Java Spring Boot

API:
Application Program Interface//<= endpoint

```
@GetMapping("/soma")
public float soma(float a, float b){
       return a + b;
}
API => Comunicação de aplicativos
```

GET(SELECT) -> cONSULTA UM RECURSO NO SERVIDOR

POST(INSERT) -> INSERE UM RECURSO NO SERVIDOR

DELETE(REMOVE) -> REMOVE UM RECURSO NO SERVIDOR

PUT(UPDATE) -> ATUALIZA VÁRIOS RECURSOS NO SERVIDOR
-> Altera todos os recursos do servidor
-> Update sem clausula Where(bd)
PATCH(UPDATE WHERE) -> ATUALIZA UM RECURSO NO SERVIDOR
->Altera um recurso em especifico, update com clausula
JAVA(ORIENTADA A OBJETO) <-> POSTGRESSQL(BD RELACIONAL)

MAPEAMENTO OBJETO-RELACIONAL(ORM)(UTILIZA-SE O JPA)

Front end - Insomnia --passa--> FilmeDTO

Back end - JavaSpringBoot --FilmeDTO-->

controler
e se comunicam atraves do filmedto
service se comunica com o postgres atraves do filme entity

FilmeDTO e filmEntity se comunicam atraves do converteDTO e atraves do convertEntity

---

O mapeamento objeto-relacional (ORM) é uma técnica que permite que desenvolvedores interajam com um banco de dados relacional usando conceitos de programação orientada a objetos. No contexto do Java, uma das implementações mais comuns de ORM é o JPA (Java Persistence API). Vamos detalhar como esse processo funciona em um cenário típico usando Java Spring Boot, Insomnia, DTOs (Data Transfer Objects), e uma entidade de banco de dados.

Cenário: Aplicação de Gerenciamento de Filmes

Front-end - Insomnia

Insomnia é uma ferramenta para testar APIs RESTful. Neste cenário, ela é usada para enviar requisições para o backend. Quando o usuário deseja criar ou atualizar informações sobre um filme, ele envia os dados no formato de um `FilmeDTO`.

Back-end - Java Spring Boot

O backend é implementado usando Java Spring Boot, que facilita a criação de aplicações robustas e escaláveis.

1. **Controller**:
   - O controller recebe requisições HTTP enviadas pelo Insomnia. Ele é o ponto de entrada da aplicação backend.
   - Ele lida com objetos `FilmeDTO` que são recebidos ou enviados como resposta.

```
@RestController
@RequestMapping("/filmes")
public class FilmeController {

    @Autowired
    private FilmeService filmeService;

    @PostMapping
    public ResponseEntity<FilmeDTO> createFilme(@RequestBody FilmeDTO film
eDTO) {
        FilmeDTO createdFilme = filmeService.createFilme(filmeDTO);
        return ResponseEntity.ok(createdFilme);
    }

    // Outros endpoints...
}
```

2. **Service**:
   - O service contém a lógica de negócios da aplicação.
   - Ele se comunica com o banco de dados através da camada de repositório, mas trabalha com objetos `FilmeDTO` e `FilmeEntity`.

```
@Service
public class FilmeService {

    @Autowired
    private FilmeRepository filmeRepository;

    public FilmeDTO createFilme(FilmeDTO filmeDTO) {
        FilmeEntity filmeEntity = converteDTOParaEntity(filmeDTO);
        FilmeEntity savedFilme = filmeRepository.save(filmeEntity);
        return converteEntityParaDTO(savedFilme);
    }
```

```
    private FilmeEntity converteDTOParaEntity(FilmeDTO filmeDTO) {
        // Implementação da conversão de DTO para Entity
    }

    private FilmeDTO converteEntityParaDTO(FilmeEntity filmeEntity) {
        // Implementação da conversão de Entity para DTO
    }
}
```

3. **Repositório**:
   - A camada de repositório é responsável pela comunicação direta com o banco de dados PostgreSQL.
   - Utiliza a entidade `FilmeEntity` para realizar operações CRUD (Create, Read, Update, Delete).

```
@Repository
public interface FilmeRepository extends JpaRepository<FilmeEntity, Long>
{
    // Métodos de consulta personalizados, se necessário
}
```

Entidade de Banco de Dados e DTO

- **FilmeDTO**:
  - Um DTO é uma classe simples que contém apenas os dados necessários para a transferência entre a camada de apresentação (frontend) e a camada de lógica de negócios (backend).

```
public class FilmeDTO {
    private Long id;
    private String titulo;
    private String diretor;
    private int ano;
    // Getters e Setters
}
```

- **FilmeEntity**:
  - Uma entidade é uma classe que mapeia para uma tabela no banco de dados. Utiliza anotações JPA para definir o mapeamento.

```
@Entity
@Table(name = "filmes")
public class FilmeEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;
    private String titulo;
    private String diretor;
    private int ano;
    // Getters e Setters
}
```

Conversão entre DTO e Entity

- **ConverterDTOParaEntity**:
  - Este método converte um `FilmeDTO` em um `FilmeEntity`. É necessário para que a camada de serviço possa salvar ou atualizar a entidade no banco de dados.

```
private FilmeEntity converteDTOParaEntity(FilmeDTO filmeDTO) {
    FilmeEntity filmeEntity = new FilmeEntity();
    filmeEntity.setId(filmeDTO.getId());
    filmeEntity.setTitulo(filmeDTO.getTitulo());
    filmeEntity.setDiretor(filmeDTO.getDiretor());
    filmeEntity.setAno(filmeDTO.getAno());
    return filmeEntity;
}
```

- **ConverterEntityParaDTO**:
  - Este método converte um `FilmeEntity` em um `FilmeDTO`. É necessário para que a camada de serviço possa retornar dados para o cliente de forma simplificada.

```
private FilmeDTO converteEntityParaDTO(FilmeEntity filmeEntity) {
    FilmeDTO filmeDTO = new FilmeDTO();
    filmeDTO.setId(filmeEntity.getId());
    filmeDTO.setTitulo(filmeEntity.getTitulo());
    filmeDTO.setDiretor(filmeEntity.getDiretor());
    filmeDTO.setAno(filmeEntity.getAno());
    return filmeDTO;
}
```

Resumo

- **Insomnia** envia um `FilmeDTO` para o backend.
- **Controller** recebe o `FilmeDTO` e passa para o **Service**.
- **Service** converte o `FilmeDTO` em uma `FilmeEntity` e usa o **Repository** para salvar no banco de dados PostgreSQL.
- O **Repository** realiza operações no banco de dados usando a `FilmeEntity`.

- **Service** converte a `FilmeEntity` salva de volta para um `FilmeDTO` e retorna para o **Controller**.
- **Controller** envia o `FilmeDTO` de volta como resposta ao Insomnia.

Este processo garante que a lógica de negócios e a comunicação com o banco de dados sejam gerenciadas de forma eficiente e separada, utilizando os conceitos de ORM com JPA e Spring Boot.

FILME CONTROLLER

```
package unifacef.edu.netflix.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import unifacef.edu.netflix.model.dto.FilmeDTO;
import unifacef.edu.netflix.model.service.FilmeService;

import java.util.List;

@RestController
@RequestMapping("/filme")
public class FilmeController {

    @Autowired
    FilmeService injecao;
    @PostMapping
    public FilmeDTO insere(@RequestBody FilmeDTO filmeDTO){
        return injecao.insere(filmeDTO);
    }

    //lista todos os filmes

    @GetMapping
    public List<FilmeDTO> consultaTodos(){
        return injecao.consultaTodos();
    }

    @GetMapping("/{id}")
    public FilmeDTO consultaPorId(@PathVariable Long id){
        return injecao.consultaPorId(id);
    }
    @DeleteMapping("/{id}")
    public String remove(@PathVariable Long id){
        return injecao.remove(id);
    }
    @PutMapping
    public List <FilmeDTO> aumentaNota(){
```

```
        return injecao.aumentaNotas();
    }
}
```

FILME CONTROLLER

```
package unifacef.edu.netflix.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import unifacef.edu.netflix.model.dto.FilmeDTO;
import unifacef.edu.netflix.model.service.FilmeService;

import java.util.List;

@RestController
@RequestMapping("/filme")
public class FilmeController {

    @Autowired
    FilmeService injecao;
    @PostMapping
    public FilmeDTO insere(@RequestBody FilmeDTO filmeDTO){
        return injecao.insere(filmeDTO);
    }

    //lista todos os filmes

    @GetMapping
    public List<FilmeDTO> consultaTodos(){
        return injecao.consultaTodos();
    }

    @GetMapping("/{id}")
    public FilmeDTO consultaPorId(@PathVariable Long id){
        return injecao.consultaPorId(id);
    }
    @DeleteMapping("/{id}")
    public String remove(@PathVariable Long id){
        return injecao.remove(id);
    }
    @PutMapping
    public List <FilmeDTO> aumentaNota(){
        return injecao.aumentaNotas();
```

```
    }
}
```

FILME ENTITY

```
package unifacef.edu.netflix.model.entity;

import jakarta.persistence.*;

@Entity // classe que é uma entidade de banco de dados
@Table(name="filme") // classe é uma tabela do banco de dados
public class FilmeEntity {
    @Id // chave primário
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name="nome")
    private String nome;
    @Column(name="nota")
    private float nota;
    @Column(name="anoLancamento")
    private int anoLancamento;
    public void setId(Long id) {
        this.id = id;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setNota(float nota) {
        this.nota = nota;
    }

    public void setAnoLancamento(int anoLancamento) {
        this.anoLancamento = anoLancamento;
    }

    public Long getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }
```

```
    public float getNota() {
        return nota;
    }

    public int getAnoLancamento() {
        return anoLancamento;
    }

    public FilmeEntity(Long id, String nome, float nota, int anoLancamento) {
        this.id = id;
        this.nome = nome;
        this.nota = nota;
        this.anoLancamento = anoLancamento;
    }

    public FilmeEntity() {
    }
}
```

FILME DTO

```
package unifacef.edu.netflix.model.dto;

public class FilmeDTO {
    private Long id;
    private String nome;
    private float nota;
    private int anoLancamento;

    public FilmeDTO(Long id, String nome, float nota, int anoLancamento) {
        this.id = id;
        this.nome = nome;
        this.nota = nota;
        this.anoLancamento = anoLancamento;
    }

    public FilmeDTO() {
    }

    public Long getId() {
        return id;
    }
```

```
    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public float getNota() {
        return nota;
    }

    public void setNota(float nota) {
        this.nota = nota;
    }

    public int getAnoLancamento() {
        return anoLancamento;
    }

    public void setAnoLancamento(int anoLancamento) {
        this.anoLancamento = anoLancamento;
    }
}
```

FILME REPOSITORY

```
package unifacef.edu.netflix.model.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import unifacef.edu.netflix.model.entity.FilmeEntity;

public interface FilmeRepository extends JpaRepository<FilmeEntity, Long> {
    //Esta interface vai conter todos os métodos de CRUD
    //Relacionado ao FIlme Entity
    //Create, Retrieve/Read, Update e Delete
}
```