**Relatório**

**1**

**API - School System**

Grupo 1: Enzo Scartezino Gropo(2401119), Giovanni Senna ferreira(2400536)

João Manoel Lelis Oliveira(2401807), Murilo Gomes Sardinha(2401763)

Data da atividade: 19/03/2025

**Objetivo**

Este relatório tem como objetivo documentar o desenvolvimento de uma API de microserviço para um sistema escolar, utilizando Flask em Python. Nesta primeira entrega, nosso foco é a implementação das operações básicas de CRUD (Create, Read, Update e Delete), garantindo a correta manipulação dos dados. Além disso, realizaremos testes para validar a funcionalidade e a integridade da codificação do CRUD, assegurando a base necessária para futuras evoluções do sistema.

1. **Introdução**

Este relatório tem como finalidade documentar o desenvolvimento de uma API de microserviço para um sistema escolar, conforme solicitado na disciplina. O objetivo desta primeira entrega é implementar um CRUD (Create, Read, Update e Delete) utilizando o framework Flask em Python, permitindo a manipulação de dados essenciais do sistema. Além disso, serão realizados testes para validar o funcionamento da API, garantindo sua integridade e confiabilidade. A realização deste trabalho visa proporcionar uma experiência prática no desenvolvimento de APIs e no uso de boas práticas de programação.

1. **Descrição e Análise do Caso**

O projeto consiste no desenvolvimento de uma API de microserviço para um sistema escolar, utilizando Flask em Python. A API será responsável por gerenciar informações essenciais, permitindo a criação, leitura, atualização e exclusão de dados (CRUD). Essa abordagem visa proporcionar maior modularidade e escalabilidade ao sistema, facilitando sua manutenção e evolução.

Nesta primeira etapa, o foco será exclusivamente na implementação das operações básicas de CRUD, garantindo que a API seja capaz de manipular os dados corretamente. Além disso, serão conduzidos testes para validar a funcionalidade e integridade do código, assegurando que os endpoints respondam adequadamente às requisições realizadas.

ANÁLISE DO CASO

O desenvolvimento de uma API baseada em microserviços é uma abordagem moderna e amplamente adotada para sistemas que exigem escalabilidade e flexibilidade. No contexto de um sistema escolar, essa arquitetura permite que diferentes módulos (como gestão de alunos, professores e turmas) sejam desenvolvidos e mantidos separadamente.

Nesta fase inicial do projeto, a implementação do CRUD é essencial, pois estabelece a base para futuras funcionalidades do sistema. Os testes serão fundamentais para verificar o correto funcionamento das operações e identificar possíveis falhas no código. A escolha do Flask como framework se justifica por sua leveza e simplicidade, facilitando o desenvolvimento rápido e eficiente da API.

Com a finalização dessa primeira etapa, espera-se obter um microserviço funcional e testado, que poderá ser expandido com novas funcionalidades em fases posteriores do projeto.

1. **Implementação ou Procedimento**

Para o desenvolvimento da API de microserviço, utilizamos a linguagem Python em conjunto com o framework Flask, devido à sua leveza e facilidade na criação de APIs REST. O ambiente de desenvolvimento escolhido foi o Visual Studio Code (VS Code), proporcionando maior praticidade na escrita e organização do código.

O trabalho foi realizado de forma colaborativa, com todos os membros do grupo participando ativamente do desenvolvimento. Durante a implementação, seguimos uma abordagem estruturada para garantir a correta funcionalidade da API. Inicialmente, configuramos o ambiente de desenvolvimento e instalamos as bibliotecas necessárias. Em seguida, criamos os endpoints para as operações CRUD (Create, Read, Update e Delete), permitindo a manipulação eficiente dos dados do sistema escolar.

Além do desenvolvimento do CRUD, elaboramos testes próprios para validar o funcionamento correto das operações. Esses testes foram fundamentais para garantir que a API respondesse corretamente às requisições, assegurando a integridade dos dados.

Para facilitar a colaboração e o versionamento do código, utilizamos o GitHub como repositório compartilhado do grupo. A cada progresso no desenvolvimento, realizamos commits e sincronizamos as alterações, garantindo um fluxo de trabalho organizado e permitindo a identificação e correção ágil de possíveis problemas.

Com essa abordagem, conseguimos construir uma API funcional e testada, seguindo boas práticas de desenvolvimento e estabelecendo uma base sólida para futuras melhorias e expansões do sistema escolar.

1. **Resultados**

Ao final da implementação da primeira etapa do projeto, conseguimos desenvolver uma API funcional utilizando Flask, que atende aos requisitos propostos. A API implementa corretamente as operações CRUD (Create, Read, Update e Delete), permitindo a manipulação dos dados do sistema escolar de forma eficiente.

Os testes desenvolvidos foram executados com sucesso, validando o correto funcionamento dos endpoints e garantindo que as operações fossem realizadas conforme esperado. Isso assegurou a integridade dos dados e a confiabilidade da API.

Com a finalização desta primeira entrega, foi estabelecida uma base sólida para futuras expansões do sistema, permitindo a implementação de novas funcionalidades e aprimoramentos conforme a evolução do projeto.

1. **Conclusão**

A construção desta API de microserviço para um sistema escolar permitiu consolidar conhecimentos sobre desenvolvimento de software, arquitetura de microserviços e boas práticas de programação. A utilização do framework Flask em Python proporcionou um ambiente ágil e flexível para a implementação das operações CRUD, garantindo uma estrutura funcional e organizada.

Além da parte técnica, este projeto reforçou a importância do trabalho em equipe e da colaboração no desenvolvimento de software. A comunicação eficiente entre os integrantes do grupo foi essencial para a resolução de desafios e para a entrega de um sistema coeso. O uso do GitHub como ferramenta de versionamento possibilitou um fluxo de trabalho mais estruturado, facilitando a integração e a gestão das contribuições de cada membro.

Com esta primeira entrega, estabelecemos uma base sólida para futuras melhorias e expansões da API, permitindo a adição de novas funcionalidades conforme a evolução do projeto.

1. **Impacto e Conexão com o Mundo real**

A criação dessa API para um sistema escolar tem um grande potencial para facilitar o dia a dia das escolas. Processos como matrícula, controle de notas e frequência, que normalmente são feitos de forma manual, podem ser automatizados, tornando tudo mais rápido e organizado. Além disso, a API pode se integrar facilmente com outras plataformas, como sistemas de ensino online e pagamentos, o que torna a gestão escolar muito mais eficiente.

Ao usar a arquitetura de microserviços, o projeto mostra como a tecnologia pode transformar a forma como as escolas funcionam, permitindo que se adaptem melhor às novas necessidades e desafios. Isso abre portas para futuras inovações que podem melhorar ainda mais a experiência de alunos, professores e gestores. Com isso, não apenas resolve problemas imediatos, mas também prepara as instituições para um futuro mais ágil e conectado.

1. **Desafios Futuros e Melhorias**

Durante o desenvolvimento desta primeira etapa da API, encontramos dificuldades na hora de fazer a validação no CRUD, na hora do processo de criação do teste tivemos muita dificuldade como realizá-los e criar todos do zero, pois essa é a primeira vez que fizemos criação de testes, para testar uma API.

Encontramos dificuldade na hora de fazer um código padronizado igual a todos, para que ficassem igual e seguir um padrão certo.

O próximo passo no desenvolvimento do projeto é aprimorar nossa API utilizando o padrão MVC (Model-View-Controller) no Flask. Implementar o MVC ajudará a separar melhor as responsabilidades dentro da aplicação, tornando o código mais organizado, modular e fácil de entender. Além disso, conectaremos nossa API a um banco de dados, o que permitirá armazenar e recuperar dados de forma eficiente, tornando a API mais robusta e adequada a um ambiente profissional.

Também vemos uma possibilidade de melhoria na hora de muita repetição do código, por exemplo na validação de campos vazios essa parte tem que ser criada para todas entidades para garantir que o código tenha um desenvolvimento correto, vimos uma possibilidade de criar uma função para essas repetições de códigos, assim tornando código mais limpo e menos poluído.

Divisão de Trabalhos: a nossa metodologia foi uma divisão de trabalhos para os endpoints, porém fizemos todos juntos um em cada computador e subimos apenas com um commit no Github.

Enzo: Criação do PUT das entidades/ Teste do PUT

Giovanni: Criação do GET das entidades/ Teste do GET

João: Criação do POST das entidades e organização do GitHub / Teste DO POST

Murilo: Relatório/documentação do entregável 1 e DELETE das entidades/ Teste do DELETE

**Referências**

Documentação GitHub: disponível em: <https://docs.github.com>

Documentação Python: disponível em: <https://docs.python.org/pt-br/3/>

Documentação Flask: Disponível em: <https://flask.palletsprojects.com/>