API de Gestão Educacional com Flask

“Sistema CRUD para Professores, Turmas e Alunos”

Autor: Guilherme de Freitas Fracasso
Enzo Pelakoski Cavinato
Instituição: Faculdade IMPACTA
Curso: ADS
Data: 17/03/2025

1.INTRODUÇÃO

O que estamos fazendo e por quê?
Nosso objetivo é criar uma API para gerenciar professores, turmas e alunos. Mas por que isso é importante? Sistemas educacionais precisam ser organizados e escaláveis, e a modularização ajuda a manter um código mais limpo e flexível.
Que tecnologias estamos usando e por quê?
Optamos pelo Flask para criar a API e usamos Blueprint para modularizar o projeto. Isso facilita a manutenção e expansão. No início, os dados serão armazenados em listas na memória, mas planejamos migrar para um banco de dados futuramente.

2.DESCRIÇÃO E ANÁLISE DO CASO

Qual é o problema que estamos resolvendo?
Precisamos de uma API que permita cadastrar, listar, editar e excluir professores, turmas e alunos de forma eficiente.
Quais ferramentas estamos usando?

- Flask para criar a API.
- Blueprint para organizar o projeto.
- Pytest para testar as funcionalidades.

Quais desafios encontramos? Um dos principais desafios foi relacionar professores a turmas. Resolvemos isso usando IDs para referenciar essas relações, garantindo um vínculo eficiente.

3.IMPLEMENTAÇÃO OU PROCEDIMENTO

Como estruturamos o desenvolvimento?

1. Criamos o CRUD de professores.
2. Expandimos para incluir o gerenciamento de turmas e suas relações com professores.
3. Implementamos testes unitários.
4. Começamos o planejamento para migração a um banco de dados.

Quais dificuldades enfrentamos e como resolvemos?
Ao trabalhar com listas em memória, garantir a consistência dos dados foi um desafio. Para contornar isso, implementamos métodos de validação para evitar erros na manipulação das informações.

4.RESULTADOS

O que testamos e o que descobrimos?
Criamos testes com Pytest para validar as funcionalidades da API. Todos os testes passaram, confirmando que a API funciona corretamente.
Os resultados atenderam às expectativas?
Sim, conseguimos implementar todas as funcionalidades esperadas. No entanto, ficou evidente a necessidade de migrar para um banco de dados para garantir persistência.

5.CONCLUSÃO

O que aprendemos?
Aprendemos a importância da modularização na construção de APIs e como garantir a consistência dos dados em um ambiente temporário.
O que faremos no futuro?

- Implementar um banco de dados.
- Adicionar autenticação.
- Expandir as funcionalidades da API.

6. IMPACTO E CONEXÃO COM O MUNDO REAL

Onde esse projeto pode ser aplicado?
Essa API pode ser usada em plataformas educacionais para gerenciar professores e turmas de forma eficiente.
Que questões esse projeto nos fez explorar?

- Como organizar o código para facilitar a expansão?
- Como garantir que os dados sejam persistentes e seguros?

7.DESAFIOS FUTUROS E MELHORIAS

Como podemos melhorar o projeto?

- Implementando um banco de dados para armazenamento persistente.
- Adicionando autenticação para maior segurança.
- Criando uma interface frontend para facilitar o uso da API.

Essas melhorias vão permitir que o projeto evolua e seja aplicado em contextos mais complexos no futuro.