ALEXANDRE JUNIOR
PROFESSOR E DESENVOLVEDOR

Criando APIs REST com 



@profalexandrejr

Aula 2

Spring MVC e Arquitetura em Camadas

Organizando a Casa

ALEXANDRE JUNIOR
PROFESSOR E DESENVOLVEDOR

Para não virar uma "bagunça", dividimos o código em responsabilidades.

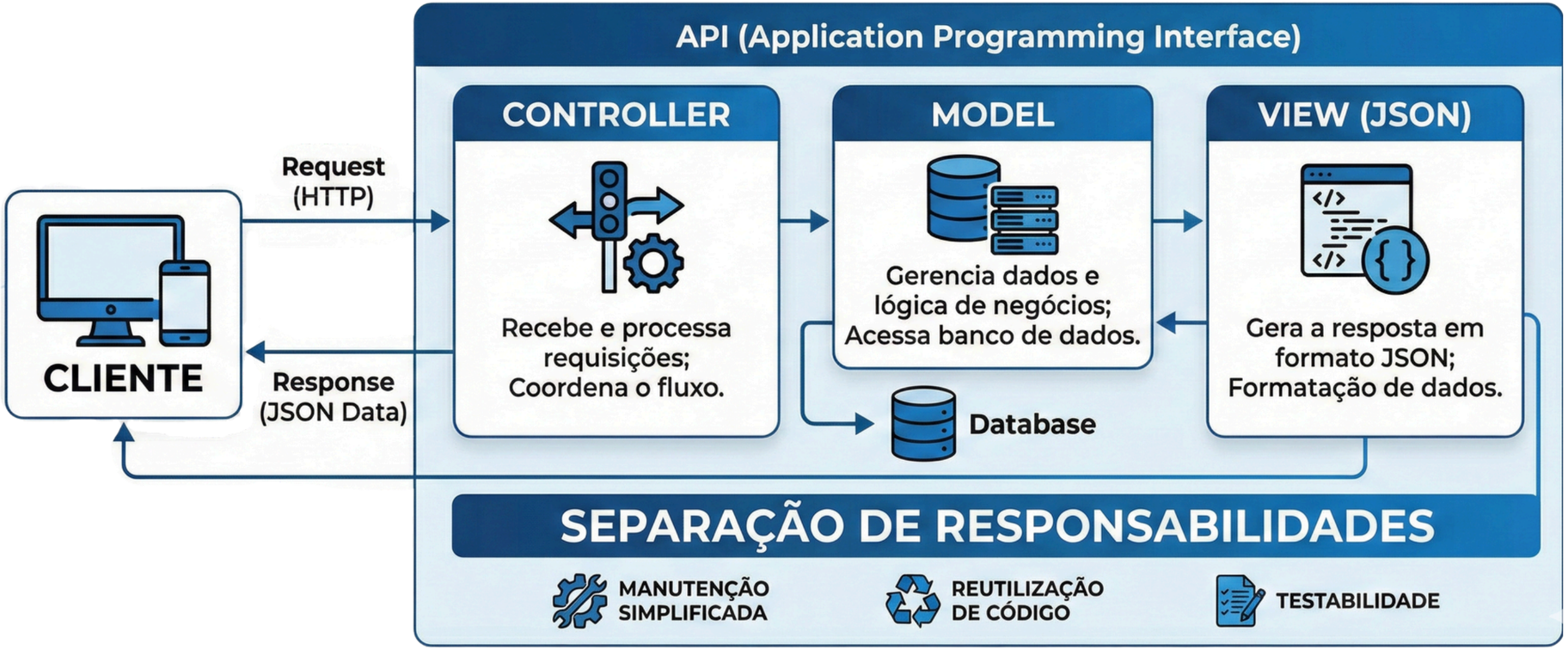
O fluxo de uma requisição é:

- **Controller (A Porta):** Recebe a requisição HTTP (o pedido do usuário/cliente).
- **Service (O Cérebro):** Onde fica a regra de negócio (validações, cálculos).
- **Repository (O Arquivo):** O único autorizado a falar diretamente com o Banco de Dados.
- **Entity/Model (O Objeto):** Representa a tabela do banco em forma de classe Java.

@profalexandrejr

O Padrão MVC no Contexto de APIs

ALEXANDRE JUNIOR
PROFESSOR E DESENVOLVEDOR



@profalexandrejr

Mapeando o Banco de Dados

ALEXANDRE JUNIOR
PROFESSOR E DESENVOLVEDOR

- **Arquivo *application.properties*:** Onde colocamos a senha e o endereço do MySQL.

```
spring.datasource.url=jdbc:mysql://localhost:3306/sgp_db
spring.datasource.username=root
spring.datasource.password=sua_senha
spring.jpa.hibernate.ddl-auto=update
```

Nota: *update* cria/atualiza as tabelas automaticamente baseadas no código Java.

Anotações da Entidade

ALEXANDRE JUNIOR
PROFESSOR E DESENVOLVEDOR

- **@Entity:** Diz ao Spring que essa classe é uma tabela.
- **@Id e @GeneratedValue:** Define a Chave Primária (PK).

```
@Entity
public class Usuario {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    // getters e setters...
}
```

Nota: criar entidade *Editora* e relacioná-la com a entidade *Livro*.

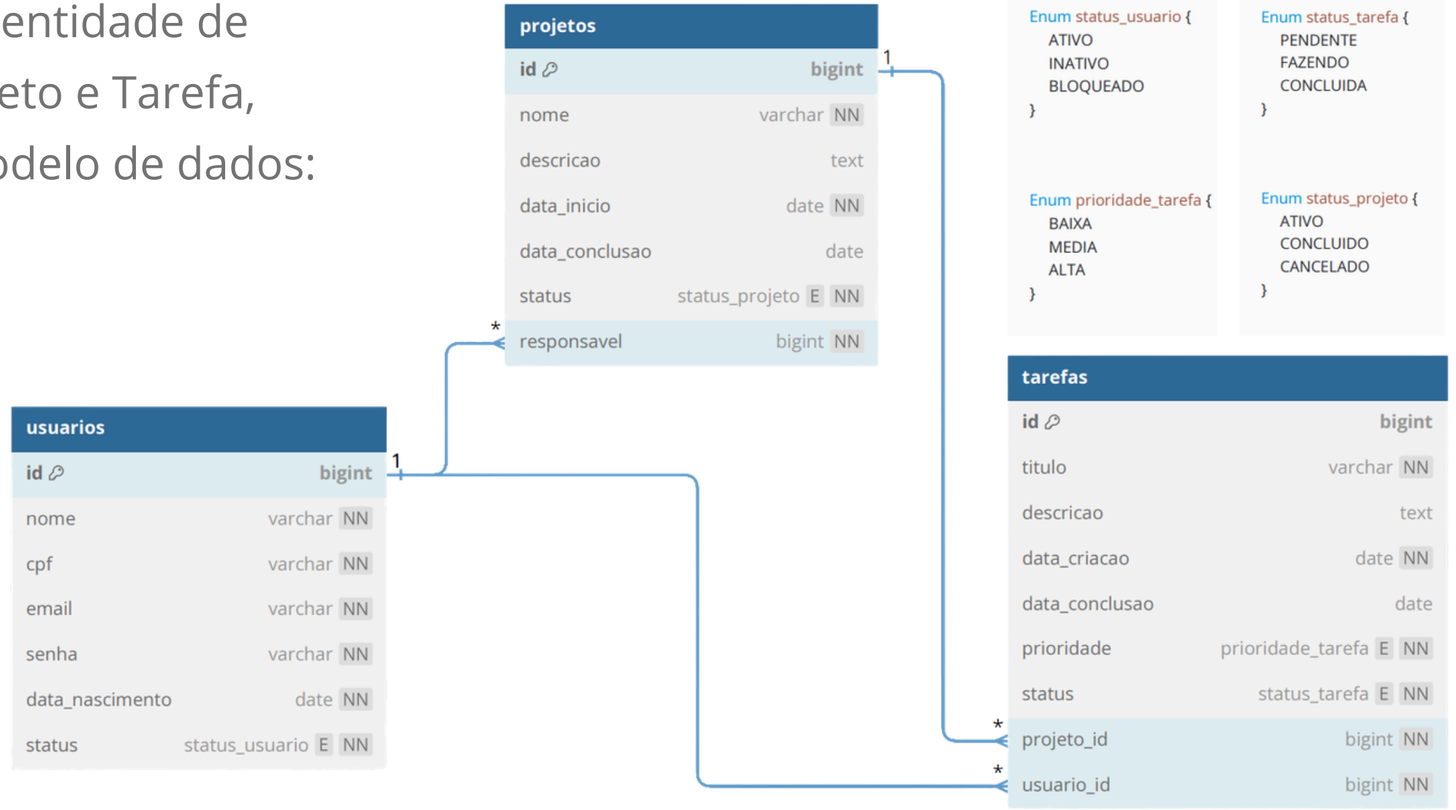


@profalexandrejr

ORM (Object-Relational Mapping)

ALEXANDRE JUNIOR
PROFESSOR E DESENVOLVEDOR

- Criar classes-entidade de Usuário, Projeto e Tarefa, conforme modelo de dados:



@profalexandrejr

Obrigado!



@profalexandrejr