|  |  |  |
| --- | --- | --- |
|  | **CENTRO UNIVERSITÁRIO DE PATOS DE MINAS – UNIPAM**  **BACHARELADO EM SISTEMAS DE INFORMAÇÃO**  **TURMA: 5º PERÍODO**  **DISCIPLINA: DESENVOLVIMENTO PARA WEB II**  **PROFESSOR RAFAEL MARINHO E SILVA** | |
| **ALUNO (A): Lucas Domingues Oliveira** | |
| **VALOR: 4 PONTOS** | **DATA DE ENTREGA: 26/05/2025** |
|
| **Obs.: O arquivo final, contendo o código e a documentação, deve ser enviado ao Portal UNIPAM até o dia 26/05/2025 (segunda-feira), às 20h40. O nome do arquivo .zip deve ser o nome do aluno. Exemplo: `Rafael-Marinho.zip`.** | | |

**ATIVIDADE PRÁTICA 03**

1 - Com base nos conceitos e exemplos apresentados durante as aulas, sobre Spring Boot, desenvolva uma aplicação completa de cadastro de veículos, aplicando JPA, Spring Data, Controllers REST e MVC com Thymeleaf. Ao final, você terá uma API e uma interface web para criar, listar e detalhar veículos.

**Instruções Gerais:**

* Use GitHub Codespaces para todo o desenvolvimento.
* Consulte os conteúdos das aulas como referência principal para estruturar o projeto.

a) Crie uma pasta na raiz do Codespace com seu nome e identificador da prática, por exemplo: seuNome\_pratica3

b) Dentro de seuNome\_pratica3, gere um novo projeto Spring Boot via Spring Initializr, nomeando-o como pratica3.

* Utilize as dependências:
  + Spring Web
  + Spring Data JPA
  + PostgreSQL Driver
  + Thymeleaf
  + Spring Boot DevTools (opcional)
* Adote o mesmo docker-compose.yml das aulas, ajustando o nome do serviço e do banco para bd\_pratica3.
* Suba o contêiner com ***docker-compose up -d*** e verifique que o PostgreSQL está disponível em ***localhost:5432***.
* Em ***src/main/resources/application.properties***, ajuste:

|  |
| --- |
| spring.application.name=projeto spring.datasource.url=jdbc:postgresql://localhost:5432/bd\_pratica3 spring.datasource.username=root spring.datasource.password=root spring.jpa.hibernate.ddl-auto=update spring.jpa.show-sql=true spring.thymeleaf.prefix=classpath:/templates/ spring.thymeleaf.suffix=.html spring.thymeleaf.cache=false |

c) - Mapeamento JPA da Entidade Veiculo

* Crie a classe Veiculo no pacote model com campos:

|  |
| --- |
| @Entity @Table(name = "veiculos") **public** **class** **Veiculo** {  @Id @GeneratedValue(strategy = GenerationType.IDENTITY)  **private** Long id;  **private** String modelo;  **private** Integer ano;  **private** BigDecimal valor;  // getters e setters } |

* Crie também as seguintes camadas:
  + VeiculoRepository
  + VeiculoService
  + VeiculoController (API REST)
  + VeiculoWebController (MVC Web)
    - cadastro (/cadastrar)
    - listagem (/listar)
    - detalhes e exclusão (/{id} e /excluir)
  + Templates Thymeleaf e CSS
    - Aproveite o estilo e fragmentos do e-book (navbar.html, form.html, lista.html, detalhe.html), adaptando rotas e nomes para veiculos.
    - Mantenha a responsividade e o CSS em static/css/style.css.
* Teste com REST Client
  + Crie o arquivo ***veiculo.http*** similar ao ***pessoa.http***.
  + **Insira pelo menos 4 veículos distintos.**

d) - Mapeamento JPA da Entidade Veiculo

* Faça capturas de tela (prints) das páginas:
  + Cadastro de veículo

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* + Detalhes de um veículo

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* + Listagem de veículos (/veiculos/listar)

Tabela

O conteúdo gerado por IA pode estar incorreto.

* Faça captura de tela (print) da resposta da API REST:
  + No arquivo veiculo.http, execute o bloco **Listar todos os veículos** e faça um print da resposta JSON.

**Boa Atividade!!!**