Documentação do Programa de Cadastro de Alunos e Professores (OOP)

Componentes:

- Ivan Varella

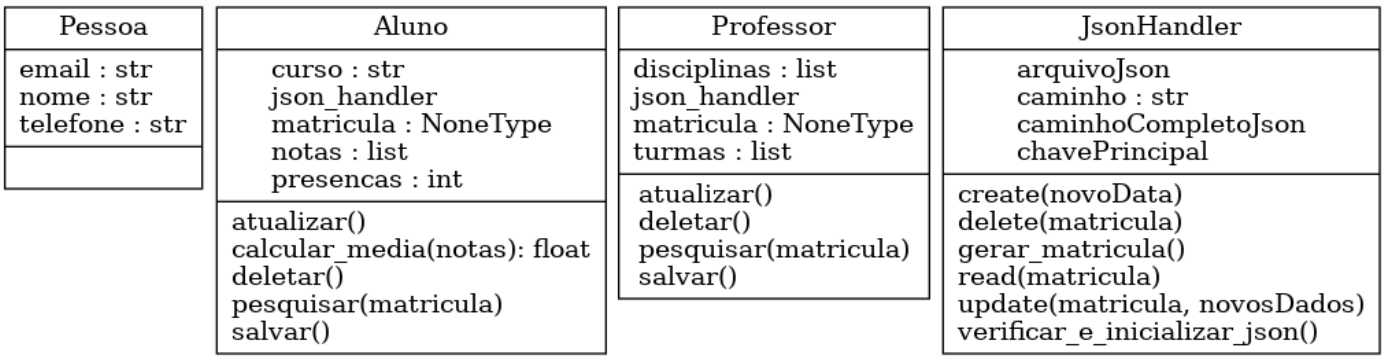
- Ricardo Nogueira

Documentação do Programa de Gerenciamento de Alunos e Professores

- Visão Geral: O programa de Gerenciamento de Alunos e Professores é uma aplicação desenvolvida em Python que permite o cadastro, listagem, pesquisa e gerenciamento de informações de alunos e professores. O programa utiliza arquivos JSON para armazenar os dados, gera arquivos HTML e PDF com as informações do arquivo JSON.
- Estrutura do Código: O programa é composto pelos seguintes arquivos:
    - Tree:

```
.
├── aluno.py
├── dados.html
├── dados.json
├── dados.pdf
├── funcoesSuporte.py
├── jsonHandler.py
├── main.py
├── pessoa.py
├── professor.py
└── rich_menus.py
```

Classes criadas:



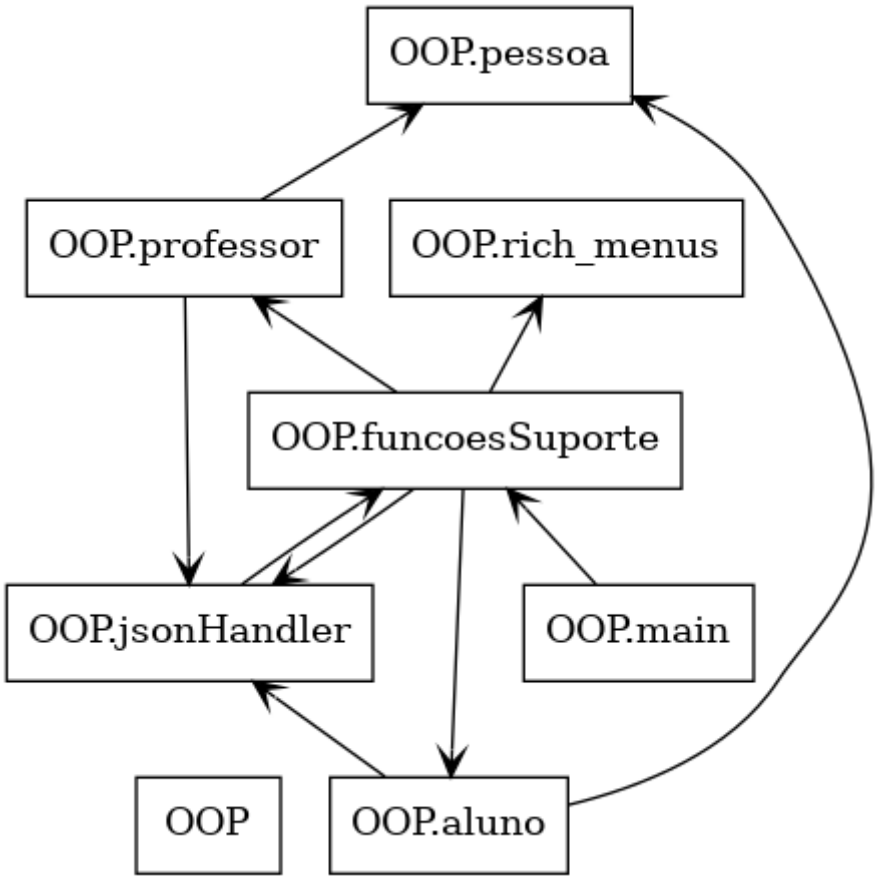
- Descrição dos arquivos:
    - main.py: Arquivo principal que controla o fluxo do programa e exibe o menu principal e realiza a chamada das funções necessárias para o funcionamento do programa.
    - funcoesSuporte.py: Arquivo que contém as funções de suporte e lógica do programa em si, como exibição de menus, cadastro, listagem e pesquisa de alunos e professores.
    - pessoa.py: Arquivo que define a classe Pessoa, que é a classe base para Aluno e Professor.
    - aluno.py: Arquivo que define a classe Aluno, que herda da classe Pessoa.
    - professor.py: Arquivo que define a classe Professor, que herda da classe Pessoa.
    - jsonHandler.py: Arquivo que contém os métodos para leitura, gravação, atualização e exclusão de dados em arquivos JSON. Essa classe é instanciada dentro das classes Alunos e Professores, sendo passadas os atributos necessários para cada classe funcionar.
    - dados.json: Arquivo JSON que armazena os dados de alunos e professores. Onde os dados são identificados pelas chaves principais (“Alunos” e “Professores”) e seus respectivos dados armazenados em uma lista de dicionários respectivamente.
        - Exemplo:

```
{
  "Alunos": [
    {
      "matricula": 1,
      "nome": "João da Silva",
      "curso": "Ciência da Computação",
      "notas": [8.5, 7.0, 9.0],
      "presencas": 18,
      "telefone": "1234-5678",
      "email": "joao.silva@email.com"
    }
  ],
  "Professores": [
    {
      "matricula": 10,
      "nome": "Caio Pereira",
      "disciplinas": ["Matemática", "Física", "Programação"],
      "turmas": ["Eng.Comp_2024.2", "Física_2023.1", "Matemática_2022.2"],
      "telefone": "987654321",
      "email": "caio.pereira@email.com"
    }
  ]
}
```

    - rich_menus.py: Arquivo que contém funções para exibição do menu principal utilizando a biblioteca rich. Separado dos demais menus com o objetivo de testar a biblioteca.

- Funcionamento do Programa:
    - O programa é iniciado pelo arquivo main.py, que exibe o menu principal.
    - O usuário pode selecionar as seguintes opções:
        - Cadastrar Alunos
        - Listar / Alterar / Excluir Alunos
        - Pesquisar Alunos
        - Cadastrar Professores
        - Listar / Alterar / Excluir Professores
        - Pesquisar Professores
        - Listar Todos
        - Pesquisar Todos
        - Gerar arquivo HTML com os dados do Json e abre para sua visualização
        - Gerar arquivo PDF com os dados do Json
        - Sair
        - Sobre
    - Cada opção do menu chama uma função correspondente do arquivo funcoesSuporte.py.
    - As funções de cadastro, listagem e pesquisa utilizam as classes Aluno e Professor definidas nos arquivos aluno.py e professor.py, respectivamente.
    - Os dados de alunos e professores são armazenados e lidos do arquivo dados.json usando os métodos da classe JsonHandler do arquivo jsonHandler.py.
    - As funções de geração de relatórios HTML e PDF utilizam as informações armazenadas no arquivo dados.json para criar os respectivos arquivos.

- Fluxo da lógica do programa em relação a seus arquivos e classes:



- Organização do Código:
  - O código está organizado de forma modular, com cada arquivo responsável por uma parte específica do programa.
  - As classes Pessoa, Aluno e Professor encapsulam as informações e comportamentos relacionados a cada tipo de entidade.
  - A classe JsonHandler é responsável pelo CRUD (Create, Read, Update e Delete) no arquivo Json, ele tanto é usado no arquivo funcoesSuporte.py quando é necessário acessar os dados de alguma forma, e também é usado dentro das classes Alunos e Professores, sendo instanciado de acordo com a classe que a utiliza, dessa forma é possível tratar os dados do arquivo Json para cada chave, assim quando é executado o método aluno.salvar(), instanciado previamente, a classe JsonHandler “sabe” que deve salvar os dados do aluno na chave “Alunos” dentro do Json.
  - As funções de suporte assim como toda a lógica do programa são encontradas no arquivo funcoesSuporte.py, a exibição de menus, cadastro, listagem, atualização, exclusão, pesquisa, tratamento de dados, criação de arquivos e outras funções, estão centralizados ali centralizados.

Bibliotecas Utilizadas:

Padrão do Python:

1. os
   Descrição: Fornece uma interface para interagir com o sistema operacional, permitindo realizar operações como criar, renomear, mover e excluir arquivos e diretórios.
2. copy (deepcopy): Permite criar uma cópia profunda de um objeto, incluindo todos os seus elementos aninhados.
3. platform: Fornece acesso a informações sobre a plataforma (sistema operacional) em que o programa está sendo executado. Utilizada para abrir o navegador após criação do arquivo HTML.
   No caso do Linux também é necessário (para o código utilizado neste programa) a instalação do pacote 'xdg-utils' que contém um conjunto de utilitários de integração entre ambiente de área de trabalho e várias operações de sistema.
   Para sua instalação no Linux (Distribuições baseadas em Debain):
   apt-get install xdg-utils
4. json: Permite a leitura e escrita de dados em formato JSON (JavaScript Object Notation), um formato leve e fácil de ler e escrever para troca de dados.
5. contextlib (contextmanager): Usada junto com o rich para criação de contextos, permitindo a definição de gerenciadores de contexto personalizados.
6. typing (Generator): Usada junto com o rich para criar dicas de tipo mais precisas para funções geradoras, melhorando a legibilidade e verificabilidade do código.
7. time: Fornece funções relacionadas ao tempo, como sleep para pausar a execução do programa.

Não padrão do Python:

1. reportlab: Para a geração de documentos PDF.
   Instalação: pip install reportlab
2. rich: Permite a criação de menus e interfaces de usuário com formatação avançada, como cores, estilos e layout.
   Instalação: pip install rich