Disciplina de Desenvolvimento Web III

PROFESSOR JEFFERSON CHAVES

jefferson.chaves@ifpr.edu.br

ROTEIRO DE AULA

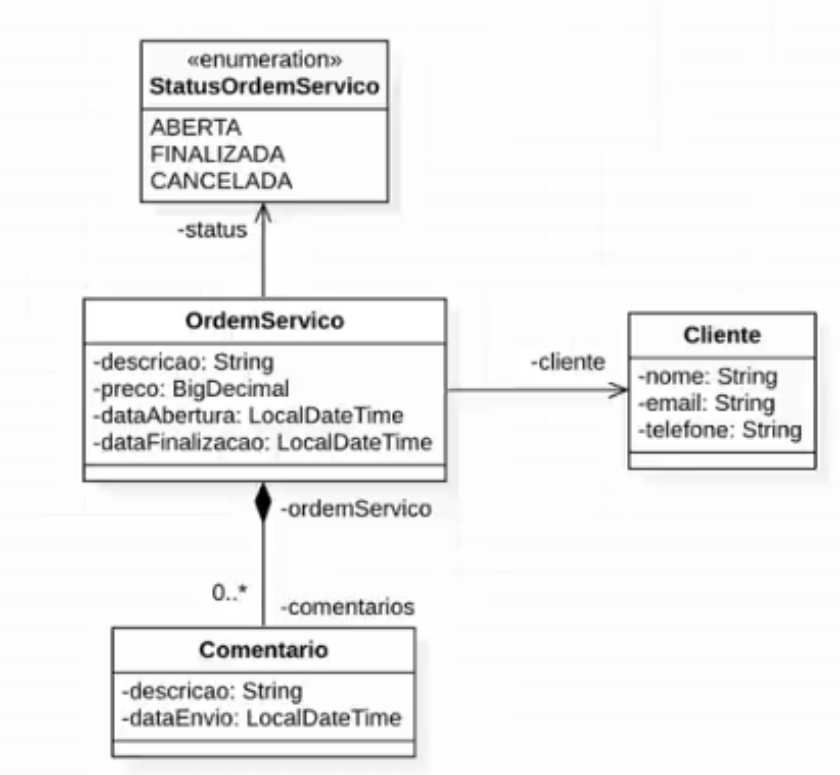
PERSISTÊNCIA DE DADOS

Objetivo geral:

- Implementar uma API Rest para Gestão de Ordens de Serviço utilizando o Framework Spring como base;

Requisitos:

- Neste projeto, poderão ser criadas ordens de serviço, associadas a um determinado cliente;
- Podem ser inseridos comentários em cada ordem de serviço.
- Ordens de serviço pode ser listadas;
- Podem ser listadas ordens de serviço pelo e-mail de um determinado cliente;



Ferramentas necessárias

- MySQL Server;
- MySQL Workbench (somente para conferência);
- Intellij Idea.

1. Preparação do projeto

- Crie um novo Projeto Spring Boot;
- Defina as propriedades do projeto como:
    - Name: ordem-servico-api;
    - Type: Maven;
    - Group: com.exemplo (ou seu domínio preferido);
    - Artifact: ordem-servico-api;
    - Packaging: Jar;
    - Java: 17 (ou a versão que você estiver usando).

Clique em **Next** para ir para a etapa de configuração de dependências.

- Inclua as seguintes dependências (as dependências informadas aqui, serão adicionadas ao arquivo de gerenciamento do projeto ao projeto POM.xml)
    - Developer Tools / Spring Boot Dev Tools;
    - Web / Spring Web;
    - SQL / Spring Data JPA;
    - SQL / MySQL Driver

Clique em **Create** para criar o projeto.

2. Estrutura do projeto

Antes de começar a implementação, vamos criar a estrutura do projeto:

```
src
 └── main
     ├── java
     │   └── com
     │       └── exemplo
     │           └── ordemservicoapi
     │               ├── models       -> Classes de domínio (OrdemDeServico, Cliente, Comentario)
     │               ├── repositories -> Interfaces que estendem JpaRepository para cada entidade
     │               ├── services     -> Classes de serviços para lógica de negócios
     │               └── controllers  -> Controladores REST para gerenciar os endpoints
     └── resources
         └── application.properties -> Configurações da aplicação
```

Crie a estrutura de pastas para o projeto!

3. Configuração da Unidade de Persistência

Material de apoio: application.properties

O Spring possui um arquivo de configuração do projeto chamado de **application.properties.** Esse arquivo é usado para configurar várias propriedades da aplicação, como banco de dados, servidor, cache, segurança, e outras configurações específicas do Spring. Ele permite que você customize o comportamento da aplicação sem precisar modificar diretamente o código.

Adicionar as seguintes configurações para o banco de dados da aplicação:

```
#database configuration
spring.datasource.url = jdbc:mysql://localhost:3306/ordem-servico-api?createDatabaseIfNotExist=true
spring.datasource.username = root
spring.datasource.password = bancodedados

spring.jpa.hibernate.ddl-auto = update

#exibir consultas realizadas
spring.jpa.show-sql=true
```

Destaco aqui o propriedade spring.jpa.hibernate.ddl-auto: esse comando é utilizado para controlar como serão gerenciadas a criação e atualização do esquema de banco de dados. OS valores possíveis para a propriedade são:

- **none:** Não faz nenhuma operação no esquema do banco de dados. Ou seja, o Hibernate não tenta criar, modificar ou apagar tabelas. Você fica responsável por gerenciar manualmente as alterações no esquema.
- **update:** Atualiza o esquema do banco de dados de acordo com as mudanças nas entidades Java. Se novas tabelas ou colunas forem adicionadas ao código, o Hibernate fará as modificações automaticamente no banco de dados, preservando os dados existentes. No entanto, ele não removerá ou alterará colunas já existentes de maneira destrutiva.
- **create:** Apaga o esquema existente e recria todas as tabelas a cada vez que a aplicação é iniciada. Todos os dados existentes no banco de dados serão perdidos.
- **create-drop:** Funciona como create, mas, além de criar o esquema ao iniciar a aplicação, ele também apaga todas as tabelas quando a aplicação é encerrada.
- **validate:** Verifica se o esquema do banco de dados corresponde às entidades Java, mas não faz nenhuma alteração no banco. Se houver discrepâncias, uma exceção será lançada.

3.Criação das classes de Domínio

Material de apoio: Comentario.java, Cliente.java, Cliente.java e OrdemServico.java

Normalmente, a criação das classes de um modelo conceitual é iniciada pelas classes que não não possuem dependências, ou seja, as que estão nas pontas.

Atividade:

- No pacote **models,** crie as classes Comentario e Cliente, conforme o modelo acima;
- Crie a enumeração StatusOrdemServico;
- Crie a classe OrdemServico.

Uma enumeração (ou enum) é um tipo especial de dado em linguagens de programação que permite definir um conjunto fixo de valores constantes nomeados. Em vez de utilizar valores numéricos ou strings diretamente no código, a enumeração permite que você trabalhe com valores significativos e legíveis, o que facilita a manutenção e a compreensão do código.

As classes criadas aqui são as classes Java normais. Contudo o framework Spring permite que sejam adicionados estereótipos a essas classes, a fim de transformá-lo em um **Bean**. Em Spring, um Bean é qualquer objeto que o contêiner de inversão de controle (IoC) gerencia. Quando você marca uma classe como um Bean, o Spring a gerencia, ou seja:

- Controla seu ciclo de vida.
- Instancia automaticamente.
- Injeta dependências quando necessário.

Atividade

1. Adicione a @Entity para cada uma das classes (com exceção da enumeração). Essa anotação transforma essa classe em um Bean que realiza o mapeamento destas classes para o modelo relacional.
2. Adicione as anotações:
    a. @Id
    b. @GeneratedValue(strategy=GenerationType.IDENTITY)
3. Crie um construtor vazio para cada classe;
4. Crie os getters e setters para cada classe;
5. Execute o servidor. Verifique se as tabelas foram todas criadas.

4. Criando os Repositórios

Material de apoio: ClienteRepository, ComentarioRepository, OrdemServicoRepository

Atividade:

- No pacote **repositories**, Crie as interfaces ClienteRepository, ComentarioRepository, e OrdemServicoRepository conforme modelos ;

5. Crie as Controladoras e Services

- No pacote controllers, crie as classes OrdemServicoController e ClienteController;
- No pacote services, crie a classe OrdemServicoService;
- Aguarde orientações do professor para implementação dessas classes;

5. Teste sua API com a ferramenta Postman:

- Criar Ordem de Serviço:Enviar um POST para http://localhost:8080/ordens com o seguinte JSON:

```
{
  "descricao": "Instalação de Software",
  "preco": 100.0,
  "cliente": {
    "nome": "João Silva",
    "email": "joao@gmail.com",
    "telefone": "123456789"
  }
}
```

- Listar Ordens de Serviço: Enviar um GET para

  http://localhost:8080/ordens.

- Cancelar uma Ordem de Serviço: Enviar um PUT para http://localhost:8080/ordens/{id}/cancelar em que {id} é o ID da ordem a ser cancelada.

- Finalizar uma Ordem de Serviço: Enviar um PUT para http://localhost:8080/ordens/{id}/finalizar.