API DE GESTÃO EDUCACIONAL COM FLASK

"SISTEMA CRUD PARA PROFESSORES, TURMAS E ALUNOS COM MYSQL, DOCKER, SWAGGER E DEPLOY"

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

1. INTRODUÇÃO

O objetivo deste projeto é demonstrar a implementação de um ecossistema baseado em microsserviços, utilizando contêineres Docker para garantir isolamento, escalabilidade e facilidade de manutenção. O foco principal está na integração entre três serviços distintos: um sistema de gerenciamento educacional e dois microsserviços auxiliares — Reservas e Atividades.
O sistema principal, denominado Sistema de Gerenciamento, já foi desenvolvido previamente e é responsável pelo controle de alunos, professores e turmas. Os novos microsserviços — Reservas e Atividades — se conectam a este sistema central utilizando IDs de turmas e professores para assegurar consistência e integridade das informações.
A arquitetura adotada visa promover a independência dos serviços, facilitando a escalabilidade e o desenvolvimento paralelo por diferentes equipes.

2. DESCRIÇÃO E ANÁLISE DO CASO

O projeto se divide em três repositórios independentes, cada um representando um serviço com função específica no ecossistema:

- Sistema de Gerenciamento: Responsável pelo cadastro e gerenciamento de entidades como alunos, professores e turmas. Fornece as APIs que servem de base para os outros microsserviços.
- Reservas: Microsserviço responsável pelo gerenciamento de reservas de salas de aula, com dependência do ID da turma proveniente do Sistema de Gerenciamento.
- Atividades: Microsserviço responsável pelo controle das atividades dos professores, utilizando o ID do professor fornecido pelo sistema central.

Todos os serviços seguem o padrão arquitetural MVC (Model-View-Controller) e utilizam Docker para conteinerização. A persistência de dados é feita com SQLite e MySQL, e as APIs expõem rotas com suporte aos verbos HTTP GET e POST.

3. IMPLEMENTAÇÃO OU PROCEDIMENTO

Passos para implementação:

1. Estruturação dos projetos com base no padrão MVC.
2. Criação dos modelos de dados adequados (professores, turmas, atividades, salas, reservas).
3. Desenvolvimento das rotas REST com suporte a GET e POST.
4. Integração entre serviços por meio de requisições HTTP, utilizando os IDs compartilhados.
5. Conteinerização com Docker, incluindo arquivos Dockerfile e docker-compose.yml para facilitar a execução dos serviços.

Cada repositório contém um README.md com instruções detalhadas de execução, arquitetura e integração dos serviços.

4. RESULTADOS

Todos os serviços foram implementados e testados com sucesso. A integração entre eles foi validada por meio de chamadas HTTP entre os contêineres Docker.

- O microsserviço de Reservas consegue acessar o ID de turmas e criar reservas associadas.
- O microsserviço de Atividades vincula atividades com base no ID de professores existentes.
- A comunicação entre serviços é feita de forma eficiente, mantendo a coesão dos dados.

5. CONCLUSÃO

A abordagem baseada em microsserviços demonstrou ser eficaz para o desenvolvimento de sistemas modulares e escaláveis. A integração entre os serviços foi realizada com sucesso, respeitando as dependências estabelecidas entre eles.
A utilização de Docker simplificou o ambiente de desenvolvimento, garantindo que todos os serviços pudessem ser executados de forma consistente e isolada.

6. IMPACTO E CONEXÃO COM O MUNDO REAL

A arquitetura de microsserviços reflete uma tendência real no desenvolvimento de software moderno, sendo amplamente adotada por empresas que buscam agilidade, escalabilidade e independência entre equipes.
Este projeto simula um cenário real de integração entre sistemas educacionais, podendo ser expandido para instituições reais, facilitando o gerenciamento de recursos e planejamento acadêmico.

7. DESAFIOS FUTUROS E MELHORIAS

Durante o desenvolvimento, identificaram-se pontos para melhorias:

- Implementar autenticação e autorização nas APIs para segurança dos dados.
- Adotar mensageria (como RabbitMQ) para comunicação assíncrona entre serviços.
- Criar uma interface web unificada para facilitar o uso do sistema por usuários finais.
- Hospedar os serviços em um ambiente de nuvem, com orquestração (ex.: Kubernetes) e monitoramento em tempo real.