FACULDADE IMPACTA

PROGRAMA DE PÓS-GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Projeto Flask

Beatriz Bramont Oliveira

Giovanna Petrilli Venditti

Isadora Cristyne de Lima Silva

Vinicios de Lima Basteiro

São Paulo

10/04/2025

Faculdade IMPACTA

Relatório

2

Relatório Projeto Flask - Entrega 2

Objetivo

Este projeto tem como objetivo apresentar a eficiência do framework Flask na linguagem Python, colocando em prática conceitos como Interface de Programação de Aplicações (API 's), dicionários, programação orientada a objetos, TDD (Test-Driven Development) e MVC.

1. Introdução

O projeto desenvolvido teve como foco a criação de uma API estruturada. Nesta segunda entrega, foi implementado o padrão MVC (Model-View-Controller), uma abordagem que organiza o código, divide responsabilidades e facilita a manutenção e escalabilidade. O padrão MVC, amplamente utilizado em aplicações web e APIs, demonstra sua relevância ao trazer organização e escalabilidade.

Model: O modelo representa os dados e as regras do negócio da aplicação. Ele é responsável por armazenar e manipular as informações, além de notificar outros componentes sobre quaisquer alterações. Em termos práticos, o modelo é a camada que define e gerencia a lógica de dados da aplicação.

View: A visão é a camada de apresentação, ou seja, a interface que o usuário interage. Ela recebe os dados do modelo e os exibe de forma compreensível e

amigável. A visão é responsável por formatar e apresentar as informações ao usuário, independentemente da origem dos dados.
Controller: O controlador atua como o intermediário entre o model e a view. Ele recebe as solicitações do usuário, processa-as e interage com o modelo para obter ou modificar os dados. Em seguida, o controlador seleciona a visão apropriada para exibir os resultados ao usuário. Essencialmente, o controlador coordena o fluxo de informações e a lógica da aplicação.

2. Descrição e Análise do Caso

Nesta etapa do projeto foi estudado MVC. Para o desenvolvimento desta aplicação seguimos utilizando o Visual Studio Code como ambiente de codificação, o Git e o GitHub para compartilhamento e controle de versionamento dos códigos. No decorrer do projeto optamos por criar uma terceira branch para edição dos códigos, utilizando a branch de desenvolvimento como um backup de código já em funcionamento, mas não como código final.

3. Implementação ou Procedimento

Configuração do Ambiente:

- Isolamento das dependências separadas por pastas (controller e model).

Models

Organizamos o CRUD de cada entidade em arquivos separados por modelAluno, modelTurma, modelProfessor, cada um com suas validações, estrutura de dados, e retornos. Também importamos models que possuem referências de outras entidades.

Controller

Utilizando a biblioteca Blueprint, instanciamos cada entidade em variáveis, facilitando assim o CRUD das mesmas no arquivo principal. Esta biblioteca nos permite dividir a aplicação em componentes reutilizáveis, otimizando a organização do programa, evitando a quebra do código.

App.py

Configuramos o arquivo principal para chamar as rotas (ou seja, os caminhos que a API responde), o config, que é responsável por parametrizar as rotas (host, port, debug), e as instâncias do blueprint.

Desafios e Decisões:

- Garantia de Qualidade: garantir que os testes passassem após as modificações;
- Arquivo de configurações: parametrizar a porta que seria utilizada pela API. Desenvolvemos o projeto em uma rede corporativa, onde a porta 8000 não é considerada segura. Precisamos alterar para 8001, para que assim a aplicação rodasse corretamente;
- Criação da terceira branch: A branch de origem para nossa branch de edição estava desatualizada, já que a primeira entrega foi realizada diretamente na main;
- Organização do grupo: A divisão das responsabilidades seguiu desbalanceada, sobrecarregando uns mais que os outros. A execução incompleta de algumas tarefas acarretou em retrabalhos para outros integrantes.

Divisão de tarefas:

- Líder: Beatriz Bramont Oliveira
    - Repartição dos modelos, ajustes modelAluno e no app.py, criação do controller, garantir que os testes passassem após mudanças, relatório.
- Isadora: Suporte na repartição dos modelos, ajustes modelAluno, modelProfessor, modelTurma, suporte na criação do controller, relatório.
- Giovanna: Ajustes modelTurma, relatório.
- Vinicios: Relatório

4. Resultados

Testes ou Observações Realizadas

Depois de desenvolver a API, fizemos uma série de testes automatizados para garantir que tudo estivesse funcionando corretamente. Esses testes foram escritos como se estivéssemos simulando ações reais de um usuário — como acessar informações, criar dados ou até mesmo errar intencionalmente para ver se o sistema contorna a situação.

Os resultados atenderam às expectativas?

- A API respondeu bem aos testes onde recebia as informações como esperado;
- Quando mandamos dados errados, ela não travou nem deu respostas confusas — mostrou mensagens claras de erro, o que é ótimo para o usuário.

Esses testes mostraram que a aplicação está funcionando como o esperado e pronta para ser usada.

5. Conclusão

O desenvolvimento desta API RESTful, utilizando o framework Flask e a arquitetura Model-View-Controller (MVC), foi conduzido visando a criação de uma solução robusta e bem estruturada para a gestão de informações em ambientes educacionais, abrangendo dados de alunos, professores e turmas. A implementação do padrão MVC, permitiu a separação clara de responsabilidades entre as diferentes camadas, resultando em um código modular, que facilitou os processos de teste e manutenção durante o desenvolvimento.

Os Controllers, atuam como intermediários, recebendo requisições HTTP e coordenando as interações com os modelos, garantindo o fluxo eficiente de informações.

Em resumo, a implementação adequada da arquitetura MVC, não apenas aprimora a qualidade do código, mas também impulsiona a eficiência e eficácia da API como um todo, tornando-a uma solução ideal para a gestão de informações em ambientes educacionais.

6. Impacto e Conexão com o Mundo Real

A partir da implementação completa da arquitetura MVC, o impacto e a conexão com o mundo real se intensificam significativamente, transcendendo a mera funcionalidade técnica e moldando a experiência educacional a partir da API desenvolvida pelo grupo. Podemos destacar dois cenários onde a nossa API irá ser de suma importância na experiência de nossos usuários.

Impacto Estrutural e Operacional:

Centralização e Organização: A implementação do MVC oferece uma organização clara e estruturada do código. Essa separação de responsabilidades facilita a manutenção, o desenvolvimento colaborativo e a evolução contínua do sistema.

Conexão com o Mundo Real: Permite que a instituição de ensino adapte rapidamente seu sistema às mudanças nas regulamentações, no corpo docente ou nas necessidades dos alunos, minimizando interrupções e otimizando recursos.

Conexão com o Mundo Real: Uma arquitetura MVC bem implementada permite que o sistema seja facilmente escalado para atender ao crescimento da instituição. A adição de novos recursos, a integração com outros sistemas e a adaptação a diferentes dispositivos (web, mobile) tornam-se tarefas mais simples e eficientes.

Impacto na Experiência do Usuário:

Conexão com o Mundo Real: Elimina a fragmentação da experiência educacional, simplificando o acesso à informação e promovendo a colaboração e o engajamento dos alunos.

7. Desafios Futuros e Melhorias

Tratando de melhorias futuras ao código , iremos implementar um banco de dados contendo informações sobre alunos, professores e turmas, que serão acessados, trabalhados e retornados pela API já implementada com o padrão MVC. Posteriormente, também integrar com outra API de microsserviços.

Referências

Introdução ao Padrão MVC: Primeiros passos na Arquitetura MVC. Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>.

O que é MVC? Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mvc#google_vignette>.

REDAÇÃO LYCEUM. O que é uma API? Tire esta e outras dúvidas! Disponível em: <https://blog.lyceum.com.br/o-que-e-api/>.