Missão Prática | Nível 4 | Mundo 3

Material de **orientações** para desenvolvimento da **missão**

prática do **4º nível de conhecimento**.

RPG0017 - Vamos integrar sistemas

Implementação de sistema cadastral com interface Web, baseado nas tecnologias de Servlets, JPA e JEE.

Objetivos da prática

1. Implementar persistência com base em JPA.
2. Implementar regras de negócio na plataforma JEE, através de EJBs.
3. Implementar sistema cadastral Web com base em Servlets e JSPs.
4. Utilizar a biblioteca Bootstrap para melhoria do design.
5. No final do exercício, o aluno terá criado todos os elementos necessários para
6. exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para
7. lidar com contextos reais de aplicação.

🔑 **As práticas devem ser feitas individualmente.**

Materiais necessários para a prática

- SQL Server, com o banco de dados gerado em prática anterior (loja).
- JDK e IDE NetBeans.
- Navegador para Internet, como o Chrome.
- Banco de dados SQL Server com o Management Studio.
- Equipamentos:

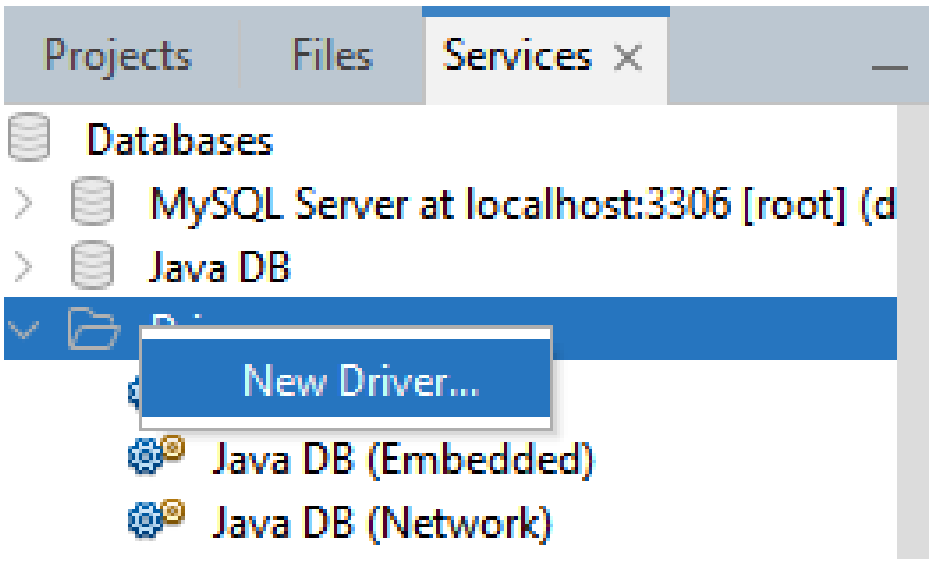
- Computador com acesso à Internet.
- JDK e IDE NetBeans.
- Banco de dados SQL Server.
- Navegador de Internet instalado no computador.

Desenvolvimento da prática

Vamos colocar a **mão na massa**?! Siga as **instruções** abaixo para desenvolvimento desta missão.

👉 **1º Procedimento | Camadas de Persistência e Controle**

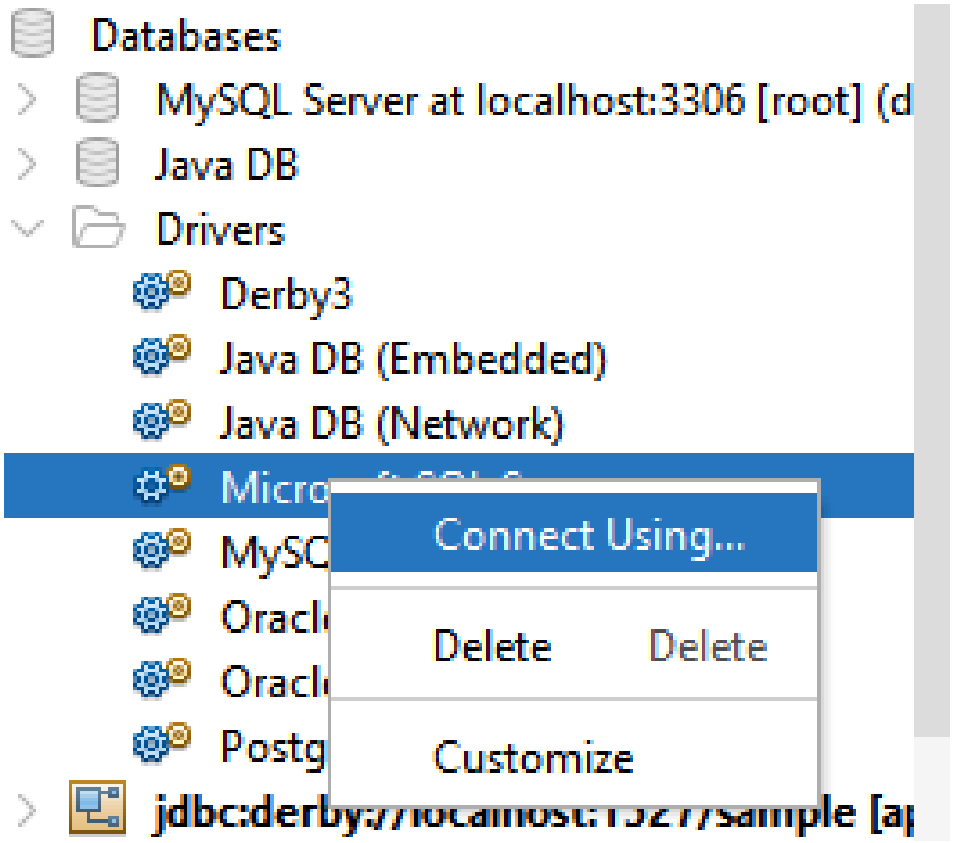
1. Configurar a conexão com SQL Server via NetBeans e o pool de conexões no
2. GlassFish Server 6.2.1:
    1. Na aba de **Serviços**, divisão **Banco de Dados**, clique com o botão direito em
    2. **Drivers** e escolha **Novo Driver.**



b. Na janela que se abrirá, clicar em **Add** (Adicionar), selecionar o arquivo **mssql-jdbc-12.2.0.jre8.jar**, que é parte do arquivo **zip** encontrado no endereço seguinte, e finalizar com **Ok**

https://learn.microsoft.com/pt-br/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16

c. O reconhecimento será automático, e podemos definir uma conexão com o clique do botão direito sobre o driver e escolha de **Conectar Utilizando.**



d. Para os campos **database**, **user** e **password**, utilizar o valor **loja**, de acordo com os elementos criados em exercício anterior sobre a criação do banco de dados de

exemplo, marcando também a opção **Lembrar Senha.**

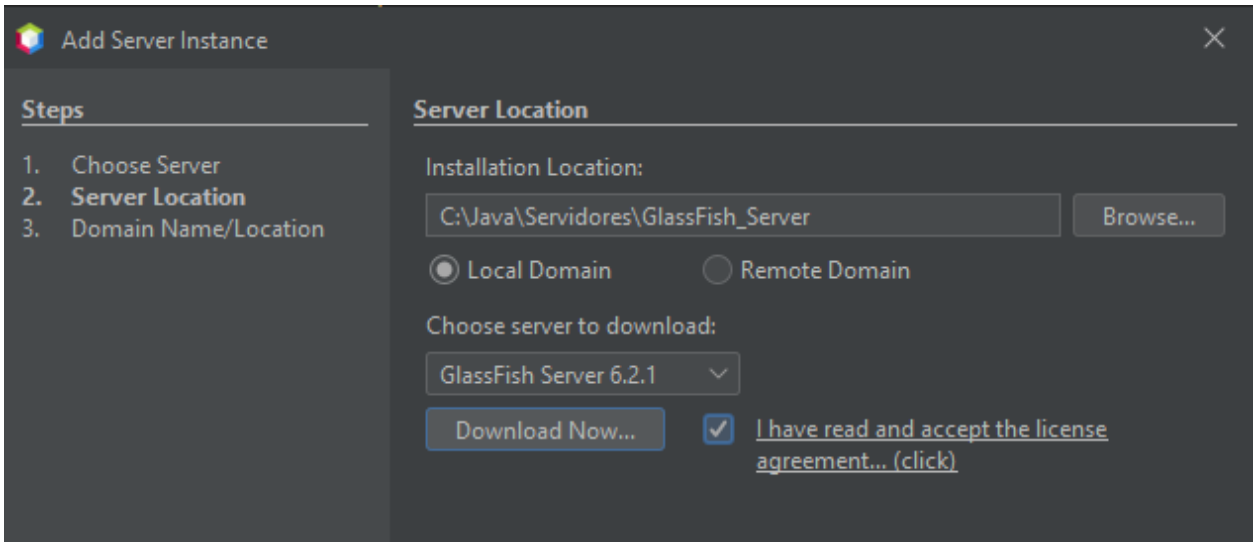
e. Para o campo JDBC URL deve ser utilizada a seguinte expressão:

jdbc:sqlserver://

localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;

f. Clicar em **Testar Conexão** e, estando tudo certo, **Finalizar.**

g. Na divisão **Servidores**, verificar se o GlassFish 6.2.1 (ou posterior) está instalado, e caso não esteja, adicionar o servidor, via clique com o botão direito e escolha da opção **Add Server**, efetuando o download a partir da própria janela que se abrirá



h. Copiar o arquivo **mssql-jdbc-12.2.0.jre8.jar** para o subdiretório **lib**, a partir do diretório de base do **GlassFish.**

i. Iniciar o servidor GlassFish a partir do NetBeans.

j. Através da linha de comando, executar o comando **asadmin**, no diretório **bin** do **GlassFish.**

k. No **prompt do asadmin**, executar o comando apresentado a seguir

create-jdbc-connection-pool

--datasourceclassname com.microsoft.sqlserver.jdbc.SQLServerDataSource

--restype javax.sql.DataSource

--property

driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver:portNumber=1433:password=loja:user=loja:serverName=localhost:databaseName=loja:trustServerCertificate=true:URL="jdbc\\:sqlserver\\://localhost\\:1433\\;databaseName\\=loja\\;encrypt\\=true\\;trustServerCertificate\\=true\\;"

l. Será solicitado o identificador do pool, que será **SQLServerPool.**

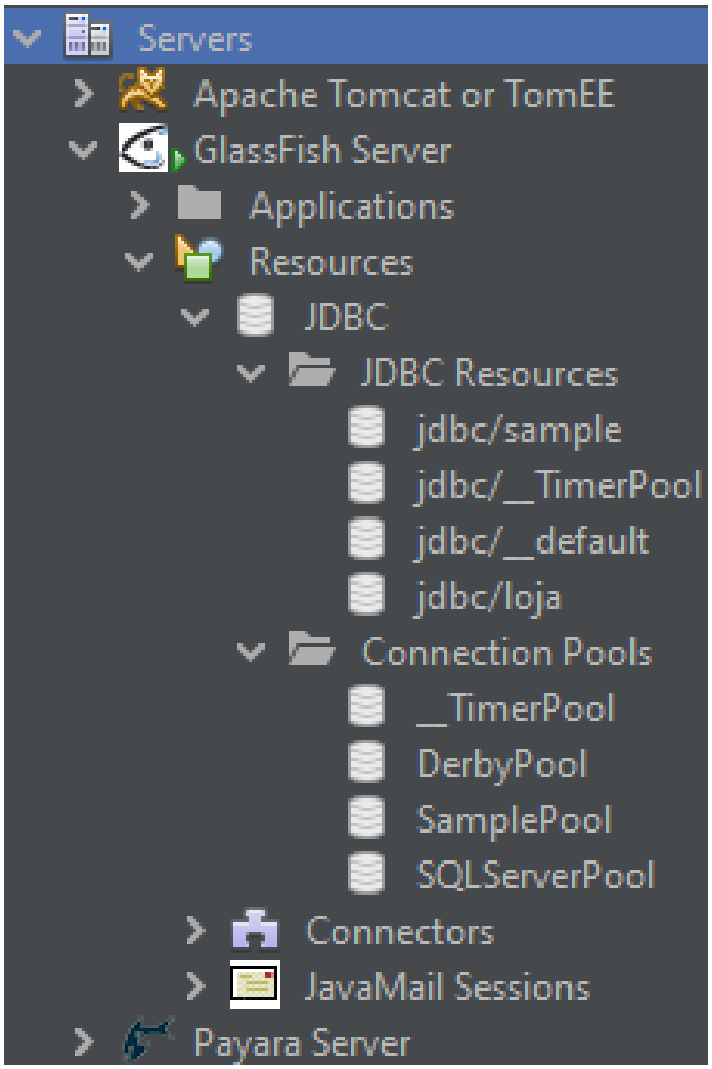
m. Testar o pool de conexões através do comando apresentado a seguir:

ping-connection-pool SQLServerPool

n. Obtendo sucesso na operação, criar o registro **JNDI**, ainda no **asadmin**, através do comando apresentado a seguir:

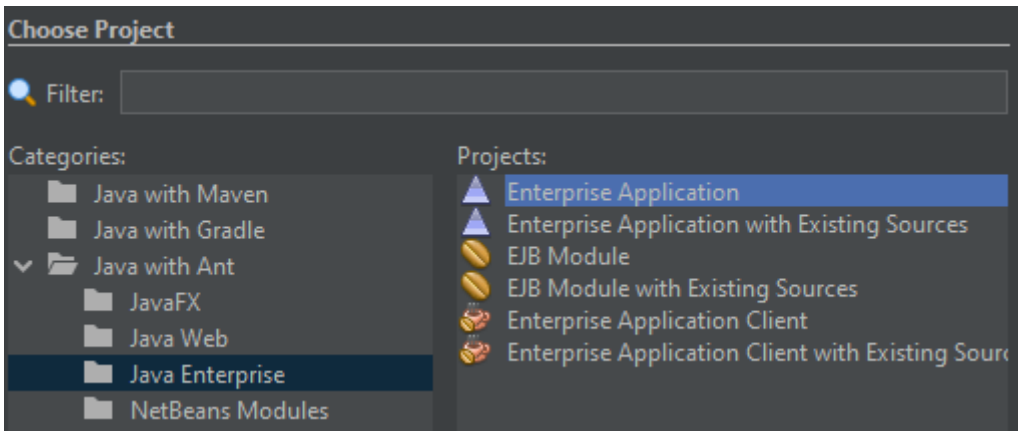
create-jdbc-resource --connectionpoolid SQLServerPool jdbc/loja

o. Atualizar o servidor no ambiente do NetBeans e verificar se tudo foi gerado corretamente



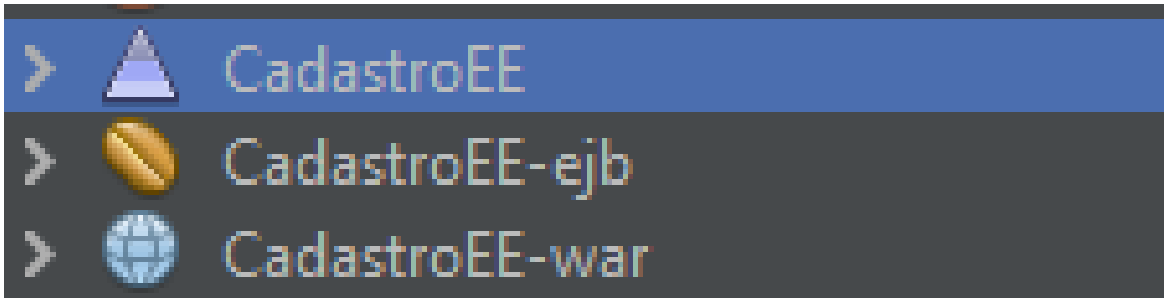
2. Criar o aplicativo corporativo no NetBeans:

a. Criar um projeto do tipo **Ant..Java Enterprise..Enterprise Application.**

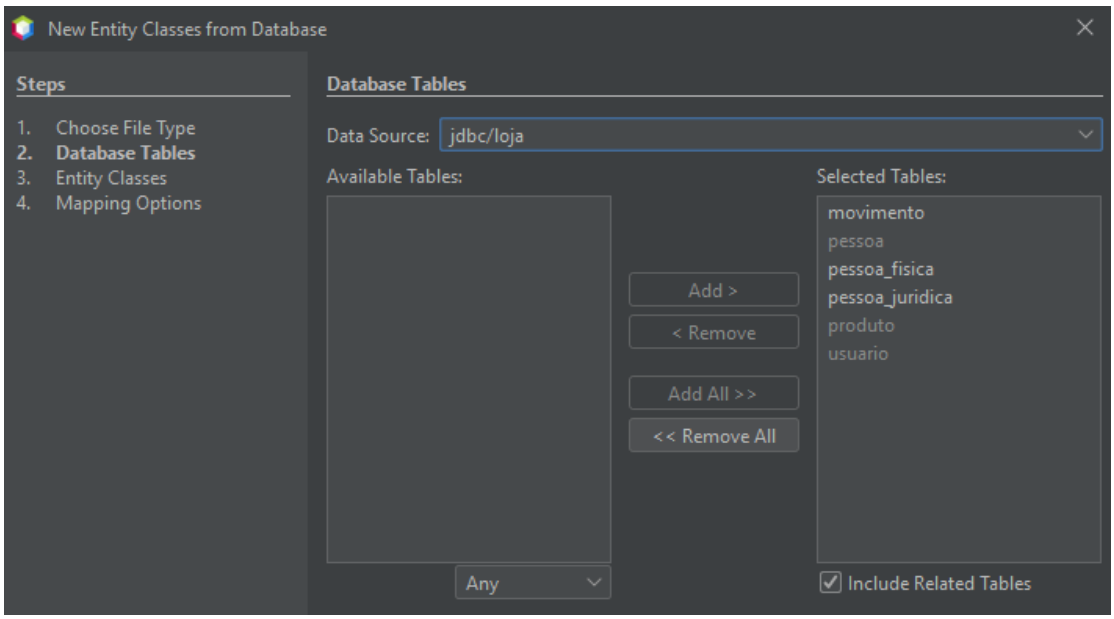


b. Adotar o nome **CadastroEE**, com escolha do servidor **GlassFish**, além de plataforma **Jakarta JEE 8.**

c. Serão gerados três projetos, onde o principal encapsula o arquivo EAR, tendo os outros dois, **CadastroEE-ejb** e **CadastroEE-war**, como projetos dependentes, relacionados aos elementos JPA, JEE e Web.



3. Definir as camadas de persistência e controle no projeto **CadastroEE-ejb.**
    1. Criar as **entidades JPA** através de **New Entity Classes from Database.**
    2. Selecionar **jdbc/loja** como **Data Source**, e selecionar todas as tabelas.



c. No passo seguinte, definir o pacote como **cadastroee.model**, além de marcar a opção para criação do arquivo **persistence.xml.**

d. Em seguida, adicionar os componentes EJB ao projeto, através da opção **New Session Beans for Entity Classes.**

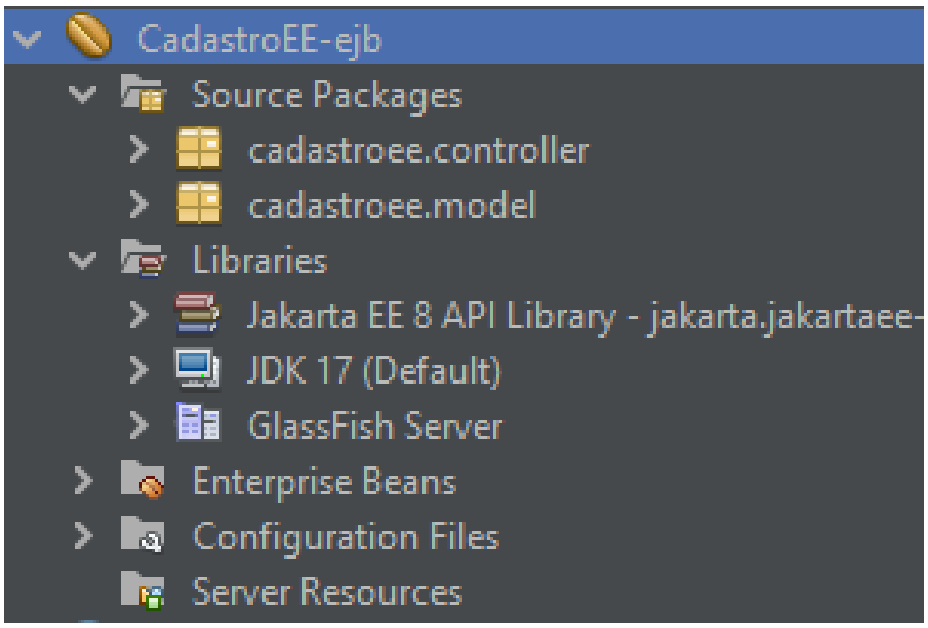
e. Selecionar **todas as entidades**, marcar a geração da **interface local**, além de definir o nome do pacote como **cadatroee.controller.**

f. Serão gerados todos os **Session Beans**, com o sufixo **Facade**, bem como as **interfaces**, com o sufixo **FacadeLocal.**

4. Efetuar pequenos acertos no projeto, para uso do Jakarta:

a. Adicionar a biblioteca **Jakarta EE 8 API** ao projeto **CadatroEE-ejb.**

b. Criados os componentes e ajustadas as bibliotecas, o projeto deverá ficar como apresentado a seguir.



c. Modificar **TODAS** as importações de pacotes **javax** para **jakarta**, em todos os arquivos do projeto **CadastroEE-ejb.**

```
5  package cadastroee.model;
6
7  import java.io.Serializable;
8  import java.util.Collection;
9  import jakarta.persistence.Basic;
10 import jakarta.persistence.CascadeType;
11 import jakarta.persistence.Column;
```

d. Na entidade **Produto**, mudar o tipo do atributo **precoVenda** para **Float** no lugar de BigDecimal.

e. Modificar o arquivo **persistence.xml** para o que é apresentado a seguir:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/
xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="CadastroEE-ejbPU" transaction-type="JTA">
    <jta-data-source>jdbc/loja</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties/>
  </persistence-unit>
```

</persistence>

5. Criar um **Servlet** de teste no projeto **CadastroEE-war**

   a. Utilizar o clique do botão direito e escolha da opção **New..Servlet**

   b. Definir o nome do Servlet como **ServletProduto**, e nome do pacote como **cadastroee.servlets**

   c. Marcar opção **Add information to deployment descriptor**, algo que ainda é necessário quando o GlassFish 6 é utilizado

   d. Adicionar, no código do Servlet, a referência para a interface do EJB

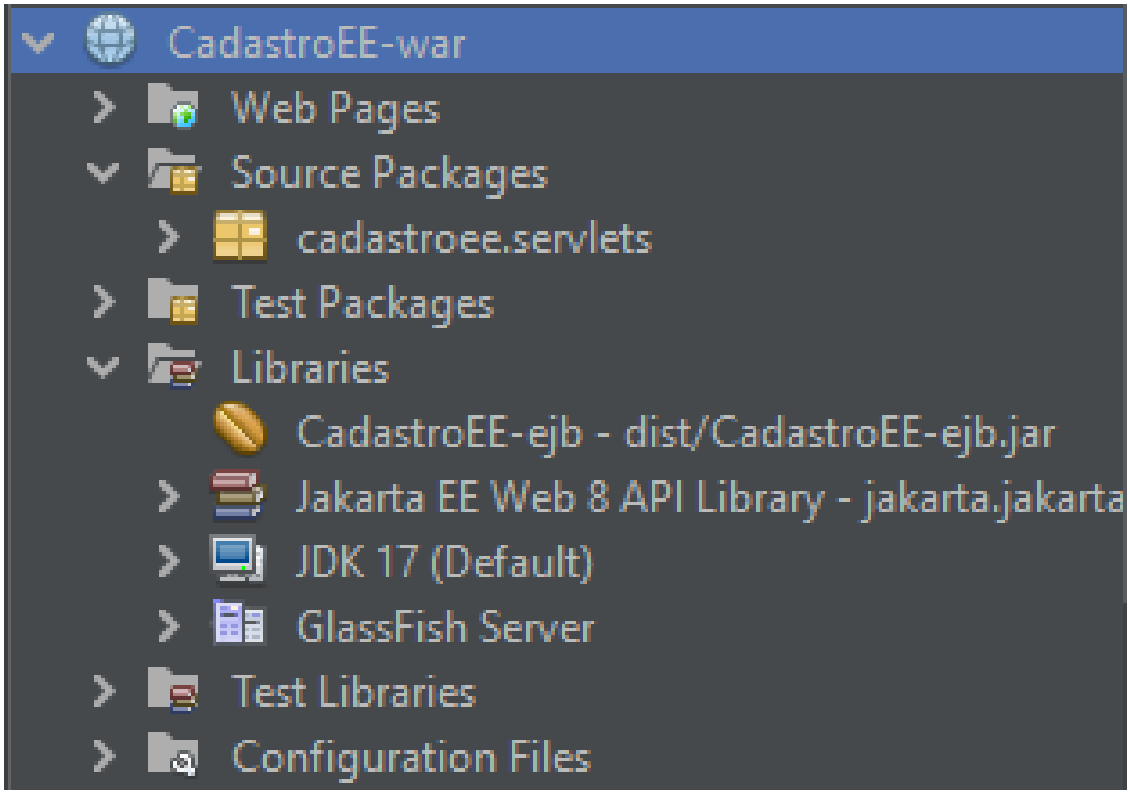
   @EJB

   ProdutoFacadeLocal facade;

   e. Modificar a resposta do Servlet, utilizando o **facade** para recuperar os dados e apresentá-los na forma de lista HTML

6. Efetuar novos acertos no projeto, para uso do Jakarta:

   1. Adicionar a biblioteca **Jakarta EE Web 8 API** ao projeto **CadatroEE-war**
   2. Criado o Servlet e ajustadas as bibliotecas, o projeto deverá ficar como
   3. apresentado a seguir:



c. Modificar **TODAS** as importações de pacotes **javax** para **jakarta**, em todos os arquivos do projeto **CadastroEE-war**

   d. Modificar o arquivo **web.xml** para o que é apresentado a seguir:

<?xml version="1.0" encoding="UTF-8"?>

```
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/
javaee/web-app_4_0.xsd">
 <servlet>
  <servlet-name>ServletProduto</servlet-name>
  <servlet-class>cadastroee.servlets.ServletProduto</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>ServletProdutoFC</servlet-name>
  <servlet-class>cadastroee.servlets.ServletProdutoFC
  </servlet-class>
 </servlet>
 <session-config>
  <session-timeout>30</session-timeout>
 </session-config>
</web-app>
```

7. Executar o projeto:
    1. A execução deve ser efetuar com o uso de **Run** ou **Deploy** no projeto principal (**CadastroEE**), simbolizado por um triângulo
    2. Acessar o endereço a seguir, para testar o Servlet**http://localhost:8080/CadastroEE-war/ServletProduto**
    3. Tendo alimentado a base via SQL Server Management Studio, ou pela aba de serviços do NetBeans, deve ser obtida uma saída como a seguinte:

✅ Resultados esperados

1. É importante que o código seja organizado.

2. Outro ponto importante é explorar as funcionalidades oferecidas pelo NetBeans para

melhoria da produtividade.

3. Nesse exercício, é esperado que o estudante demonstre as habilidades básicas para a construção de aplicativos Web na plataforma JEE.

📝 Relatório discente de acompanhamento

Os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo, nome dos integrantes da Prática. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação. A documentação do projeto deve conter:

1. Título da Prática;
2. Objetivo da Prática;
3. Todos os códigos solicitados neste roteiro de aula;
4. Os resultados da execução dos códigos também devem ser apresentados;
5. Análise e Conclusão:
    1. Como é organizado um projeto corporativo no NetBeans?
    2. Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?
    3. Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?
    4. O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?
    5. Como é feita a comunicação entre os Serlvets e os Session Beans do pool de EJBs?

👉 2º Procedimento | Interface Cadastral com Servlet e JSPs

1. Criar um Servlet com o nome **ServletProdutoFC**, no projeto **CadastroEE-war:**
    1. Utilizar o padrão **Front Controller**
    2. Adicionar uma referência para **ProdutoFacadeLocal**, utilizando o nome **facade** para o atributo
    3. Apagar o conteúdo interno do método **processRequest**, e efetuar nele as modificações seguintes
    4. Capturar o parâmetro **acao** a partir do **request**, o qual poderá assumir os valores

       listar, **incluir**, **alterar**, **excluir**, **formIncluir** e **formAlterar**

5. Definir a variável **destino**, contendo o nome do JSP de apresentação, que terá os

   valores **ProdutoDados.jsp**, para acao valendo formAlterar ou formIncluir, ou

   ProdutoLista.jsp, para as demais opções
6. Para o valor **listar**, adicionar a **listagem de produtos** como atributo da requisição

   (**request**), com a consulta efetuada via facade
7. Para o valor **formAlterar**, capturar o **id** fornecido como parâmetro do request,

   consultar a **entidade** via facade, e adicioná-la como atributo da requisição

   (**request**)
8. Para o valor **excluir**, capturar o **id** fornecido como parâmetro do request, remover

   a entidade através do facade, e adicionar a **listagem de produtos** como atributo

   da requisição (**request**)
9. Para o valor **alterar**, capturar o **id** fornecido como parâmetro do request, consultar

   a entidade através do facade, preencher os demais campos com os valores

   fornecidos no request, alterar os dados via facade e adicionar a **listagem de**

   produtos como atributo da requisição (**request**)
10. Para o valor **incluir**, instanciar uma entidade do tipo Produto, preencher os

    campos com os valores fornecidos no request, inserir via facade e adicionar a

    listagem de produtos como atributo da requisição (**request**)
11. Ao final redirecionar para **destino** via **RequestDispatcher**, obtido a partir do

    objeto request

2. Criar a página de consulta, com o nome **ProdutoLista.jsp**
    1. Incluir um link para **ServletProdutoFC**, com acao **formIncluir**, voltado para a

       abertura do formulário de inclusão.
    2. Definir uma tabela para apresentação dos dados.
    3. Recuperar a **lista de produtos** enviada pelo Servlet.
    4. Para cada elemento da lista, apresentar id, nome, quantidade e preço como

       células da tabela.
    5. Criar, também, de forma dinâmica, links para **alteração** e **exclusão**, com a

       chamada para ServletProdutoFC, passando as ações corretas e o id do elemento

       corrente.

6. Organizar o código para obter uma página como a seguinte.

3. Criar a página de cadastro, com o nome

ProdutoDados.jsp

1. Definir um formulário com envio para
2. **ServletProdutoFC**, modo **post.**
3. Recuperar a **entidade** enviada pelo Servlet.
4. Definir a variável **acao**, com valor **incluir**, para entidade
5. nula, ou **alterar**, quando a entidade é fornecida.
6. Incluir um campo do tipo **hidden**, para envio do valor
7. de **acao** .
8. Incluir um campo do tipo **hidden**, para envio do **id**,
9. apenas quando o valor de **acao** for **alterar.**
10. Incluir os campos para nome, quantidade e preço de
11. venda, preenchendo os dados quando a entidade é
12. fornecida.
13. Concluir o formulário com um botão de envio, com o
14. texto adequado para as situações de inclusão ou
15. alteração de dados .
16. Organizar o código para obter uma página como a
17. seguinte.

4. Testar as funcionalidades do sistema:
    1. Listar os produtos com a chamada para o endereço seguinte: **http://localhost:8080/CadastroEE-war/ServletProdutoFC?acao=listar**
    2. Efetuar uma inclusão a partir do link da tela de listagem
    3. Efetuar uma alteração a partir do link dinâmico da listagem
    4. Efetuar uma exclusão a partir do link dinâmico da Listagem

✅ Resultados esperados

1. É importante que o código seja organizado.
2. Outro ponto importante é explorar as funcionalidades oferecidas pelo NetBeans para melhoria da produtividade.
3. Nesse exercício, é esperado que o estudante demonstre habilidade para utilizar Servlets e JSPs na construção de interfaces cadastrais para Web.

📝 Relatório discente de acompanhamento

Os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo, nome dos integrantes da Prática.
Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação. A documentação do projeto deve conter:

1. Título da Prática;
2. Objetivo da Prática;
3. Todos os códigos solicitados neste roteiro de aula;
4. Os resultados da execução dos códigos também devem
5. ser apresentados;
6. Análise e Conclusão:
    1. Como funciona o padrão Front Controller, e como ele é
    2. implementado em um aplicativo Web Java, na

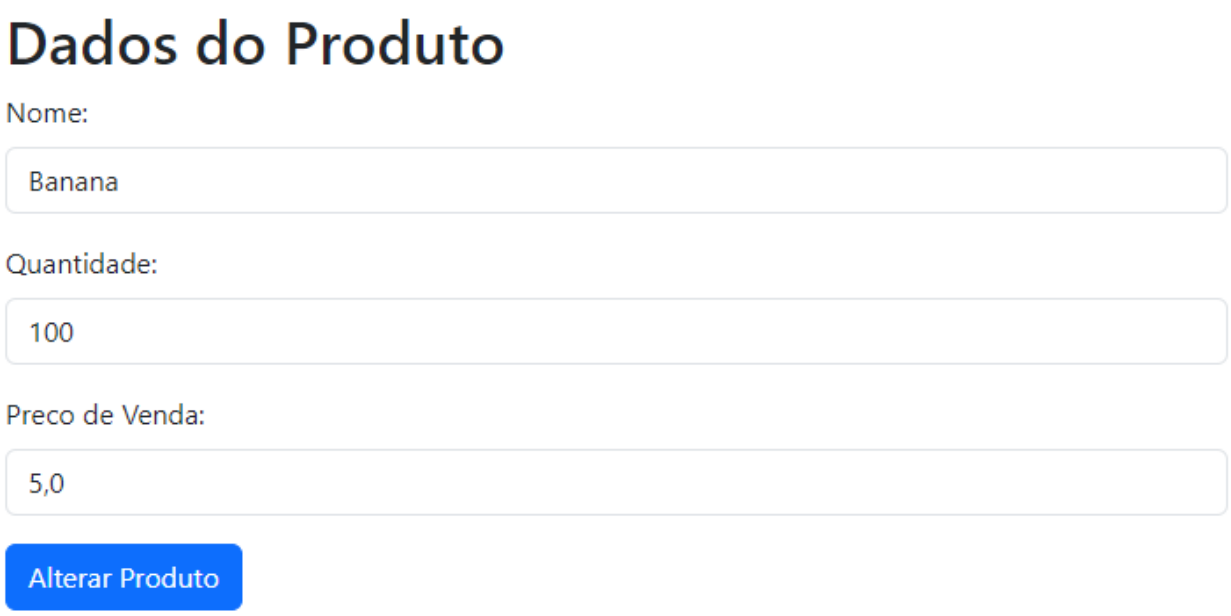
3. arquitetura MVC?
4. Quais as diferenças e semelhanças entre Servlets e
5. JSPs?
6. Qual a diferença entre um redirecionamento simples e
7. o uso do método forward, a partir do
8. RequestDispatcher? Para que servem parâmetros e
9. atributos nos objetos HttpRequest?

- 👉 **3º Procedimento | Melhorando o Design da Interface**
- Incluir as bibliotecas do framework **Bootstrap** nos
- arquivos **ProdutoLista.jsp e ProdutoDados.jsp**
- Visitar o site do **BootStrap**, no endereço **https://**
- **getbootstrap.com/**
- Rolar para baixo até encontrar a inclusão via **CDN**
- 

- c. Clicar no botão para cópia do link **CSS** e colar na
- divisão **head** de cada uma das páginas JSP.
- d. Clicar no botão para cópia do link para a biblioteca
- **Java Script** e colar na divisão **head** de cada uma das
- páginas JSP
- 2. Modificar as características de **ProdutoLista.jsp**
- Adicionar a classe **container** ao body.
- Adicionar as classes **btn**, **btn-primary** e **m-2** no link de
- inclusão.
- Adicionar as classes **table** e **table-striped** na tabela.
- Adicionar a classe **table-dark** ao thead.
- Adicionar as classes **btn**, **btn-primary** e **btn-sm** no link
- de alteração.

- Adicionar as classes **btn**, **btn-danger** e **btn-sm** no link
- de exclusão.
- Ajustar as características para obter o design
- apresentado a seguir.

3. Modificar as características de **ProdutoDados.jsp**
    1. Adicionar a classe **container** ao body.
    2. Encapsule cada par label / input em **div** com classe
    3. **mb-3.**
    4. Adicionar a classe **form** ao formulário.
    5. Adicionar a classe **form-label** em cada label.
    6. Adicionar a classe **form-control** em cada input.
    7. Adicionar as classes **btn** e **btn-primary** ao botão de
    8. inclusão.
    9. Ajustar as características para obter o design
    10. apresentado a seguir.



Material de **orientações** para

desenvolvimento da **missão prática** do **4º**

nível de conhecimento.

RPG0017 - Vamos integrar sistemas

Implementação de sistema cadastral com interface Web,

baseado nas tecnologias de Servlets, JPA e JEE.

Objetivos da prática

1. Implementar persistência com base em JPA.
2. Implementar regras de negócio na plataforma JEE,
3. através de EJBs.
4. Implementar sistema cadastral Web com base em
5. Servlets e JSPs.
6. Utilizar a biblioteca Bootstrap para melhoria do design.
7. No final do exercício, o aluno terá criado todos os
8. elementos necessários para exibição e entrada de dados
9. na plataforma Java Web, tornando-se capacitado para
10. lidar com contextos reais de aplicação.

🎗 **As práticas devem ser feitas individualmente.**

Materiais necessários para a prática

- SQL Server, com o banco de dados gerado em prática
- anterior (loja).

- JDK e IDE NetBeans.
- Navegador para Internet, como o Chrome.
- Banco de dados SQL Server com o Management Studio.
- Equipamentos:

- Computador com acesso à Internet.
- JDK e IDE NetBeans.
- Banco de dados SQL Server.
- Navegador de Internet instalado no computador.

Desenvolvimento da prática

Vamos colocar a **mão na massa**?! Siga as **instruções** abaixo para desenvolvimento desta missão.

👉 **1º Procedimento | Camadas de Persistência e Controle**

1. Configurar a conexão com SQL Server via NetBeans e o
2. pool de conexões no GlassFish Server 6.2.1:
    1. Na aba de **Serviços**, divisão **Banco de Dados**, clique
    2. com o botão direito em **Drivers** e escolha **Novo Driver.**

b. Na janela que se abrirá, clicar em **Add** (Adicionar), selecionar o arquivo **mssql-jdbc-12.2.0.jre8.jar**, que é parte do arquivo **zip** encontrado no endereço seguinte, e finalizar com **Ok**

https://learn.microsoft.com/pt-br/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16

c. O reconhecimento será automático, e podemos definir uma conexão com o clique do botão direito sobre o driver e escolha de **Conectar Utilizando.**

d. Para os campos **database**, **user** e **password**, utilizar o valor **loja**, de acordo com os elementos criados em exercício anterior sobre a criação do banco de dados de exemplo, marcando também a opção **Lembrar Senha.**

e. Para o campo JDBC URL deve ser utilizada a seguinte expressão:

jdbc:sqlserver://

localhost:1433;databaseName=loja;encrypt=true;trustSer

verCertificate=true;

f. Clicar em **Testar Conexão** e, estando tudo certo, **Finalizar.**

g. Na divisão **Servidores**, verificar se o GlassFish 6.2.1 (ou posterior) está instalado, e caso não esteja, adicionar o servidor, via clique com o botão direito e escolha da opção **Add Server**, efetuando o download a partir da própria janela que se abrirá

h. Copiar o arquivo **mssql-jdbc-12.2.0.jre8.jar** para o subdiretório **lib**, a partir do diretório de base do **GlassFish.**

i. Iniciar o servidor GlassFish a partir do NetBeans.

j. Através da linha de comando, executar o comando **asadmin**, no diretório **bin** do **GlassFish.**

k. No **prompt do asadmin**, executar o comando apresentado a seguir

create-jdbc-connection-pool

--datasourceclassname

com.microsoft.sqlserver.jdbc.SQLServerDataSource

--restype javax.sql.DataSource

--property

driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriv

er:portNumber=1433:password=loja:user=loja:serverNa

me=localhost:databaseName=loja:trustServerCertificate

=true:URL="jdbc\\:sqlserver\\://localhost\\:1433

\;databaseName\\=loja\\;encrypt\\=true

\;trustServerCertificate\\=true\\;"

l. Será solicitado o identificador do pool, que será **SQLServerPool.**

m. Testar o pool de conexões através do comando apresentado a seguir:

ping-connection-pool SQLServerPool

n. Obtendo sucesso na operação, criar o registro **JNDI**,

ainda no **asadmin**, através do comando apresentado a seguir:

create-jdbc-resource --connectionpoolid SQLServerPool jdbc/loja

o. Atualizar o servidor no ambiente do NetBeans e verificar se tudo foi gerado corretamente

2. Criar o aplicativo corporativo no NetBeans:

a. Criar um projeto do tipo **Ant..Java Enterprise..Enterprise Application.**

b. Adotar o nome **CadastroEE**, com escolha do servidor **GlassFish**, além de plataforma **Jakarta JEE 8.**

c. Serão gerados três projetos, onde o principal encapsula o arquivo EAR, tendo os outros dois, **CadastroEE-ejb** e **CadastroEE-war**, como projetos dependentes, relacionados aos elementos JPA, JEE e Web.

3. Definir as camadas de persistência e controle no projeto **CadastroEE-ejb.**

1. Criar as **entidades JPA** através de **New Entity Classes**
2. **from Database.**
3. Selecionar **jdbc/loja** como **Data Source**, e selecionar
4. todas as tabelas.

c. No passo seguinte, definir o pacote como **cadastroee.model**, além de marcar a opção para criação do arquivo **persistence.xml.**

d. Em seguida, adicionar os componentes EJB ao projeto, através da opção **New Session Beans for Entity Classes.**

e. Selecionar **todas as entidades**, marcar a geração da **interface local**, além de definir o nome do pacote como **cadatroee.controller.**

f. Serão gerados todos os **Session Beans**, com o sufixo **Facade**, bem como as **interfaces**, com o sufixo **FacadeLocal.**

4. Efetuar pequenos acertos no projeto, para uso do Jakarta:

a. Adicionar a biblioteca **Jakarta EE 8 API** ao projeto **CadatroEE-ejb.**

b. Criados os componentes e ajustadas as bibliotecas, o projeto deverá ficar como apresentado a seguir.

c. Modificar **TODAS** as importações de pacotes **javax** para **jakarta**, em todos os arquivos do projeto **CadastroEE-ejb.**

d. Na entidade **Produto**, mudar o tipo do atributo **precoVenda** para **Float** no lugar de BigDecimal.

e. Modificar o arquivo **persistence.xml** para o que é apresentado a seguir:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0"
xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/
persistence http://java.sun.com/xml/ns/persistence/
persistence_1_0.xsd">
  <persistence-unit name="CadastroEE-ejbPU"
transaction-type="JTA">
    <jta-data-source>jdbc/loja</jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-
classes>
    <properties/>
  </persistence-unit>
</persistence>
```

5. Criar um **Servlet** de teste no projeto **CadastroEE-war**

a. Utilizar o clique do botão direito e escolha da opção **New..Servlet**

b. Definir o nome do Servlet como **ServletProduto**, e nome do pacote como **cadastroee.servlets**

c. Marcar opção **Add information to deployment descriptor**, algo que ainda é necessário quando o GlassFish 6 é utilizado

d. Adicionar, no código do Servlet, a referência para a interface do EJB

@EJB

ProdutoFacadeLocal facade;

e. Modificar a resposta do Servlet, utilizando o **facade** para recuperar os dados e apresentá-los na forma de lista HTML

6. Efetuar novos acertos no projeto, para uso do Jakarta:

    1. Adicionar a biblioteca **Jakarta EE Web 8 API** ao projeto
    2. **CadatroEE-war**
    3. Criado o Servlet e ajustadas as bibliotecas, o projeto
    4. deverá ficar como apresentado a seguir:

c. Modificar **TODAS** as importações de pacotes **javax** para **jakarta**, em todos os arquivos do projeto **CadastroEE-war**

d. Modificar o arquivo **web.xml** para o que é apresentado a seguir:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/
ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">
 <servlet>
  <servlet-name>ServletProduto</servlet-name>
  <servlet-class>cadastroee.servlets.ServletProduto</
servlet-class>
 </servlet>
 <servlet>
  <servlet-name>ServletProdutoFC</servlet-name>
  <servlet-class>cadastroee.servlets.ServletProdutoFC
  </servlet-class>
 </servlet>
 <session-config>
```

```
  <session-timeout>30</session-timeout>
 </session-config>
</web-app>
```

7. Executar o projeto:
    1. A execução deve ser efetuar com o uso de **Run** ou
    2. **Deploy** no projeto principal (**CadastroEE**), simbolizado
    3. por um triângulo
    4. Acessar o endereço a seguir, para testar o
    5. Servlet**http://localhost:8080/CadastroEE-war/**
    6. **ServletProduto**
    7. Tendo alimentado a base via SQL Server Management
    8. Studio, ou pela aba de serviços do NetBeans, deve ser
    9. obtida uma saída como a seguinte:

✅ Resultados esperados

1. É importante que o código seja organizado.
2. Outro ponto importante é explorar as funcionalidades oferecidas pelo NetBeans para melhoria da produtividade.
3. Nesse exercício, é esperado que o estudante demonstre as habilidades básicas para a construção de aplicativos Web na plataforma JEE.

📝 Relatório discente de acompanhamento

Os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo, nome dos integrantes da Prática.
Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação. A documentação do projeto deve conter:

1. Título da Prática;
2. Objetivo da Prática;
3. Todos os códigos solicitados neste roteiro de aula;
4. Os resultados da execução dos códigos também devem
5. ser apresentados;

6. Análise e Conclusão:
    1. Como é organizado um projeto corporativo no
    2. NetBeans?
    3. Qual o papel das tecnologias JPA e EJB na construção
    4. de um aplicativo para a plataforma Web no ambiente
    5. Java?
    6. Como o NetBeans viabiliza a melhoria de produtividade
    7. ao lidar com as tecnologias JPA e EJB?
    8. O que são Servlets, e como o NetBeans oferece
    9. suporte à construção desse tipo de componentes em
    10. um projeto Web?
    11. Como é feita a comunicação entre os Serlvets e os
    12. Session Beans do pool de EJBs?

👉 2º Procedimento | Interface Cadastral com Servlet e JSPs

1. Criar um Servlet com o nome **ServletProdutoFC**, no
2. projeto **CadastroEE-war:**
    1. Utilizar o padrão **Front Controller**
    2. Adicionar uma referência para **ProdutoFacadeLocal**,
    3. utilizando o nome **facade** para o atributo
    4. Apagar o conteúdo interno do método
    5. **processRequest**, e efetuar nele as modificações
    6. seguintes
    7. Capturar o parâmetro **acao** a partir do **request**, o qual
    8. poderá assumir os valores **listar**, **incluir**, **alterar**,
    9. **excluir**, **formIncluir** e **formAlterar**
    10. Definir a variável **destino**, contendo o nome do JSP de
    11. apresentação, que terá os valores **ProdutoDados.jsp**,
    12. para acao valendo formAlterar ou formIncluir, ou
    13. **ProdutoLista.jsp**, para as demais opções
    14. Para o valor **listar**, adicionar a **listagem de produtos**
    15. como atributo da requisição (**request**), com a consulta
    16. efetuada via facade

17. Para o valor **formAlterar**, capturar o **id** fornecido como
18. parâmetro do request, consultar a **entidade** via facade,
19. e adicioná-la como atributo da requisição (**request**)
20. Para o valor **excluir**, capturar o **id** fornecido como
21. parâmetro do request, remover a entidade através do
22. facade, e adicionar a **listagem de produtos** como
23. atributo da requisição (**request**)
24. Para o valor **alterar**, capturar o **id** fornecido como
25. parâmetro do request, consultar a entidade através do
26. facade, preencher os demais campos com os valores
27. fornecidos no request, alterar os dados via facade e
28. adicionar a **listagem de produtos** como atributo da
29. requisição (**request**)
30. Para o valor **incluir**, instanciar uma entidade do tipo
31. Produto, preencher os campos com os valores
32. fornecidos no request, inserir via facade e adicionar a
33. **listagem de produtos** como atributo da requisição
34. (**request**)
35. Ao final redirecionar para **destino** via
36. **RequestDispatcher**, obtido a partir do objeto request

2. Criar a página de consulta, com o nome

ProdutoLista.jsp

1. Incluir um link para **ServletProdutoFC**, com acao
2. **formIncluir**, voltado para a abertura do formulário de
3. inclusão.
4. Definir uma tabela para apresentação dos dados.
5. Recuperar a **lista de produtos** enviada pelo Servlet.
6. Para cada elemento da lista, apresentar id, nome,
7. quantidade e preço como células da tabela.
8. Criar, também, de forma dinâmica, links para **alteração**
9. e **exclusão**, com a chamada para ServletProdutoFC,
10. passando as ações corretas e o id do elemento
11. corrente.
12. Organizar o código para obter uma página como a

13. seguinte.

3. Criar a página de cadastro, com o nome

ProdutoDados.jsp

1. Definir um formulário com envio para
2. **ServletProdutoFC**, modo **post.**
3. Recuperar a **entidade** enviada pelo Servlet.
4. Definir a variável **acao**, com valor **incluir**, para entidade
5. nula, ou **alterar**, quando a entidade é fornecida.
6. Incluir um campo do tipo **hidden**, para envio do valor
7. de **acao** .
8. Incluir um campo do tipo **hidden**, para envio do **id**,
9. apenas quando o valor de **acao** for **alterar.**
10. Incluir os campos para nome, quantidade e preço de
11. venda, preenchendo os dados quando a entidade é
12. fornecida.
13. Concluir o formulário com um botão de envio, com o
14. texto adequado para as situações de inclusão ou
15. alteração de dados .
16. Organizar o código para obter uma página como a
17. seguinte.

4. Testar as funcionalidades do sistema:
    1. Listar os produtos com a chamada para o endereço
    2. seguinte: **http://localhost:8080/CadastroEE-war/**
    3. **ServletProdutoFC?acao=listar**
    4. Efetuar uma inclusão a partir do link da tela de
    5. listagem
    6. Efetuar uma alteração a partir do link dinâmico da
    7. listagem
    8. Efetuar uma exclusão a partir do link dinâmico da
    9. listagem

✅ **Resultados esperados**

1. É importante que o código seja organizado.

2. Outro ponto importante é explorar as funcionalidades

oferecidas pelo NetBeans para melhoria da produtividade.

3. Nesse exercício, é esperado que o estudante demonstre habilidade para utilizar Servlets e JSPs na construção de interfaces cadastrais para Web.

📝 **Relatório discente de acompanhamento**

Os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo, nome dos integrantes da Prática. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação. A documentação do projeto deve conter:

1. Título da Prática;
2. Objetivo da Prática;
3. Todos os códigos solicitados neste roteiro de aula;
4. Os resultados da execução dos códigos também devem
5. ser apresentados;
6. Análise e Conclusão:
    1. Como funciona o padrão Front Controller, e como ele é
    2. implementado em um aplicativo Web Java, na
    3. arquitetura MVC?
    4. Quais as diferenças e semelhanças entre Servlets e
    5. JSPs?
    6. Qual a diferença entre um redirecionamento simples e
    7. o uso do método forward, a partir do
    8. RequestDispatcher? Para que servem parâmetros e
    9. atributos nos objetos HttpRequest?

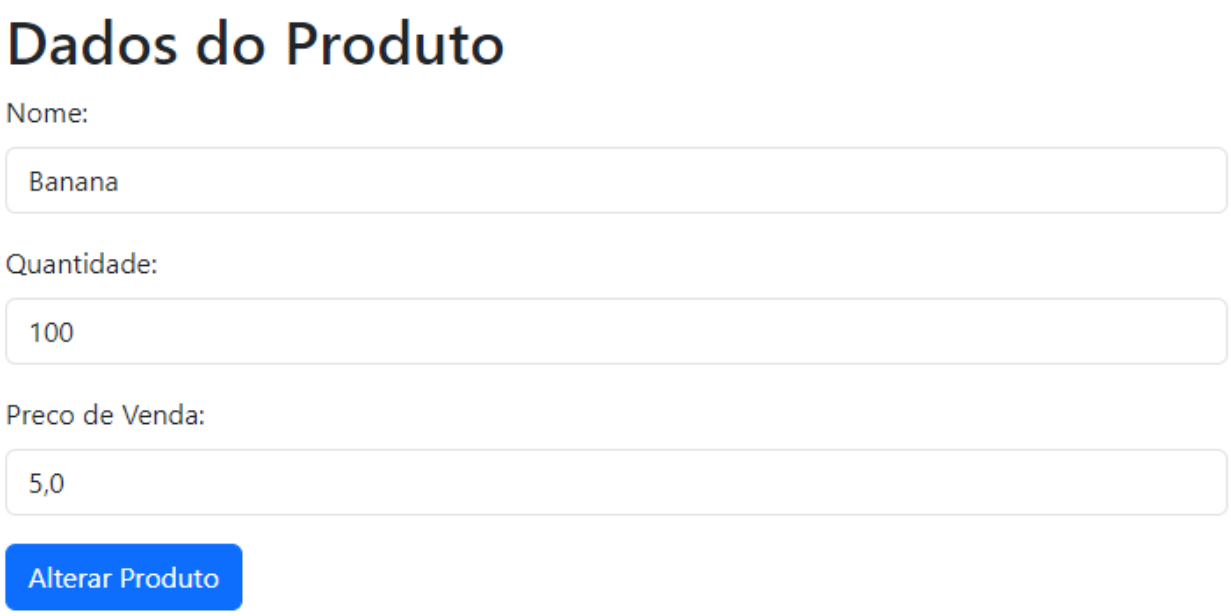
👉 **3º Procedimento | Melhorando o Design da Interface**

1. Incluir as bibliotecas do framework **Bootstrap** nos
2. arquivos **ProdutoLista.jsp e ProdutoDados.jsp**
    1. Visitar o site do **BootStrap**, no endereço **https://**
    2. **getbootstrap.com/**
    3. Rolar para baixo até encontrar a inclusão via **CDN**

c. Clicar no botão para cópia do link **CSS** e colar na divisão **head** de cada uma das páginas JSP.

d. Clicar no botão para cópia do link para a biblioteca **Java Script** e colar na divisão **head** de cada uma das páginas JSP

2. Modificar as características de **ProdutoLista.jsp**
    1. Adicionar a classe **container** ao body.
    2. Adicionar as classes **btn**, **btn-primary** e **m-2** no link de
    3. inclusão.
    4. Adicionar as classes **table** e **table-striped** na tabela.
    5. Adicionar a classe **table-dark** ao thead.
    6. Adicionar as classes **btn**, **btn-primary** e **btn-sm** no link
    7. de alteração.
    8. Adicionar as classes **btn**, **btn-danger** e **btn-sm** no link
    9. de exclusão.
    10. Ajustar as características para obter o design
    11. apresentado a seguir.
3. Modificar as características de **ProdutoDados.jsp**
    1. Adicionar a classe **container** ao body.
    2. Encapsule cada par label / input em **div** com classe
    3. **mb-3.**
    4. Adicionar a classe **form** ao formulário.
    5. Adicionar a classe **form-label** em cada label.
    6. Adicionar a classe **form-control** em cada input.
    7. Adicionar as classes **btn** e **btn-primary** ao botão de
    8. inclusão.
    9. Ajustar as características para obter o design
    10. apresentado a seguir.



✅ Resultados esperados

1. É importante que o código seja organizado.
2. Nesse exercício, é esperado que o estudante demonstre habilidade para incluir o framework Bootstrap e utilizá-lo na melhoria do design.

📝 Relatório discente de acompanhamento

Os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a Logo da Universidade, nome do Campus, nome do Curso, nome da Disciplina, número da Turma, semestre letivo, nome dos integrantes da Prática. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação. A documentação do projeto deve conter:

1. Título da Prática;
2. Objetivo da Prática;
3. Todos os códigos solicitados neste roteiro de aula;
4. Os resultados da execução dos códigos também devem
5. ser apresentados;
6. Análise e Conclusão:
    1. Como o framework Bootstrap é utilizado?
    2. Por que o Bootstrap garante a independência estrutural

3. do HTML?
4. Qual a relação entre o Boostrap e a responsividade da
5. página?

Observações

Pré-requisitos:

1. Os estudantes precisam instalar o JDK e o NetBeans;
2. Também é necessário instalar o SQL Server e criar o
3. banco de dados que foi solicitado na **Prática 2 – Vamos**
4. **manter as informações**;
5. Acesso à Internet para utilizar o repositório CDN do
6. Bootstrap.

Referência Bibliográfica:

1. https://stensineme.blob.core.windows.net/
2. hmlgrepoh/00212ti/00905/index.html
3. https://stensineme.blob.core.windows.net/
4. hmlgrepoh/00212ti/00965/index.html
5. https://stensineme.blob.core.windows.net/
6. hmlgrepoh/00212ti/01678/index.html
7. Introdução ao middleware JDBC pela Dev Media.
8. Disponível no endereço https://www.devmedia.com.br/
9. jdbc-tutorial/6638. Acessado em 01/03/2023.
10. Introdução à Java Persistence API pela Dev Media.
11. Disponível no endereço https://www.devmedia.com.br/
12. introducao-a-jpa-java-persistence-api/28173. Acessado
13. em 10/03/2023.
14. Introdução aos EJBs produzido pela Geek for Geeks.
15. Disponível no endereço https://www.geeksforgeeks.org/
16. enterprise-java-beans-ejb/. Acessado em 10/03/2023.
17. Bootstrap Docs. Disponível em https://getbootstrap.com/
18. docs/5.0/getting-started/introduction/. Acessado em
19. 10/03/2023.

Entrega da prática

Chegou a hora, gamer!

✍ Armazene o projeto em um repositório no GIT.

✍ Anexar a documentação do projeto (PDF) no GIT.

✍ Compartilhe o link do repositório do GIT com o seu tutor para correção da prática, por meio da **Sala de Aula Virtual**, na aba "**Trabalhos**" do respectivo nível de conhecimento.

✍ **Ei, não se esqueça de entregar este trabalho na data estipulada no calendário acadêmico!**