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 objetivo deste projeto é apresentar o desenvolvimento de uma API para gerenciar professores, turmas e alunos utilizando Flask. Durante a implementação, optamos por adotar a biblioteca Blueprint do Flask para modularizar a aplicação, o que proporciona uma estrutura mais organizada e facilita a expansão do projeto à medida que novas funcionalidades são adicionadas. Além disso, decidimos inicialmente armazenar os dados em listas em memória, uma solução prática e rápida para o desenvolvimento inicial, mas que será substituída por um banco de dados no futuro para garantir a persistência dos dados.

2. DESCRIÇÃO E ANÁLISE DO CASO

O projeto busca criar uma API RESTful capaz de realizar operações básicas de CRUD (Create, Read, Update, Delete) para as entidades professores, turmas e alunos. A escolha do Flask com Blueprint visa tornar o código mais modular e fácil de manter à medida que o projeto cresce. Desde o início, foi percebido que a solução de armazenamento em listas simples seria limitada, principalmente devido à falta de persistência entre as reinicializações do servidor.
Neste capítulo, discutimos como cada parte do projeto foi implementada e os desafios encontrados, especialmente ao lidar com as operações de relacionamento entre as entidades (como vincular um aluno a uma turma e um professor a uma turma).

3. IMPLEMENTAÇÃO OU PROCEDIMENTO

A implementação foi realizada de maneira iterativa, começando pela criação das rotas básicas para os CRUDs. Para a entidade professor, implementamos as operações de criação, leitura, atualização e exclusão, armazenando os dados em uma lista. A seguir, implementamos o CRUD de turmas, que exigiu a associação de um professor a cada turma. Por fim, criamos o CRUD de alunos, associando cada aluno a uma turma.
Para garantir que as funcionalidades estavam corretas, foram implementados testes automatizados com o pytest, validando as rotas de criação, listagem, atualização e exclusão de cada entidade. A API foi estruturada para ser expandida futuramente, com o objetivo de adicionar novas funcionalidades, como autenticação de usuários e implementação de um banco de dados para persistência.

4. RESULTADOS

Até o momento, a API está funcionando conforme o esperado. As funcionalidades de CRUD para professores, turmas e alunos foram implementadas e testadas com sucesso. Através dos testes automatizados, foi possível validar que todas as rotas estão respondendo corretamente, garantindo que a API seja robusta e confiável.
No entanto, a utilização de listas para armazenar os dados é uma solução provisória, já que, a cada reinicialização do servidor, os dados são perdidos. Esse problema será resolvido com a implementação de um banco de dados, planejado para a próxima fase do projeto.

5. CONCLUSÃO

A criação da API foi um sucesso até o momento. A utilização do Flask com Blueprint proporcionou uma estrutura limpa e fácil de expandir. Apesar das limitações atuais com o armazenamento em memória, conseguimos implementar uma API funcional, com operações CRUD básicas para as entidades principais. O projeto está em um bom ponto de partida, e a adição de um banco de dados e autenticação será o próximo passo para aprimorar ainda mais a aplicação.

6. IMPACTO E CONEXÃO COM O MUNDO REAL

Este projeto tem um grande potencial de aplicação no mundo real, especialmente em sistemas de gestão escolar ou educacional. A API pode ser facilmente expandida para incluir novas funcionalidades, como o gerenciamento de notas, horários e até mesmo integração com sistemas de comunicação entre professores e alunos. Além disso, o uso do Flask torna o projeto escalável, o que significa que ele pode ser ampliado para suportar um número maior de usuários e dados no futuro.

7. DESAFIOS FUTUROS E MELHORIAS

Embora a aplicação já esteja funcional, existem várias melhorias que podem ser implementadas:

1. Persistência de Dados: A solução de listas em memória é temporária e precisa ser substituída por um banco de dados para garantir a persistência dos dados a longo prazo.
2. Autenticação e Autorização: Implementar um sistema de login para proteger rotas que exigem permissão de acesso, como a criação ou edição de registros.
3. Documentação da API: Gerar uma documentação completa da API usando ferramentas como Swagger, para facilitar o uso por desenvolvedores que possam vir a integrar a API em outros sistemas.
4. Melhorias na Interface: Criar uma interface mais amigável e interativa para os usuários da API, caso o sistema seja exposto para utilização em larga escala.