Desenvolvimento de API

Grupo 15

Integrantes:

* Bruna Bispo Andreata – 2400468
* Diego da Silva Criscuolo - 2400805
* Fábio Luiz Garrote Ramaldes - 2400715
* Kaio Nogueira Mungo - 2401350
* Luiz Henrique Barros Calazans – 2400684

Data da atividade: 19/03/2025

1. Introdução

O objetivo deste projeto foi criar uma API utilizando Flask para gerenciar informações de professores, alunos e turmas. O desenvolvimento da API envolveu a implementação de endpoints CRUD para cada entidade, garantindo que os dados fossem armazenados e recuperados. Ademais, as respostas foram padronizadas no formato JSON e os testes foram realizados utilizando o Postman.

2. Descrição e Análise do Caso

A administração dessas informações acadêmicas é de suma importância para uma instituição educacional. Dessa maneira, a API foi projetada para permitir o cadastro, a edição, a remoção e a consulta de professores, turmas e alunos. O uso do Flask foi escolhido por sua facilidade de implementação, enquanto o Postman foi utilizado para validar as requisições HTTP.

3. Implementação ou Procedimento

3.1. Configuração do Ambiente

* Instalação do Flask via pip:

pip install flask 01

* Estrutura do projeto:

ProjetoFlask/

app.py

erros.py

requirements.txt

tests.py

3.2. Implementação dos Endpoints

Os seguintes endpoints foram criados e implementados:

Professores:

* GET /professores - Retorna todos os professores.
* POST /professores - Cria um novo professor.
* GET /professores/{id} - Retorna um professor pelo ID.
* PUT /professores/{id} - Atualiza informações do professor.
* DELETE /professores/{id} - Remove um professor.

Turmas:

* GET /turmas - Retorna todas as turmas.
* POST /turmas - Cria uma nova turma.
* GET /turmas/{id} - Retorna uma turma pelo ID.
* PUT /turmas/{id} - Atualiza informações da turma.
* DELETE /turmas/{id} - Remove uma turma.

Alunos:

* GET /alunos - Retorna todos os alunos.
* POST /alunos - Cria um novo aluno.
* GET /alunos/{id} - Retorna um aluno pelo ID.
* PUT /alunos/{id} - Atualiza informações do aluno.
* DELETE /alunos/{id} - Remove um aluno.

02

4. Testes e Resultados

Os testes foram realizados utilizando o Postman. Cada endpoint foi testado individualmente para validar se as respostas estão corretas conforme o esperado. Os principais resultados incluem:

* Requisições **GET** retornando corretamente os registros.
* Criação de novos registros com **POST** funcionando conforme esperado.
* Atualização e remoção de dados com **PUT** e **DELETE** funcionando conforme esperado.

5. Conclusão

O projeto foi bem-sucedido na criação de uma API funcional para gerenciar informações de professores, turmas e alunos. Utilizando os princípios do REST, garantindo uma padronização para as requisições e respostas. O uso do Postman facilitou a validação da API.

6. Impacto e Conexão com o Mundo Real

O uso dessa API pode ser usado em sistemas educacionais, ajudando a gestão de dados acadêmicos. A separação das entidades ajuda na integração com outras aplicações.

7. Desafios Futuros e Melhorias

Para aprimorar o projeto, algumas melhorias podem ser feitas:

* Implementar autorização para os endpoints.
* Utilizar um banco de dados SQL para persistência de dados.
* Implementar padrão de arquitetura MVC.

03