FIAP GRADUAÇÃO

FIAP

SISTEMAS DE INFORMAÇÃO

MICROSERVICE AND WEB ENGINEERING

PROF. Thiago Xavier

Microsserviço - Arquitetura e conexão NoSql

Agenda

Aula de Hoje:

- Conceitos de arquitetura: Controller, model e repository pattern;
- Conexão banco noSql;

Separação de responsabilidades

- **Controller:**
  - requisições http;
  - lógica de apresentação,
  - ponte entre sistema e “usuário”;
- **Model:**
  - estrutura de dados;
  - definir os esquemas do documento;
- **Repository:**
  - Comunicação com a base de dados;
  - Abstração da camada de acesso aos dados.

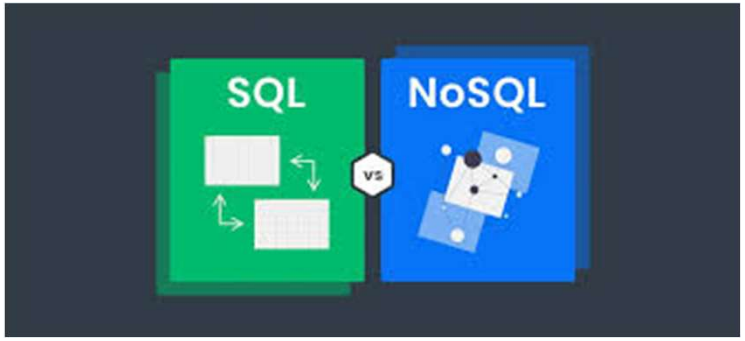
Facilidade de manutenção e escalabilidade

- **Manutenção:** ao manter a regra de negócio na model, operações de dados na Repository e apresentação na controller teremos:
  - Mudanças serão feitas de forma separada;
  - As lógicas ficam isoladas;
  - Evita que mudanças impacte em outras partes do sistema.

- **Testabilidade:**
  - Ao separar as responsabilidades, facilita a criação de teste de unidade;
  - Garante que podemos testar model, Controller e repository, antes de integrá-los.
- **Reutilização de código:**
  - A lógica da repository pode ser utilizada em parte das aplicações;
  - Reutilizada em outros projetos

Estrutura dos dados

- **SQL:**
  - dados armazenados em tabelas;
  - esquemas fixos;
  - tipos de dados definidos.
- NoSql:
  - armazenamento flexível;
  - estruturas: Json, par valor, coluna se grafos;
  - sem esquema fixo.



Banco noSql - Vantagens

- **Flexibilidade de Esquema:** Dados podem ser armazenados sem um esquema pré-definido;
- **Alto desempenho:** ideal para grandes volumes;
- **Suporte de dados não estruturados:** Sem estrutura fixa, aceita Json, XML, etc.

Banco noSql - Desvantagens

- **Complexidade:** pode ser mais complexo para configurar e gerenciar;
- **Falta de suporte a transações:** Nem todos suportam transações;
- **Menor maturidade:** Algumas soluções podem não ser tão maduras quanto banco de dados relacionais.

Show me code

- Vamos seguir o material de apoio;

Microsserviço - Vamos praticar

Exercício 3: Status 202 (Accepted)

Descrição: Crie uma rota POST que simule a criação de um processo assíncrono para enviar um e-mail. Retorne o status 202 com um ID para rastrear o envio.

Requisitos:

1. Crie uma rota `POST /email` que aceite um JSON com um campo `email`.
2. Retorne um status 202 (Accepted) com um `processId` para rastrear o envio do e-mail.

Exercício 4: Status 204 (No Content)

Descrição: Crie uma rota DELETE que exclua um usuário. Quando o usuário for excluído com sucesso, retorne o status 204.

Requisitos:

1. Crie uma rota `DELETE /users/:id` para excluir um usuário pelo ID.
2. Retorne um status 204 (No Content) após a exclusão.

Exercício 5: Status 400 (Bad Request)

Descrição: Crie uma rota POST que exige um campo obrigatório no corpo da requisição. Se o campo não estiver presente, retorne o status 400.

Requisitos:

1. Crie uma rota `POST /update-email` que aceite um JSON com um campo `email`.
2. Se o campo `email` não estiver presente, retorne um status 400 (Bad Request) com uma mensagem de erro.

Microsserviço - Vamos praticar

Exercício 6: Status 404 (Not Found)

Descrição: Crie uma rota GET para recuperar um usuário pelo ID. Se o usuário não for encontrado, retorne o status 404.

Requisitos:

1. Crie uma rota GET /users/:id para recuperar um usuário por ID.
2. Se o usuário não existir, retorne um status 404 (Not Found).

FIAP