**Relatório**

**2**

**API – MVC, Model View e Controller**

**Grupo 1:** Enzo Scartezino Gropo (2401119) (ADS), Giovanni Senna ferreira(2400536) (SI), João Manoel Lelis Oliveira(2401807) (ADS), Murilo Gomes Sardinha(2401763) (SI)

Data da atividade: 10/04/2025

**Objetivo**

Justificativa do porque foi adicionado o método de arquitetura MVC(Model, View e Controller), na API de sistema escolar em flask.

1. **Introdução:**

Até o momento todo o código estava em um único arquivo, o que dificultava a manutenção do código, o reaproveitamento e a expansão da API, sem contar que uma manipulação de uma rota poderia afetar outra que não tem conexão, por esses motivo, usar o padrão de arquitetura MVC se torna uma boa opção.

1. **Justificativa:**

O padrão MVC separa a API em três camadas principais:

* **Model:** Responsável pela lógica de dados e interação com o banco, que seria: Consultas, atualizações, validações de dados etc.
* **View:** Representa o que é apresentado ao usuario. Nesse nosso caso, refere-se a criação da API e ligação dos arquivos a uma só API.
* **Controller:** Responsável pelo fluxo da aplicação, onde são tratadas as requisições, chamadas aos models e definição das respostas e a definição das rotas.

A APi nesse padrão de arquitetura trouxe as seguintes vantagens em relação a arquitetura antiga de tudo em um só arquivo:

1. **Melhor organização do código:** Com a separação das responsabilidades, cada parte do código tem sua função bem definida, o que facilita a leitura, compreensão e localização de trechos específicos.
2. **Facilidade de manutenção e testes:** Modificações em regra de negócio, rotas ou estrutura de dados podem ser feitas com menos impacto em outras partes do sistema, assim a chance é zero de uma alteração atrapalhar alguma outra rota
3. **Maior escalabilidade:** A nova estrutura está bem mais preparada para suportar o crescimento do sistema que a primeira, com a adição de novas rotas, modelos e funcionalidades sem comprometer a integridade ou legibilidade do projeto
4. **Reutilização de código:** Com o uso de controllers e models independentes, é possível reaproveitar funções e classes em diferentes partes do sistema, evitando duplicações
5. **Problemas:**

Não teve muitas dificuldades na implementação e na separação, mas na definição das rotas teve problemas pois não se tinha conhecimento que nos testes estava diferente, e agora na url tinha “/api” no começo, além disso só pequenos erros de código e indentação, porém nenhuma grande dificuldade que se demandou muito tempo, novamente reforçando que é de fácil aplicação, criação e expansão usando a arquitetura MVC, evitando conflitos ente as rotas e melhor organização.

1. **Conclusão**

A adoção do padrão MVC trouxe benefícios concretos à organização, manutenção e expansão da API. A mudança foi necessária e estratégica para garantir a qualidade do código, segurança do código evitando conflitos entre as rotas e a produtividade da equipe e a sustentabilidade do projeto a longo prazo