_mostrar_um_Evento PASSED                      [ 58%]
Test/test_mostrar.py::test_mostrar_dois_ou_mais_Eventos PASSED           [ 66%]
Test/test_remover.py::test_remover_Evento_existente PASSED               [ 75%]
Test/test_remover.py::test_remover_Evento_sem_ter_Eventos PASSED         [ 83%]
Test/test_remover.py::test_remover_Evento_que_nao_existe PASSED          [ 91%]
Test/test_sair.py::test_sair PASSED                                      [100%]

============================== 12 passed in 0.04s ==============================
```