

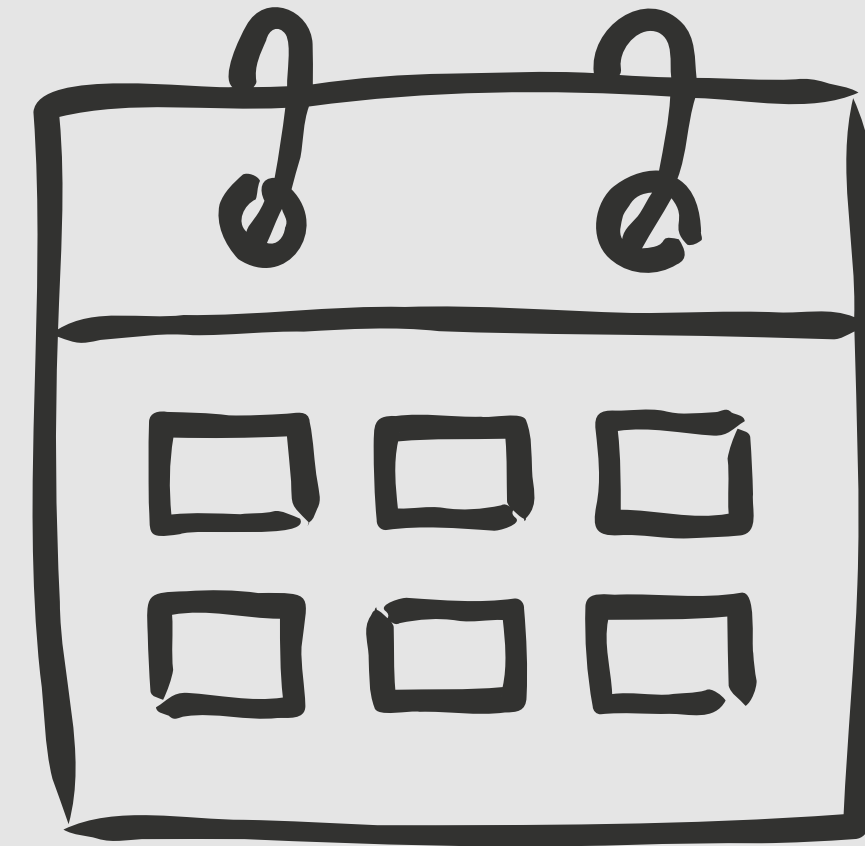
Gestão de Calendário Pessoal

Martim Amador Vasco 28694

Programação Orientada Objetos



Objetivo do Projeto



1

Desenvolver um sistema de gestão de calendário pessoal que permita gerir utilizadores, eventos e categorias, com validações e persistência de dados.

Tecnologias Utilizadas



Python



Programação Orientada a Objetos



UML (PlantUML)



JSON (persistência)



VS Code

Diagrams

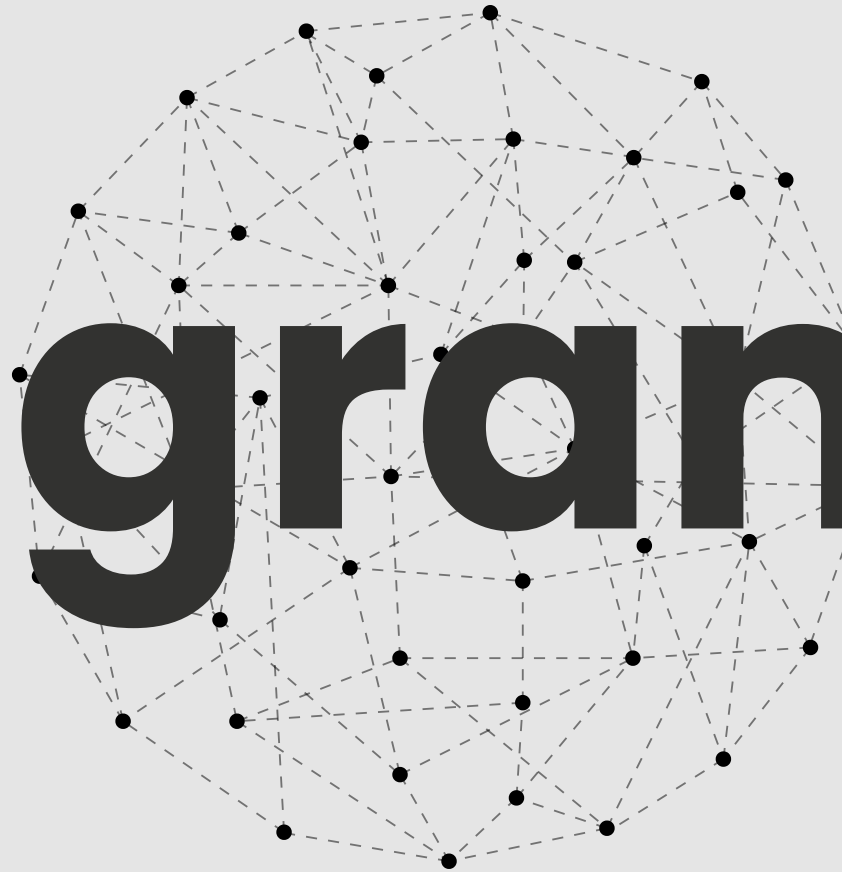


Diagrama de Classes

1

O sistema é composto pelas classes Event, Category, User e CalendarManager, responsáveis pela lógica e gestão dos eventos.

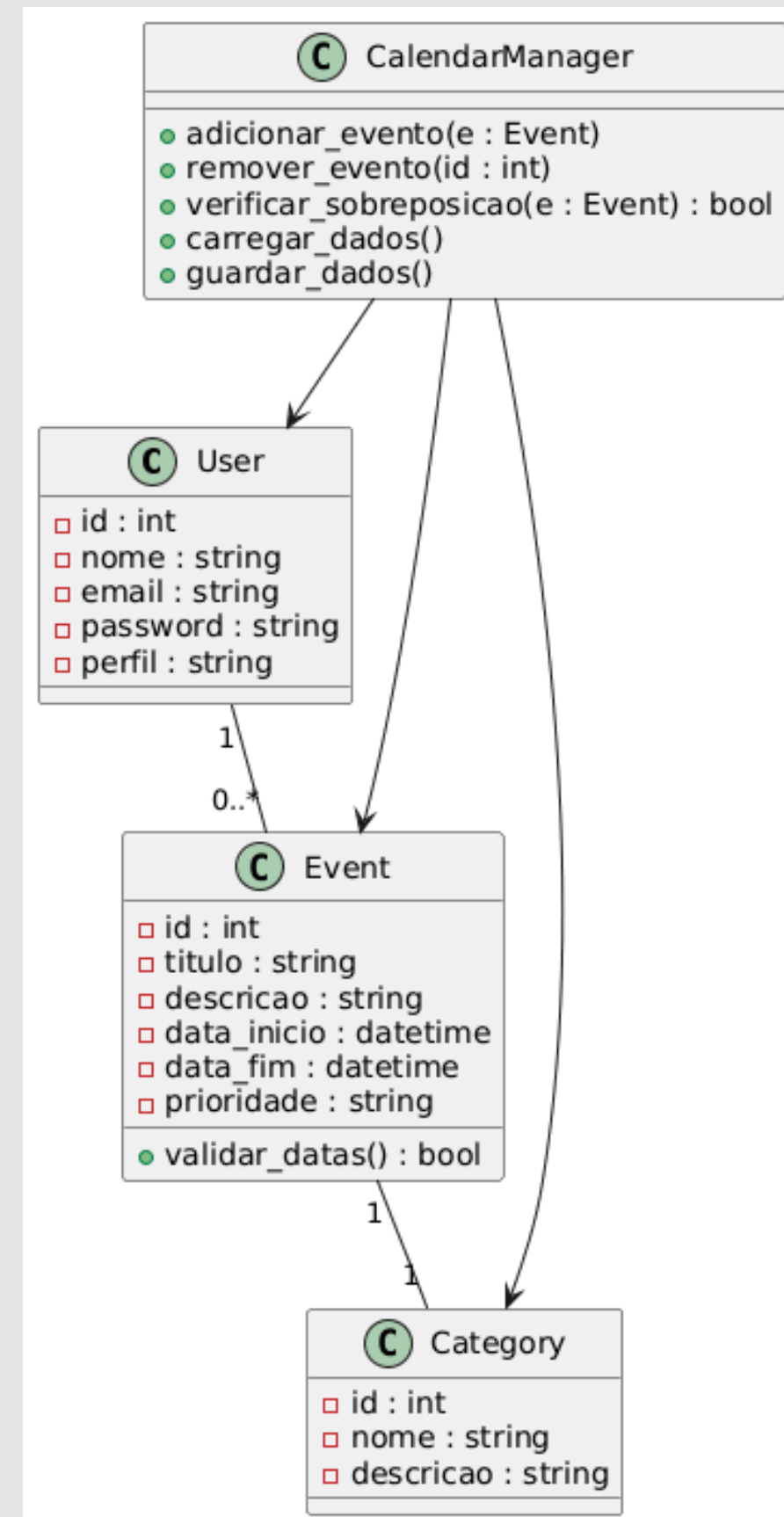


Diagrama de Casos de Uso

1

O utilizador pode adicionar, remover e visualizar eventos no sistema

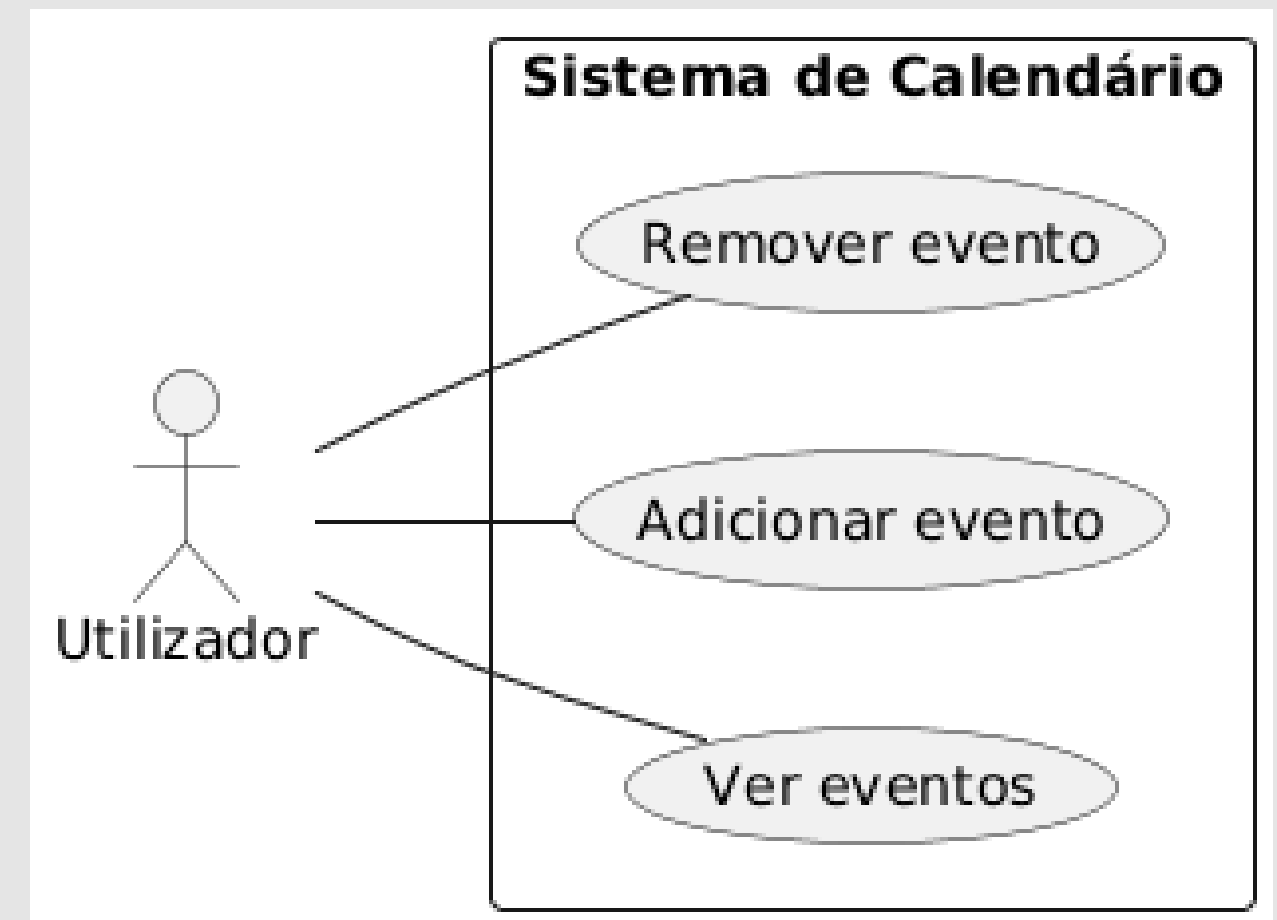
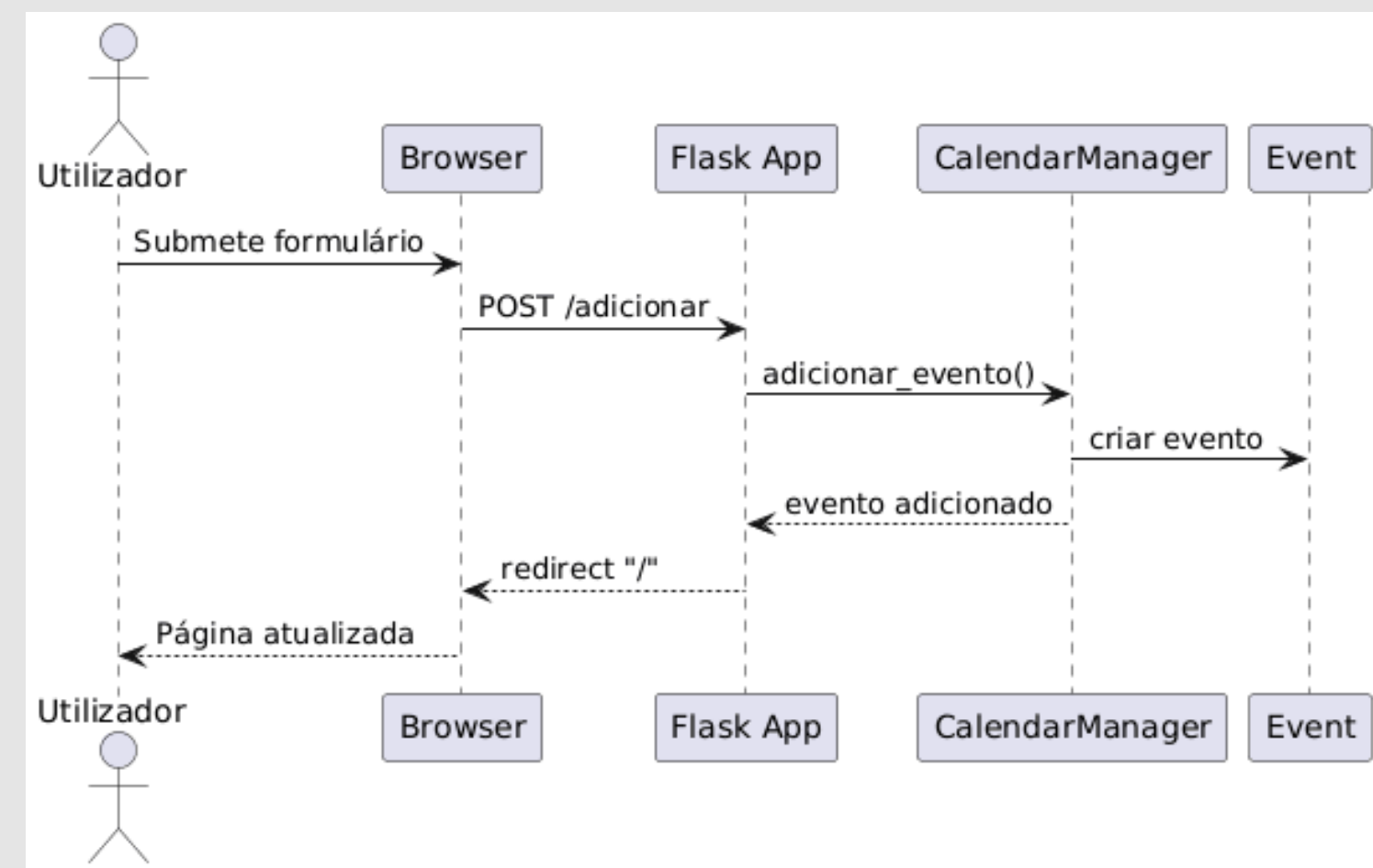
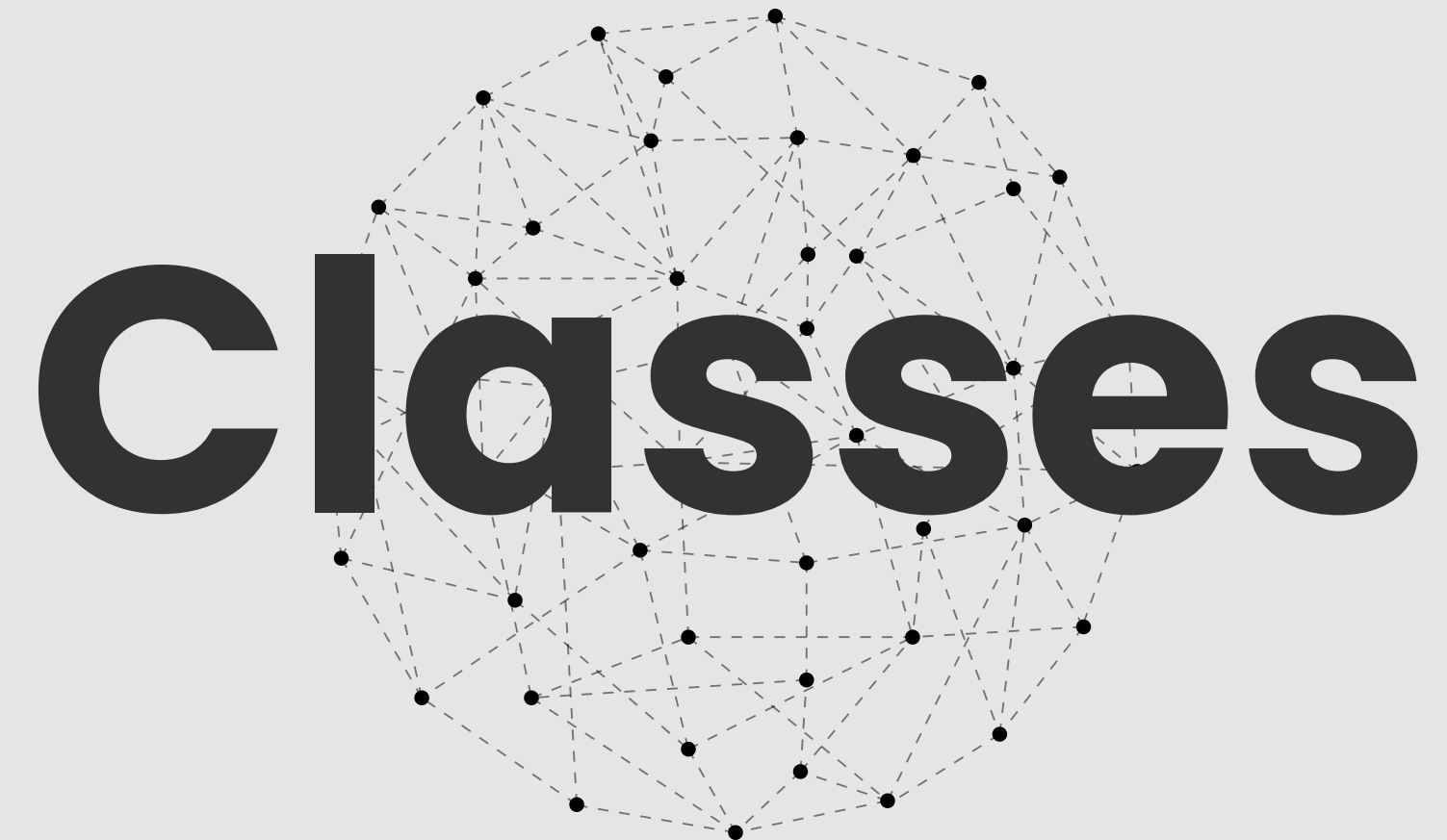


Diagrama de Sequência

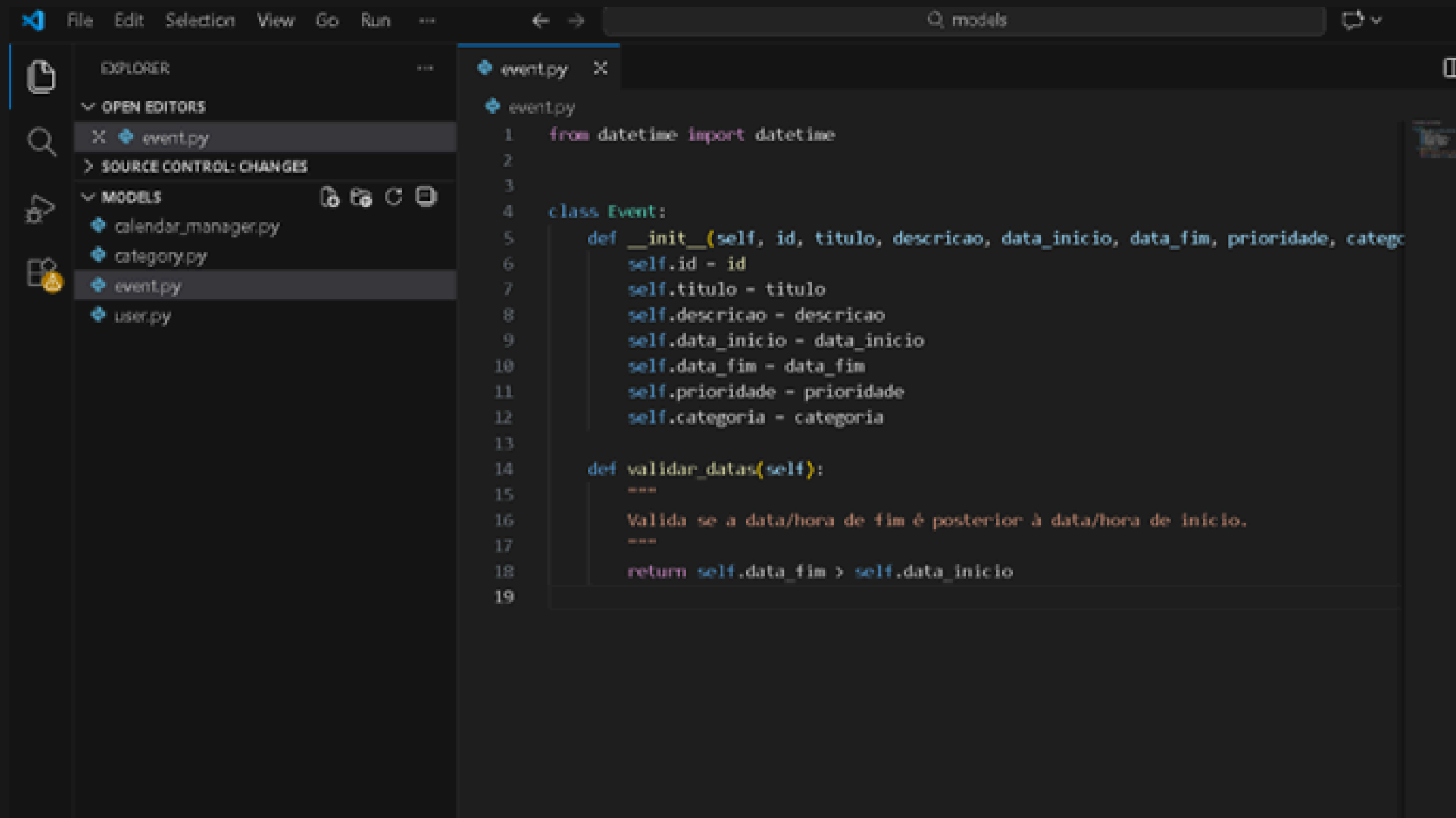
1

O diagrama representa o fluxo de criação de um evento desde a interface até ao gestor do calendário





ClasseEvent

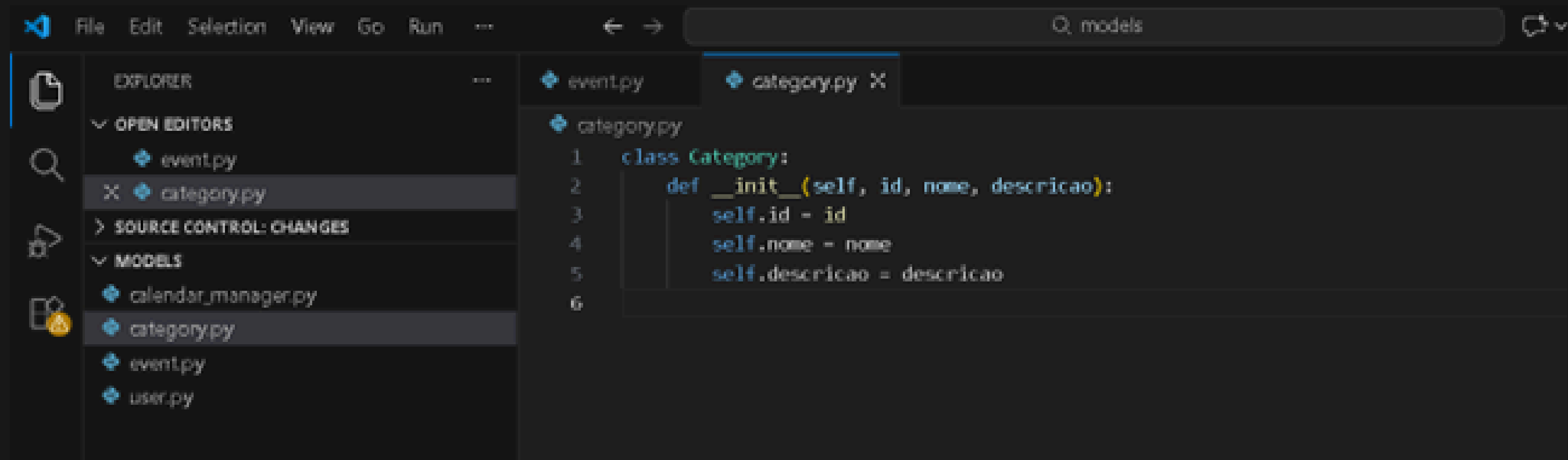


The screenshot shows a code editor with a dark theme. The Explorer sidebar on the left shows a project structure with files: calendar_manager.py, category.py, event.py (selected), and user.py. The main editor window displays the content of event.py. The code defines a class Event with an __init__ method and a validar_datas method. The __init__ method initializes attributes: id, titulo, descricao, data_inicio, data_fim, prioridade, and categoria. The validar_datas method checks if the end date is after the start date.

```
1 from datetime import datetime
2
3
4 class Event:
5     def __init__(self, id, titulo, descricao, data_inicio, data_fim, prioridade, categoria):
6         self.id = id
7         self.titulo = titulo
8         self.descricao = descricao
9         self.data_inicio = data_inicio
10        self.data_fim = data_fim
11        self.prioridade = prioridade
12        self.categoria = categoria
13
14    def validar_datas(self):
15        """
16        Valida se a data/hora de fim é posterior à data/hora de início.
17        """
18        return self.data_fim > self.data_inicio
19
```

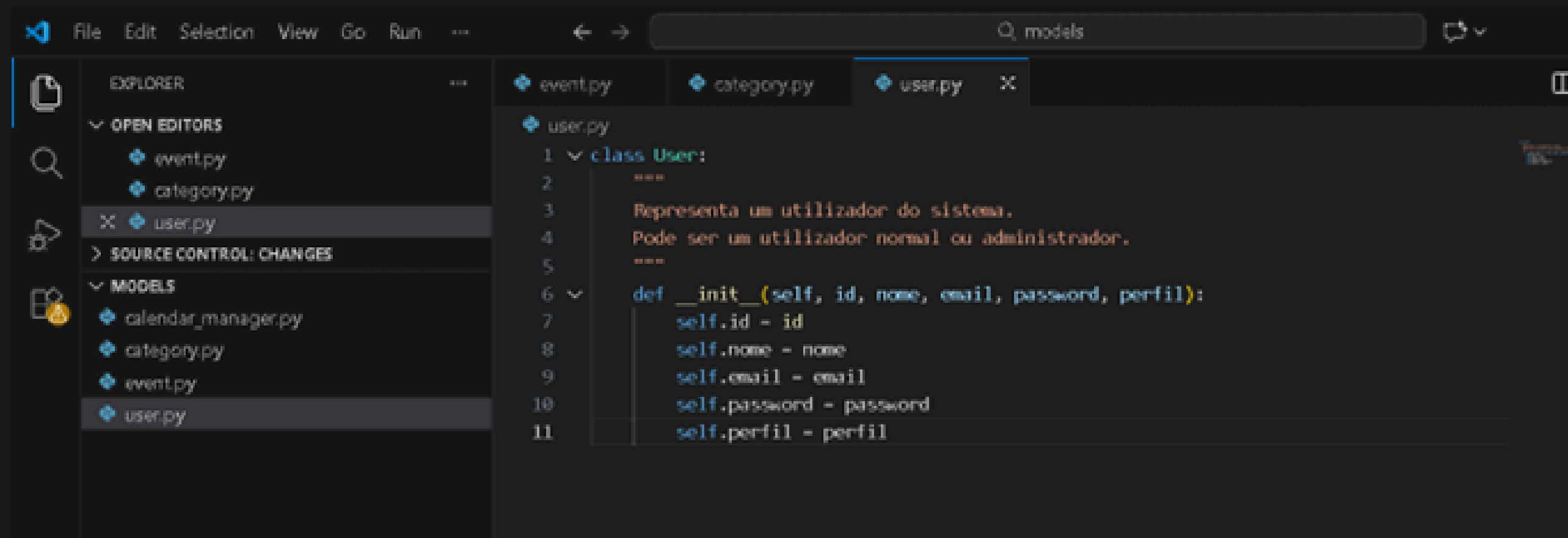
A classe Event representa um evento do calendário e valida as datas de início e fim

Classe Category



A classe Category permite classificar eventos sem conter regras de negócio próprias

Classe User



The screenshot shows a code editor with the following structure:

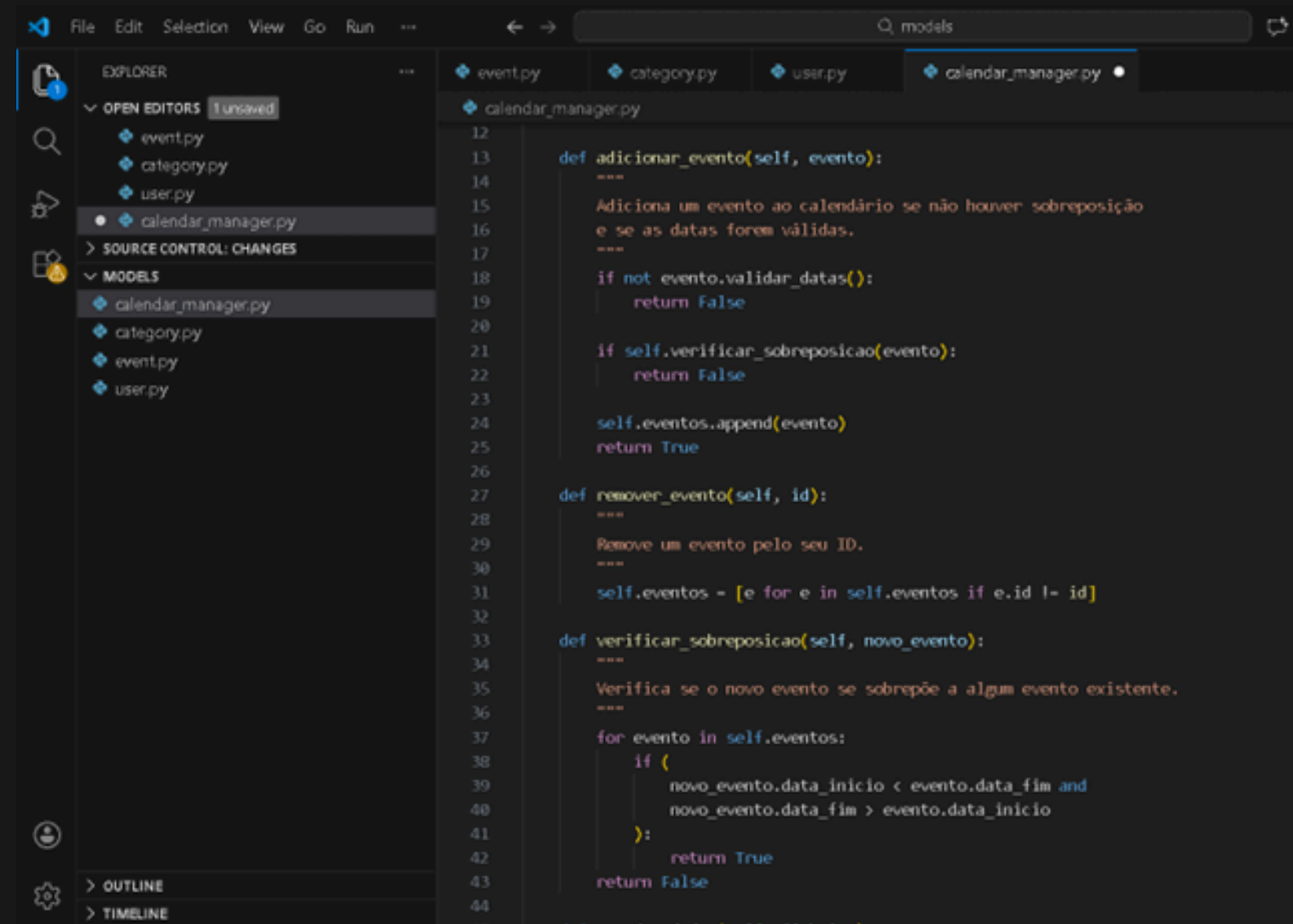
- EXPLORER**
 - OPEN EDITORS
 - event.py
 - category.py
 - user.py (selected)
 - SOURCE CONTROL: CHANGES
 - MODELS
 - calendar_manager.py
 - category.py
 - event.py
 - user.py (selected)
- EDITOR**
 - event.py
 - category.py
 - user.py (selected)

The **user.py** file contains the following Python code:

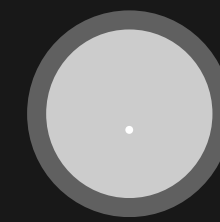
```
1 class User:
2     """
3     Representa um utilizador do sistema.
4     Pode ser um utilizador normal ou administrador.
5     """
6     def __init__(self, id, nome, email, password, perfil):
7         self.id = id
8         self.nome = nome
9         self.email = email
10        self.password = password
11        self.perfil = perfil
```

Representa os utilizadores do sistema, distinguindo perfis.

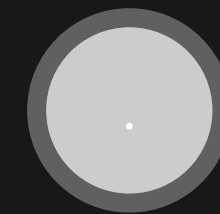
Classe CalendarManager



```
12
13 def adicionar_evento(self, evento):
14     """
15     Adiciona um evento ao calendário se não houver sobreposição
16     e se as datas forem válidas.
17     """
18     if not evento.validar_datas():
19         return False
20
21     if self.verificar_sobreposicao(evento):
22         return False
23
24     self.eventos.append(evento)
25     return True
26
27 def remover_evento(self, id):
28     """
29     Remove um evento pelo seu ID.
30     """
31     self.eventos = [e for e in self.eventos if e.id != id]
32
33 def verificar_sobreposicao(self, novo_evento):
34     """
35     Verifica se o novo evento se sobrepõe a algum evento existente.
36     """
37     for evento in self.eventos:
38         if (
39             novo_evento.data_inicio < evento.data_fim and
40             novo_evento.data_fim > evento.data_inicio
41         ):
42             return True
43     return False
44
```



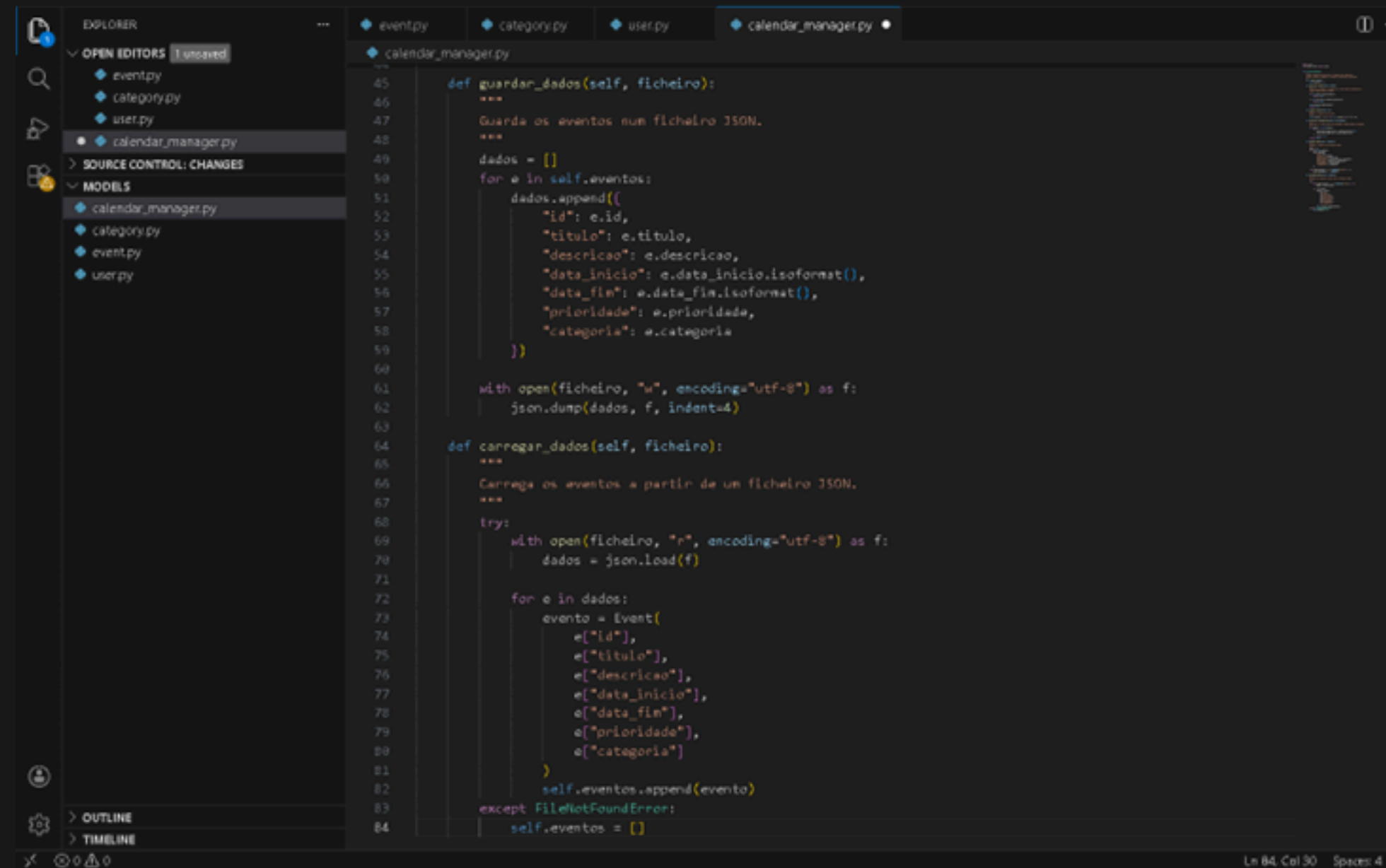
adicionar evento



verificar sobreposição

Centraliza as regras de negócio, validações e gestão de eventos.

Classe CalendarManager



```
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

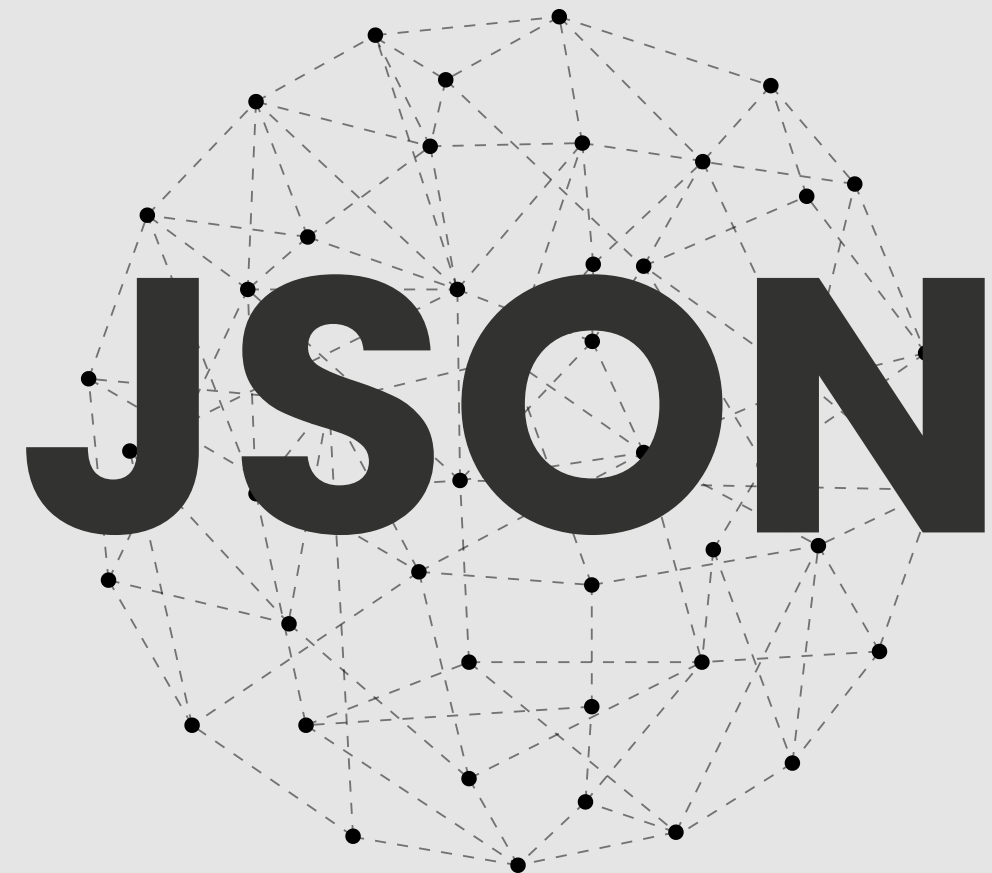
class CalendarManager:
    def guardar_dados(self, ficheiro):
        """
        Guarda os eventos num ficheiro JSON.
        """
        dados = []
        for e in self.eventos:
            dados.append({
                "id": e.id,
                "titulo": e.titulo,
                "descricao": e.descricao,
                "data_inicio": e.data_inicio.isoformat(),
                "data_fim": e.data_fim.isoformat(),
                "prioridade": e.prioridade,
                "categoria": e.categoria
            })

        with open(ficheiro, "w", encoding="utf-8") as f:
            json.dump(dados, f, indent=4)

    def carregar_dados(self, ficheiro):
        """
        Carrega os eventos a partir de um ficheiro JSON.
        """
        try:
            with open(ficheiro, "r", encoding="utf-8") as f:
                dados = json.load(f)

            for e in dados:
                evento = Event(
                    e["id"],
                    e["titulo"],
                    e["descricao"],
                    e["data_inicio"],
                    e["data_fim"],
                    e["prioridade"],
                    e["categoria"]
                )
                self.eventos.append(evento)
        except FileNotFoundError:
            self.eventos = []
```

Guardar eventos e Carrega os eventos a partir de um ficheiro JSON



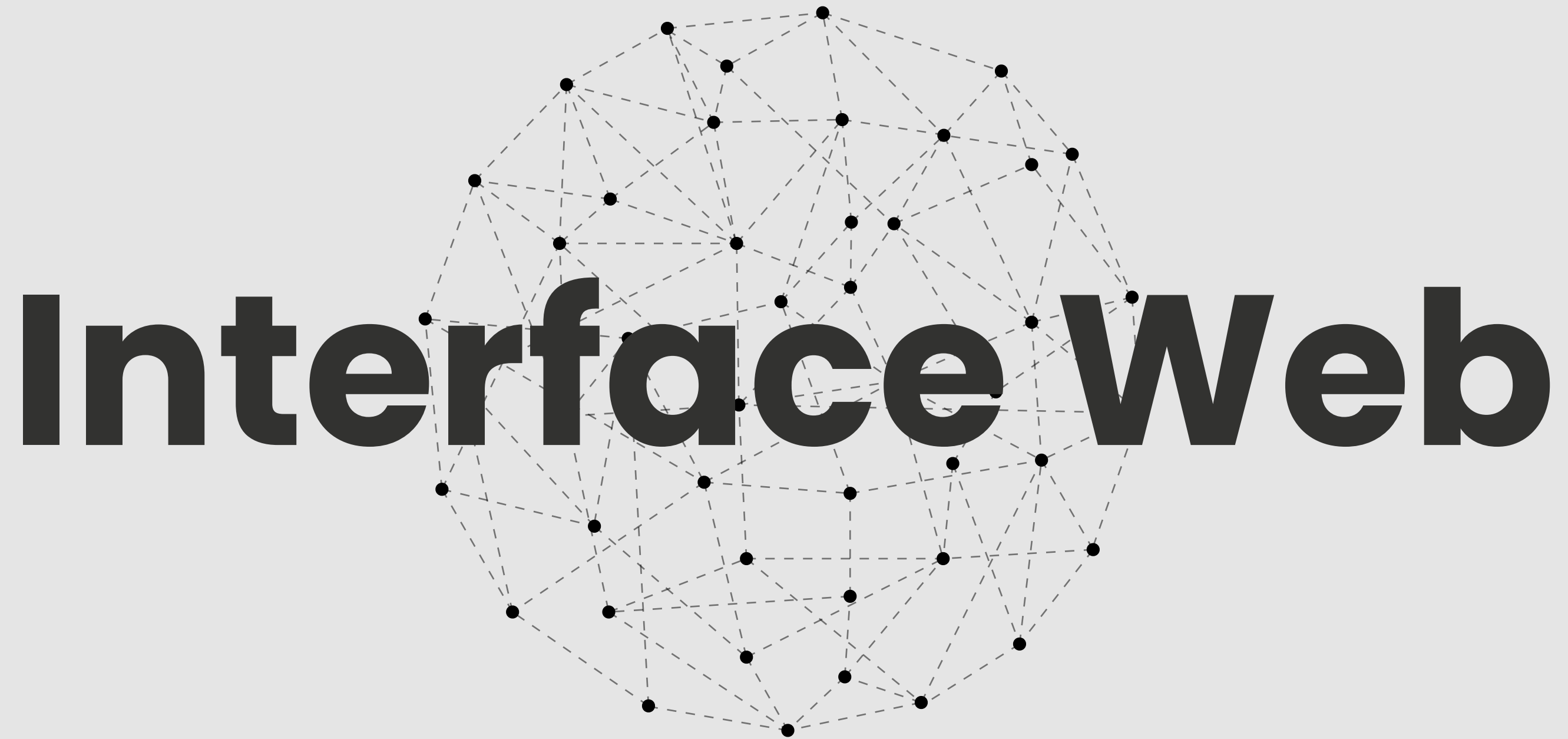
Persistência de dados (JSON)

1

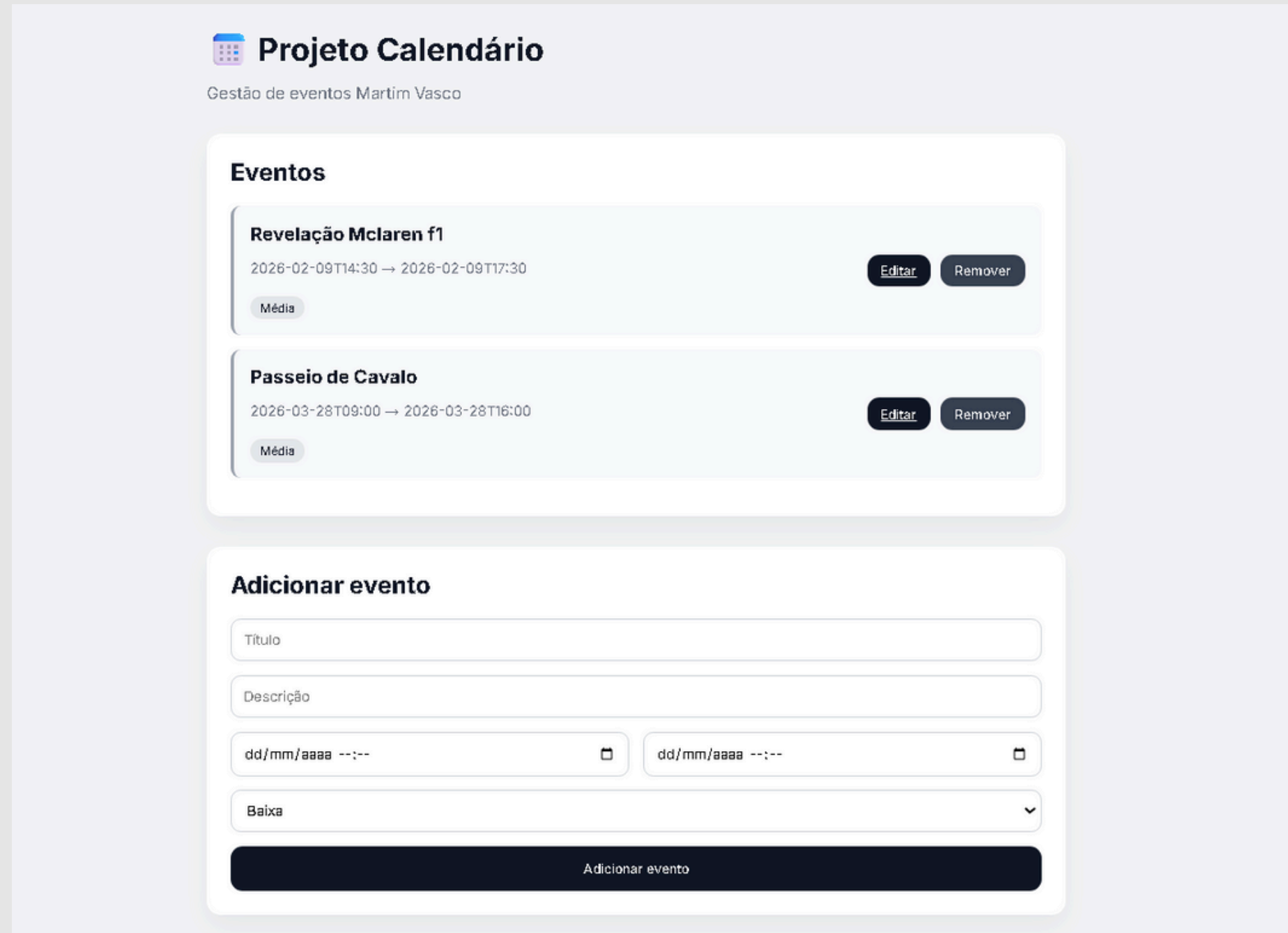
Os eventos são guardados num ficheiro JSON, permitindo persistência dos dados entre execuções da aplicação.



```
eventos.json
Esquema: <Nenhum esquema selecionado>
1  [
2    {
3      "id": 2,
4      "titulo": "Revelação McLaren f1 ",
5      "descricao": "Novo Carro",
6      "data_inicio": "2026-02-09T14:30",
7      "data_fim": "2026-02-09T17:30",
8      "prioridade": "Média"
9    },
10   {
11     "id": 3,
12     "titulo": "Passeio de Cavalo",
13     "descricao": "",
14     "data_inicio": "2026-03-28T09:00",
15     "data_fim": "2026-03-28T16:00",
16     "prioridade": "Média"
17   }
18 ]
```



Interface Web da Aplicação



The screenshot displays a web application titled "Projeto Calendário" with the subtitle "Gestão de eventos Martin Vasco". It features two main sections: "Eventos" and "Adicionar evento".

Eventos

- Revelação McLaren f1**
2026-02-09T14:30 → 2026-02-09T17:30
Média (button)
Editar (button) Remove (button)
- Passeio de Cavalo**
2026-03-28T09:00 → 2026-03-28T16:00
Média (button)
Editar (button) Remove (button)

Adicionar evento

Form fields for adding a new event:

- Título (text input)
- Descrição (text input)
- Start date/time: dd/mm/aaaa --:-- (calendar icon)
- End date/time: dd/mm/aaaa --:-- (calendar icon)
- Category: Baixa (dropdown menu)
- Adicionar evento (submit button)

A aplicação disponibiliza uma interface web desenvolvida com Flask, permitindo adicionar, editar e remover eventos de forma intuitiva. <http://127.0.0.1:5000/>

Fluxo de Funcionamento

Adicionar evento

Festa de Anos joao

Aniversário João

23/01/2026 16:0023/01/2026 19:30

Alta

Adicionar evento

Festa de Anos joao

2026-01-23T16:00 → 2026-01-23T19:30

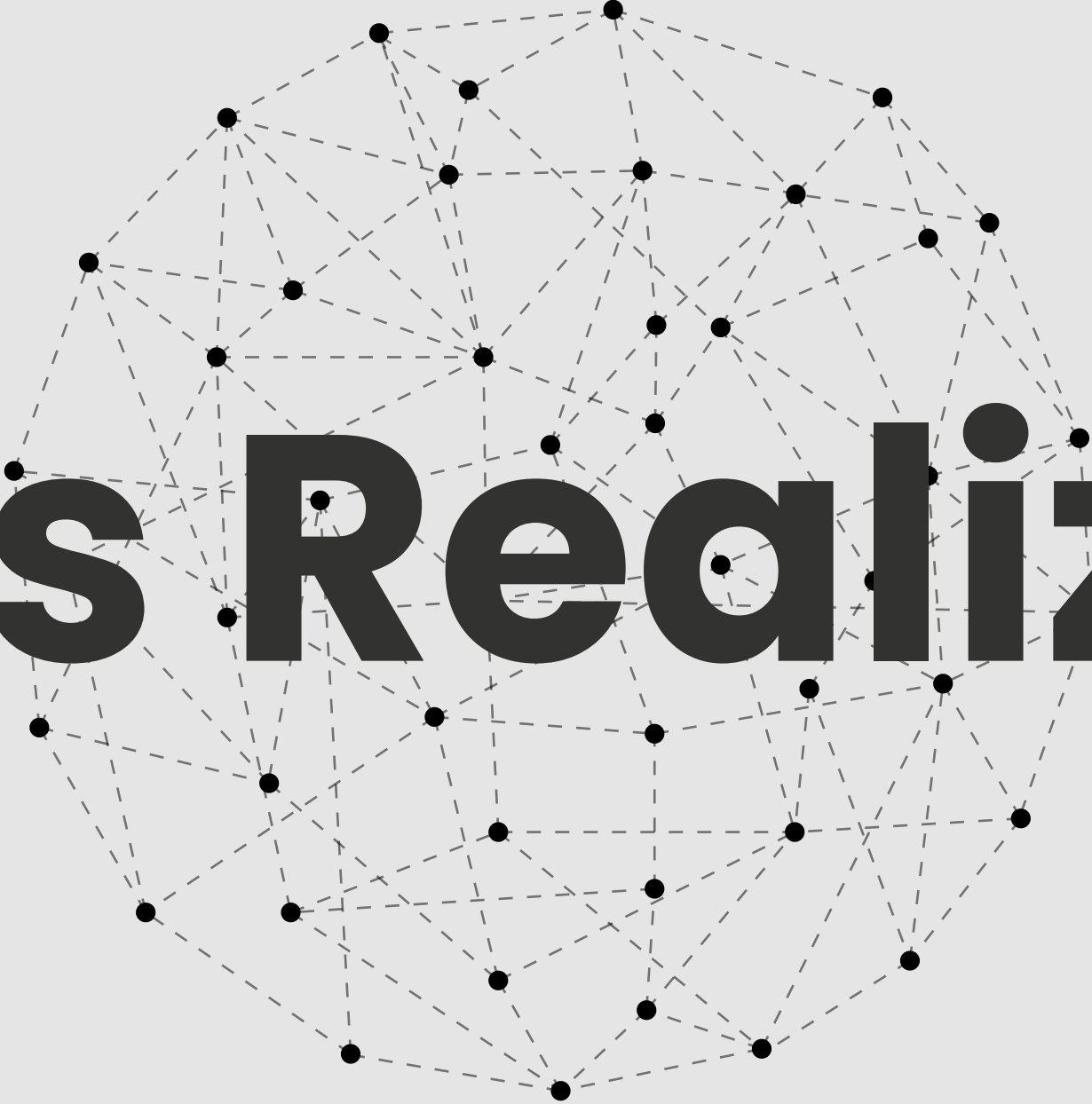
Alta

EditarRemover

```
eventos.json
Esquema: <Nenhum esquema selecionado>

1  [
2    {
3      "id": 2,
4      "titulo": "Revelação McLaren f1 ",
5      "descricao": "Novo Carro",
6      "data_inicio": "2026-02-09T14:30",
7      "data_fim": "2026-02-09T17:30",
8      "prioridade": "Média"
9    },
10   {
11     "id": 3,
12     "titulo": "Passeio de Cavalo",
13     "descricao": "",
14     "data_inicio": "2026-03-28T09:00",
15     "data_fim": "2026-03-28T16:00",
16     "prioridade": "Média"
17   },
18   {
19     "id": 4,
20     "titulo": "Festa de Anos joao",
21     "descricao": "Aniversário João",
22     "data_inicio": "2026-01-23T16:00",
23     "data_fim": "2026-01-23T19:30",
24     "prioridade": "Alta"
25   }
26 ]
```

O utilizador interage com a interface web, os dados são processados em Python e armazenados em JSON.



Testes Realizados

Testes

```
PS C:\Users\marti\OneDrive\Ambiente de Trabalho\Projeto_Calendario\E4_Projeto> python app.py
>>
True
False
❖ PS C:\Users\marti\OneDrive\Ambiente de Trabalho\Projeto_Calendario\E4_Projeto> 
```

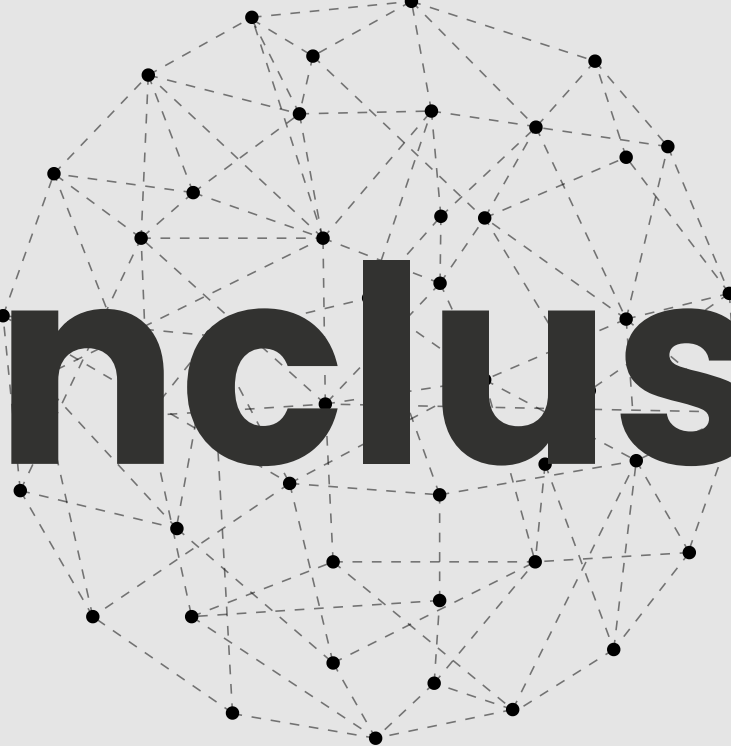
Foram realizados testes para validar a criação de eventos e detecção de sobreposição.



Limitações e melhorias futuras

Tecnologias Utilizadas

- 1 Visualização em calendário mensal
- 2 Filtros por prioridade
- 3 Filtros por prioridade



Conclusão

O projeto cumpre os requisitos propostos e demonstra a aplicação de POO, validações e persistência de dados.