

Projeto de API: School System

Relatório

1

Projeto de API: School System

Integrantes do Grupo: André Luiz Anastácio Neves, Eduardo Zavati Souza, Júlia Oliveira de Miranda, Vinicius Dias

Data da Atividade: 17/03/2025

Objetivo:

Desenvolver uma API para um sistema escolar, permitindo a gestão de operações CRUD sobre as entidades dadas. O projeto seguirá a arquitetura Model-View-Controller (MVC), possuindo integração com um banco de dados relacional e implementação de boas práticas de desenvolvimento, segurança e escalabilidade.

1. Introdução:

A API (Application Programming Interface — em português, Interface de Programação de Aplicações) é uma tecnologia baseada em um conjunto de rotas e padrões de programação estabelecidos por uma aplicação, permitindo sua integração com diversos sistemas e plataformas diferentes, se tornando desta forma a base da comunicação entre sistemas em aplicações modernas.

Nos tempos atuais, a utilização de APIs encontra-se cada vez mais presente em projetos de startups, empresas voltadas para tecnologia e para gestão de trabalhos que buscam digitalizar e automatizar seus processos. A utilização de banco de dados relacionais e o uso de frameworks com Flask

garantem a persistência dos dados de uma forma organizada e estruturada, garantindo a segurança e escalabilidade das aplicações

Em um contexto de sistema escolar, o principal conceito abordado é o desenvolvimento de APIs RESTful utilizando o framework Flask em Python, com foco em práticas de organização de código, integração com banco de dados relacional e criação de sistemas escaláveis. Desta forma, a utilização de APIs permite gerenciar alunos, professores, turmas e notas de forma eficiente e organizada, centralizando e integrando os dados em um único ambiente. Ademais, a implementação de operações CRUD (Create, Read, Update, Delete — em português, Criar, Ler, Atualizar e Deletar) nas entidades torna o sistema mais dinâmico e flexível, permitindo sua atualização de forma mais prática.

2. Descrição e Análise do Caso:

O problema abordado foi relacionado à ausência de uma ferramenta eficaz para gestão e administração de dados escolares. A fim de solucionar a situação apresentada, foi desenvolvida uma aplicação com utilização de framework Flask, adequada por sua simplicidade e flexibilidade, permitindo o desenvolvimento de uma API eficiente para o gerenciamento dos dados.

O processo de programação do código foi realizado em Python, e seus testes conduzidos com o auxílio da plataforma Postman, que permitiu a validação da API, garantindo a qualidade da implementação antes de seu uso real. A criação de APIs também foi uma etapa essencial para o desenvolvimento da solução do problema, permitindo o gerenciamento de dados como alunos, turmas e professores e promovendo integração e comunicação entre os componentes do sistema.

3. Implementação ou Procedimento:

Durante a execução do projeto, foi realizada uma etapa inicial de planejamento e divisão de atividades entre os membros da equipe. Logo após, foi organizado o repositório na plataforma do GitHub, para que seja realizado o controle de versão e colaboração eficiente. Foi realizada a criação do código principal *app.py*, desenvolvido com base nos métodos previamente definidos e testados, servindo como base para a funcionalidade do sistema.

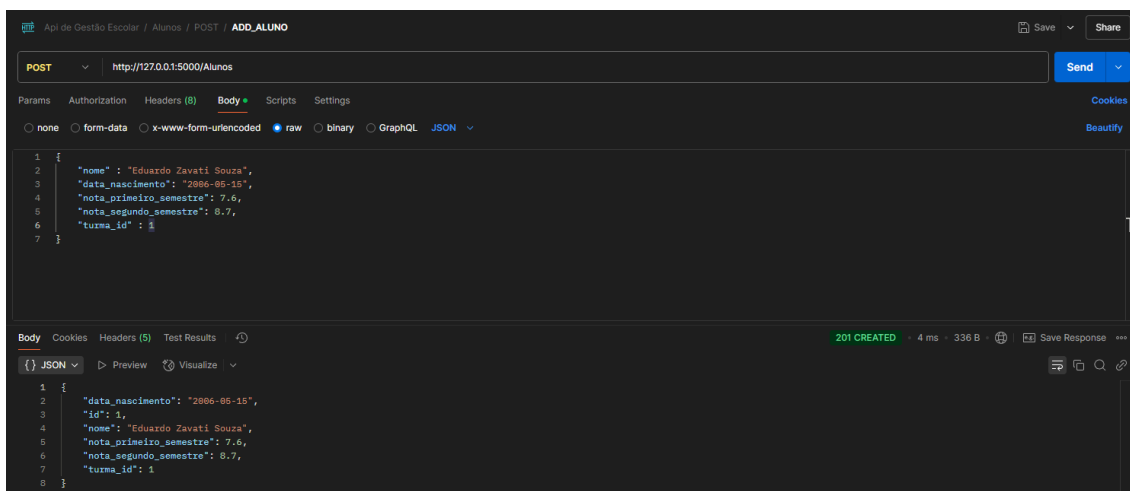
As operações CRUD foram realizadas, fundamentais para o gerenciamento de dados, além de simular um banco de dados por meio de dicionários em Python com geração de identificadores únicos.

Os testes CRUD foram implementados no Postman, assim como uma página de testes a fim de verificar a consistência e eficiência das implementações. Todos os procedimentos e etapas foram devidamente registrados e documentados em um relatório final, após os processos realizados.

Os principais desafios enfrentados incluíram a implementação dos testes CRUD no Postman, que demandou várias iterações e ajustes. A geração de chaves identificadores aleatórias também exigiu um certo nível de conhecimento e pesquisa, para garantir sua eficiência. A criação de testes unitários também se tornou uma dificuldade em nosso processo, que nos fez optar por abordagens simplificadas para clareza e qualidade no código.

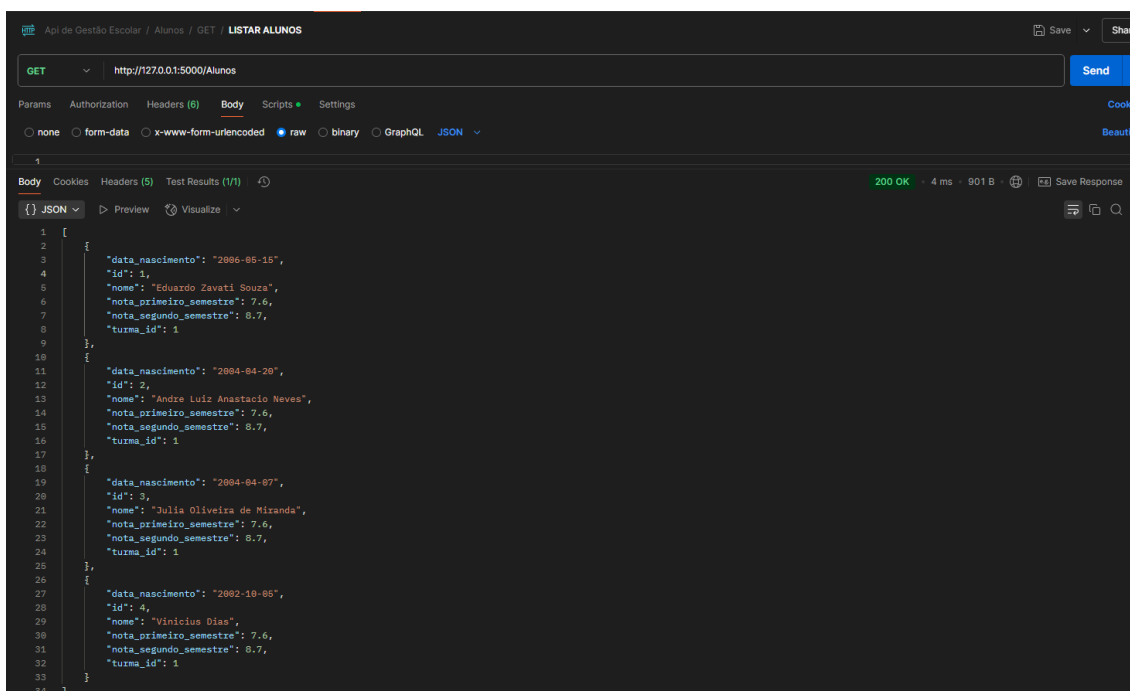
4. Resultados

POST ALUNOS



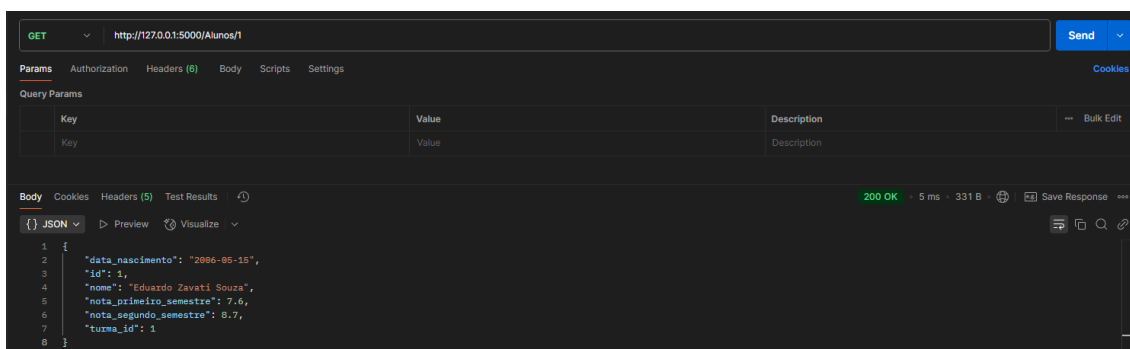
O resultado apresentado foi o que a equipe estava esperando.

GET(LISTAR) ALUNOS



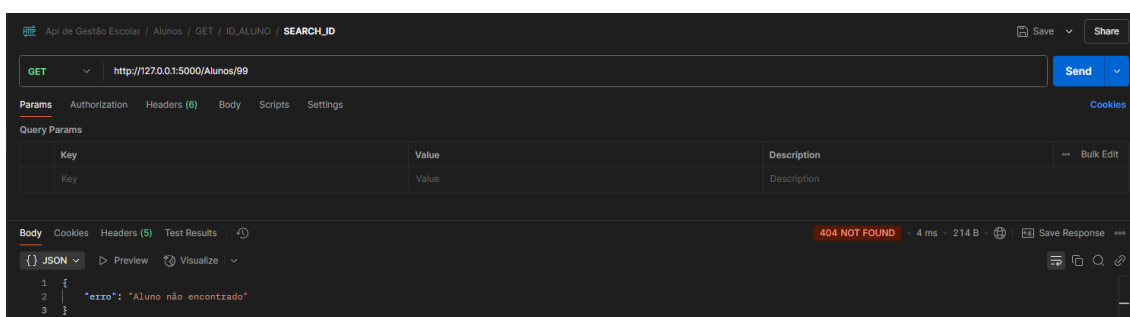
O resultado apresentado foi o que a equipe estava esperando.

GET (SEARCH BY ID) SUCESSO



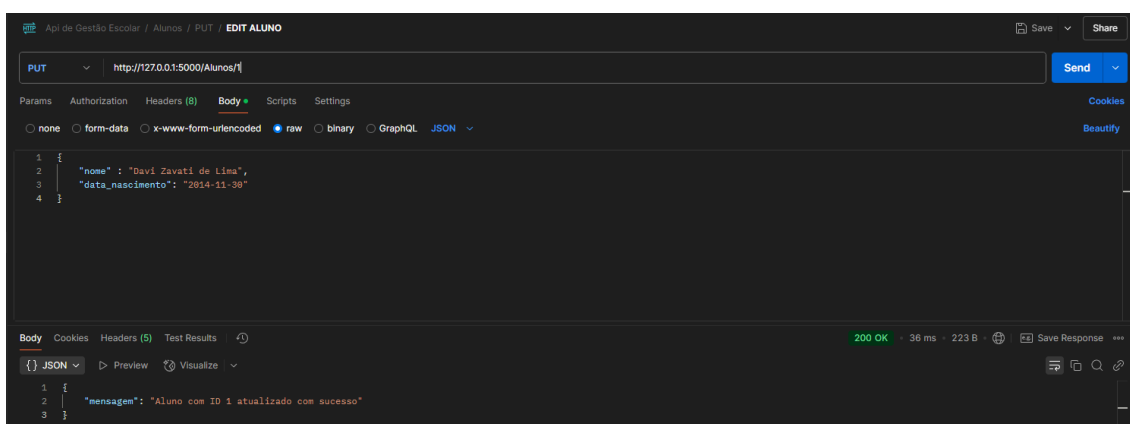
O resultado apresentado foi o que a equipe estava esperando.

GET (SEARCH BY ID) FALHA



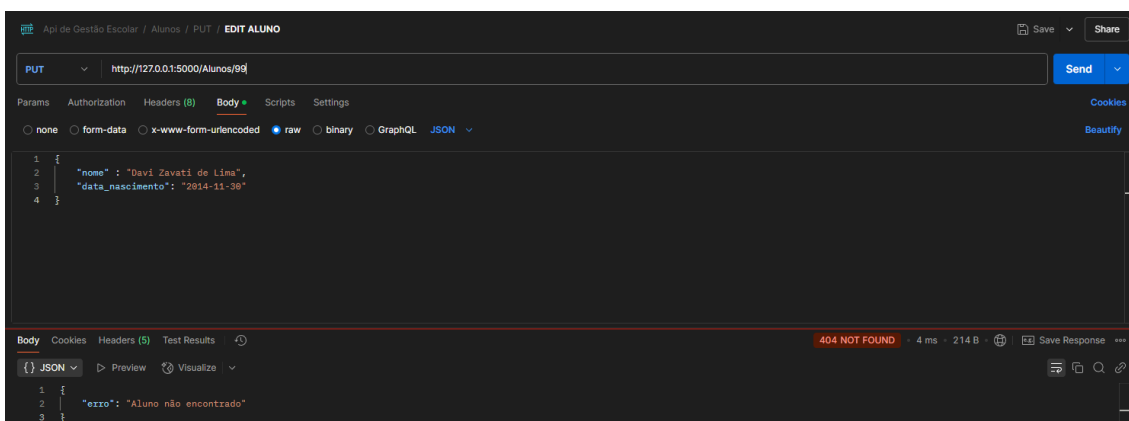
O resultado apresentado foi o que a equipe estava esperando.

PUT SUCESSO ALUNO



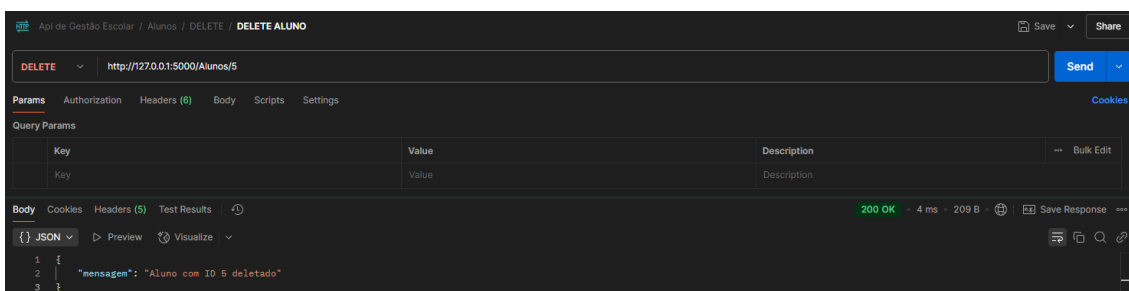
O resultado apresentado foi o que a equipe estava esperando.

PUT FALHA ALUNO



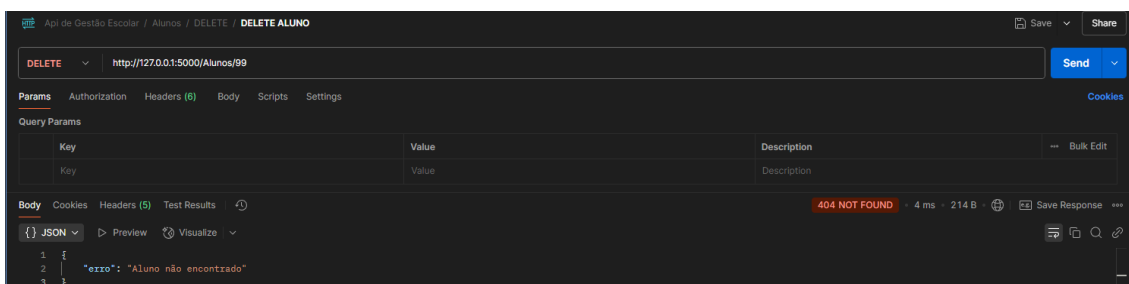
O resultado apresentado foi o que a equipe estava esperando.

DELETE SUCESSO ALUNO



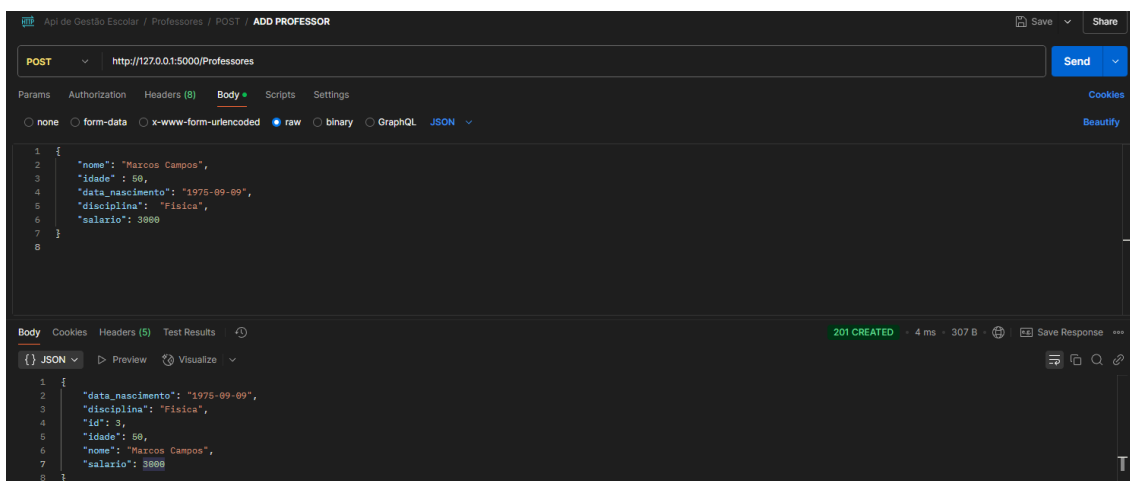
O resultado apresentado foi o que a equipe estava esperando.

DELETE FALHA ALUNO



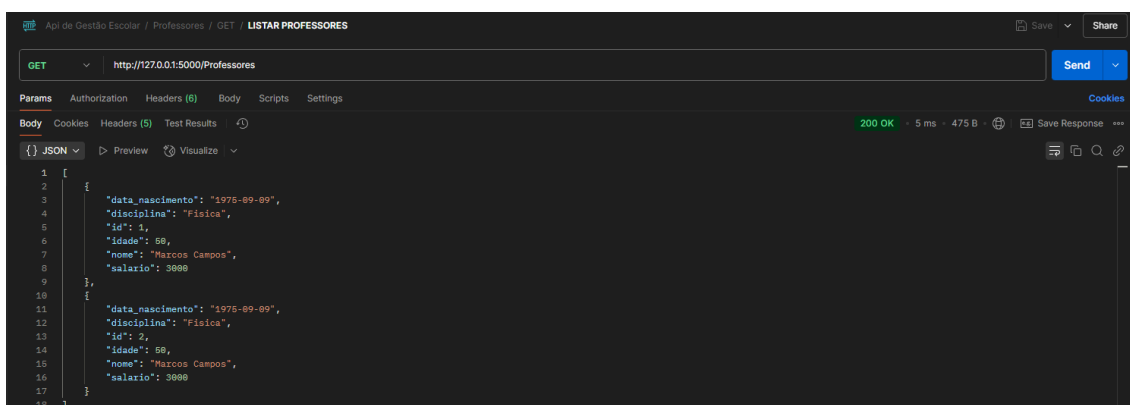
O resultado apresentado foi o que a equipe estava esperando.

POST PROFESSOR



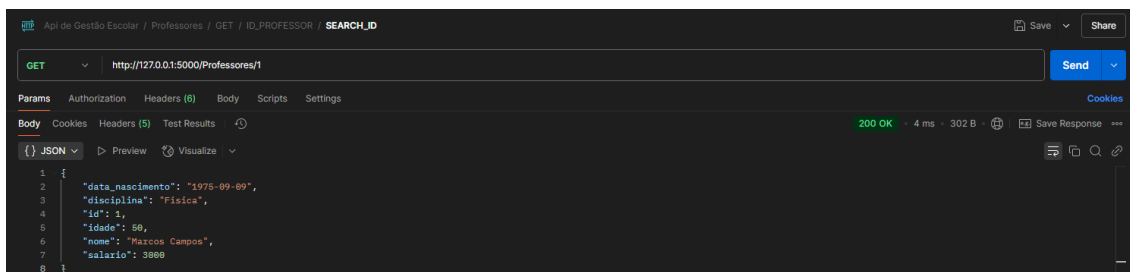
O resultado apresentado foi o que a equipe estava esperando.

GET(LISTAR) PROFESSOR



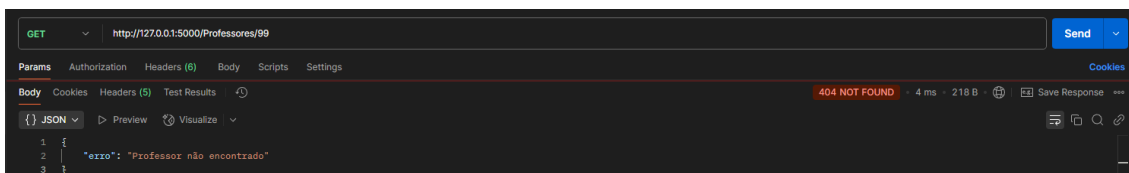
O resultado apresentado foi o que a equipe estava esperando.

GET(SEARCH BY ID) SUCESSO PROFESSOR



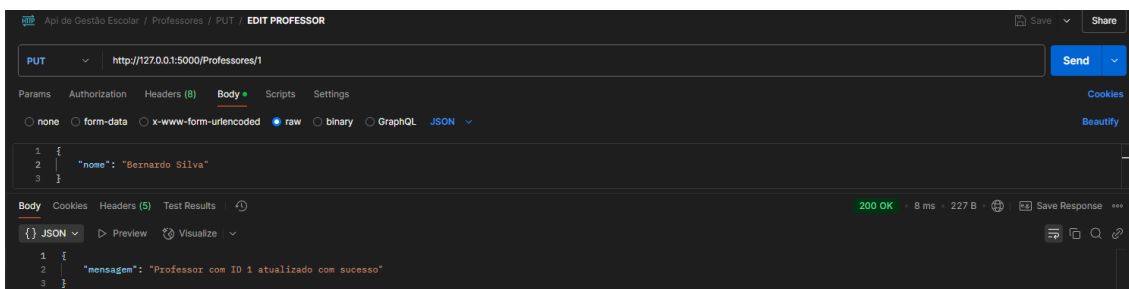
O resultado apresentado foi o que a equipe estava esperando.

GET(SEARCH BY ID) FALHA PROFESSOR



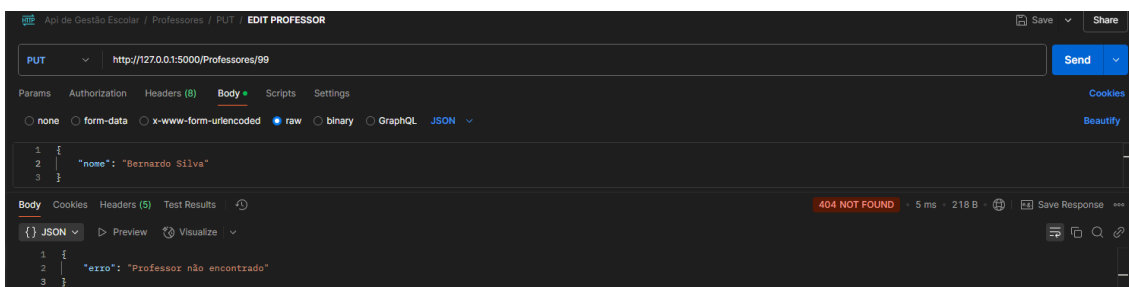
O resultado apresentado foi o que a equipe estava esperando.

PUT SUCESSO PROFESSOR



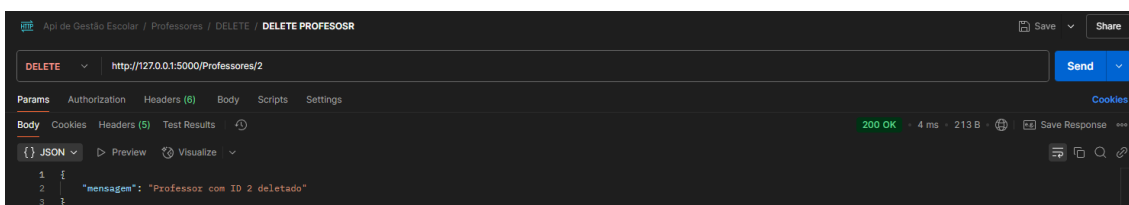
O resultado apresentado foi o que a equipe estava esperando.

PUT ERRO PROFESSOR



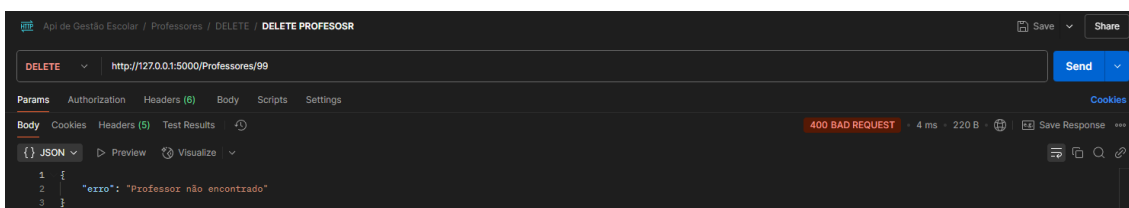
O resultado apresentado foi o que a equipe estava esperando.

DELETE SUCESSO PROFESSOR



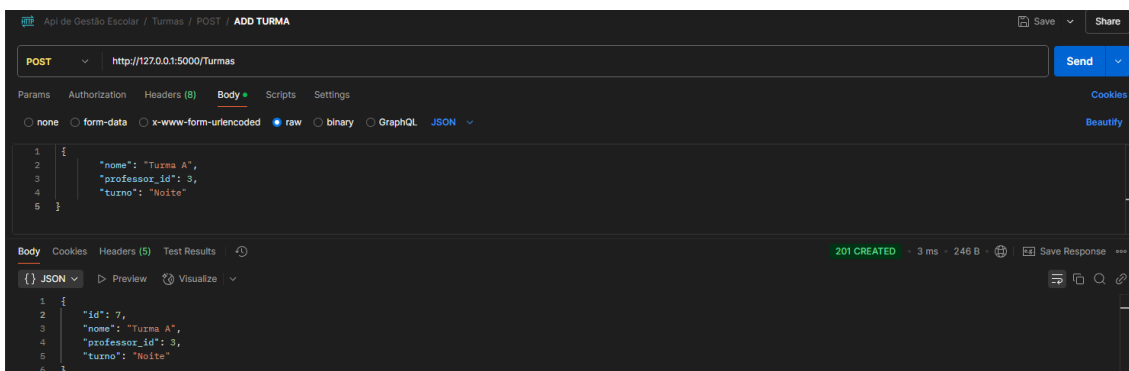
O resultado apresentado foi o que a equipe estava esperando.

DELETE FALHA PROFESSOR



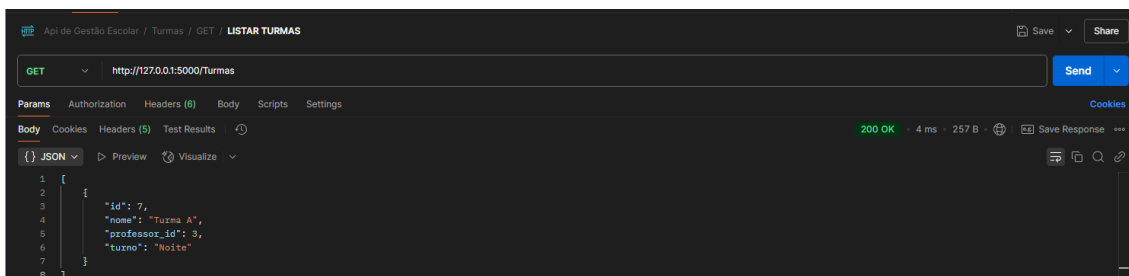
O resultado apresentado foi o que a equipe estava esperando.

POST TURMAS



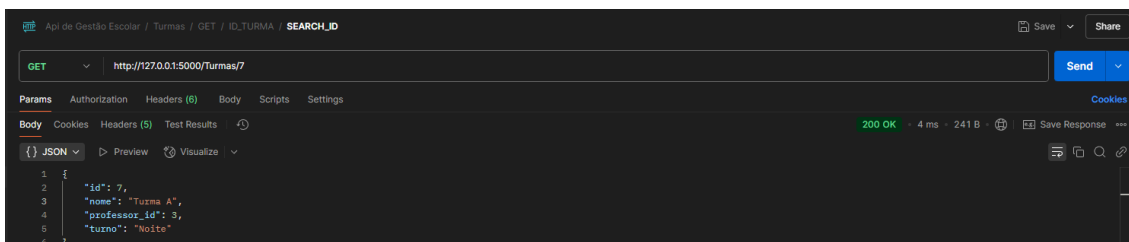
O resultado apresentado foi o que a equipe estava esperando.

GET(LISTAR) TURMAS



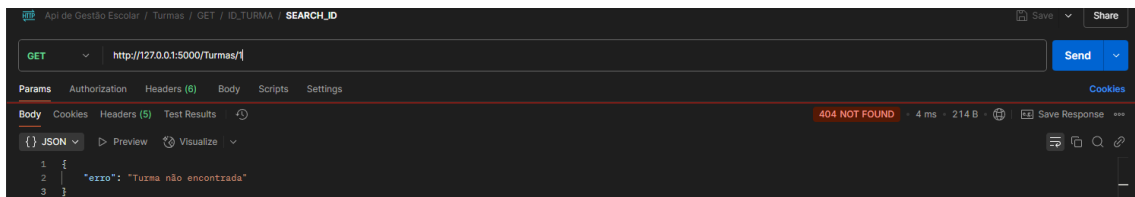
O resultado apresentado foi o que a equipe estava esperando.

GET (SEARCH BY ID) SUCESSO



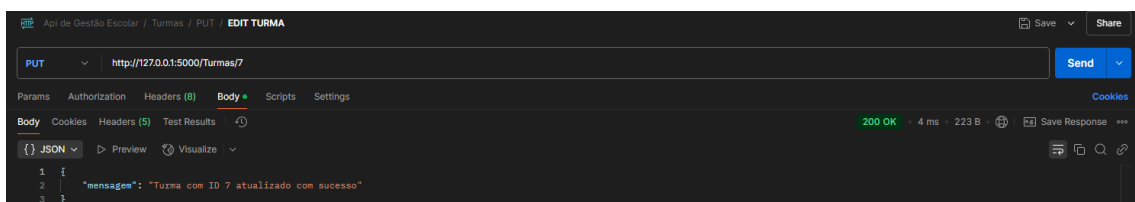
O resultado apresentado foi o que a equipe estava esperando.

GET (SEARCH BY ID) FALHA



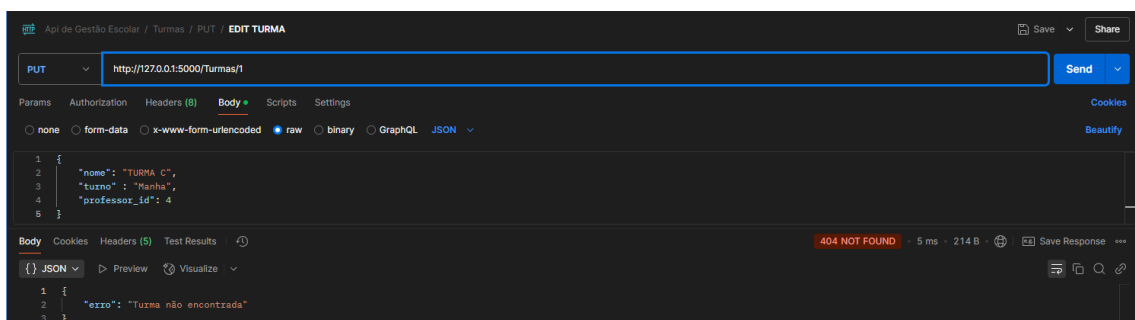
O resultado apresentado foi o que a equipe estava esperando.

PUT TURMAS



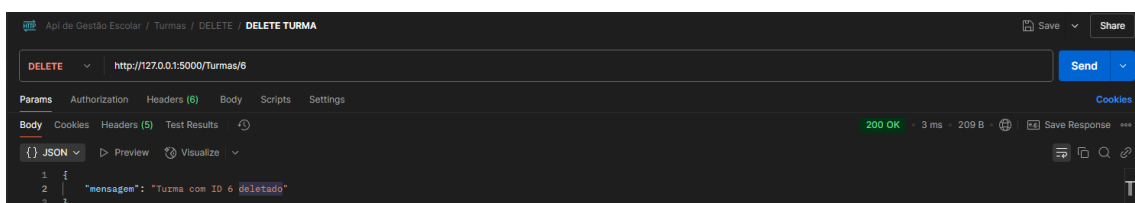
O resultado apresentado foi o que a equipe estava esperando.

PUT FALHAS TURMAS



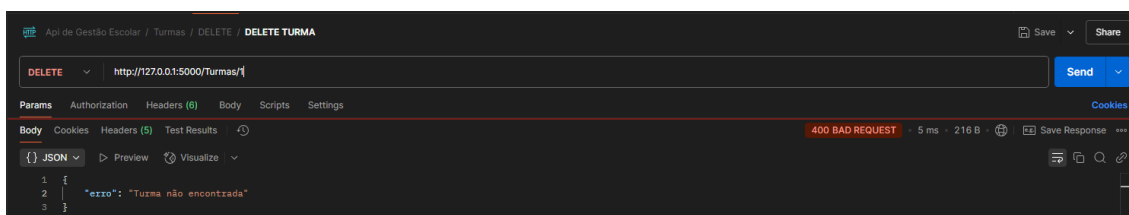
O resultado apresentado foi o que a equipe estava esperando.

DELETE SUCESSO TURMAS



O resultado apresentado foi o que a equipe estava esperando.

DELETE FALHAS TURMAS



O resultado apresentado foi o que a equipe estava esperando.

TESTE PRINCIPAL DA PÁGINA TEST.PY

```
Eduardo@DESKTOP-F1IL4F5 MINGW64 ~/ApiFlask (Develop)
$ python -m unittest test.py
.....FFF...F.FF.FF..F.FF.F.
```

O resultado apresentado possui algumas falhas em testes, no qual o objetivo da equipe e o resultado esperado era que passasse em todas as situações. Nesse teste em específico, a equipe precisa buscar melhores resultados.

5. Conclusão

O desenvolvimento do projeto proporcionou um aprendizado e compreensão mais aprofundada e prática em relação ao uso de ferramentas e técnicas para desenvolvimento de sistemas baseados em APIs, especialmente utilizando framework Flask. Além disso, a criação do projeto consolidou conhecimentos sobre métodos CRUD e o uso dessas operações para manipulação de dados em sistemas.

A aplicação de testes unitários e o uso do Postman para validação da API reforçaram a confiabilidade do código, alinhando-se com o objetivo inicial de um projeto de ferramenta de gestão escolar.

Para futuros projetos, sugere-se a inclusão de um banco de dados relacional a fim de substituir a utilização de dicionários, o que ampliaria a escalabilidade e funcionalidade do sistema. Ademais, um conhecimento mais avançado em relação a algoritmos para a gestão de dados poderia otimizar ainda mais a solução. Seria também interessante considerar a expansão da API, incluindo mais funcionalidades e trazendo uma maior aplicabilidade do projeto.

6. Impacto e Conexão com o Mundo real

O tema do projeto possui um uso prático em sistemas de automação e administração, em recursos como APIs e técnicas CRUD, que são fundamentais em áreas como educação, saúde e logística. A criação de identificadores exclusivos também é comumente utilizada para gerenciar informações em bancos de dados e plataformas digitais.

O aprendizado adquirido no desenvolvimento do projeto pode ser implementado em soluções empresariais, conectando diferentes sistemas e aprimorando procedimentos. Setores como administração escolar, gerenciamento de estoques e verificação de sistemas também podem aproveitar-se de maneira direta, favorecendo a escalabilidade e a eficiência.

7. Desafios Futuros e Melhorias

A reorganização dos testes visa aprimorar a eficiência e reduzir falhas no código. A criação de chaves estrangeiras fortalece a integridade do banco de dados, promovendo conexões estruturadas. A busca constante por melhorias reflete o compromisso com a qualidade e o aprendizado.

Propõe-se o uso de bancos de dados relacionais robustos, implementação de testes automatizados e adoção de UUIDs para identificadores únicos. O aprendizado pode ser expandido para sistemas de gestão em diferentes setores, integrando segurança, escalabilidade e novas funcionalidades.

Referências

Sites: RedHat, Alura, MOD: Mercado Online Digital, GitHub, Dev Community, YouTube