



(✓) Curso Técnico

Quem Investe no futuro faz QI

Informática para Internet

Desenvolvimento de Sistemas

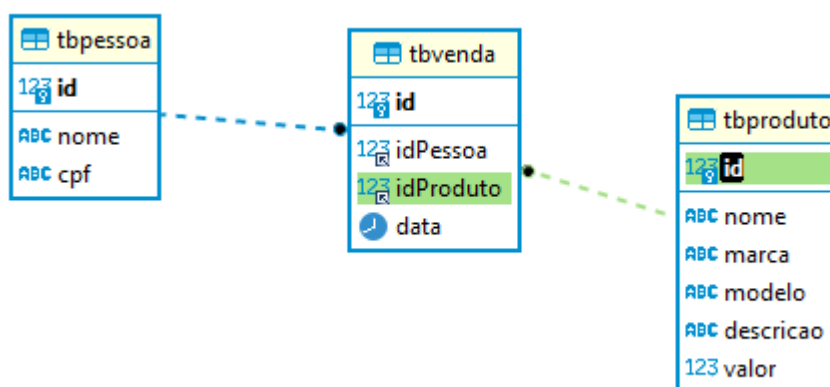
Web III

Unidade VII

1. CRUD

CRUD, acrônimo de Create, Read/Retrieve, Update and Delete em língua inglesa ou , são as quatro operações básicas de manipulação de dados/registros em um banco de dados relacionais, ou seja, inserção, consulta, atualização e exclusão de dados/registros. Não é exagero dizer que a maior parte dos sistemas empresariais detém um CRUD.

No desenvolvimento de um sistema Java web que haja a necessidade de realizar o armazenamento de dados em um banco de dados, a rotina abaixo poderá ser adotada para implementação do armazenamento, considerando a existência de um banco de dados chamado dbaula que contém a seguinte estrutura:



O sistema permitirá a inserção ou cadastro de pessoas, produtos e o registro de venda dos produtos para as pessoas, com o uso do modelo de relacionamento. O Código SQL para criação do banco de dados do diagrama acima é:






```
CREATE DATABASE dbaula;
use dbaula1;
CREATE TABLE `tbpessoa` (
  `id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  `nome` varchar(70) NOT NULL,
  `cpf` varchar(14) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

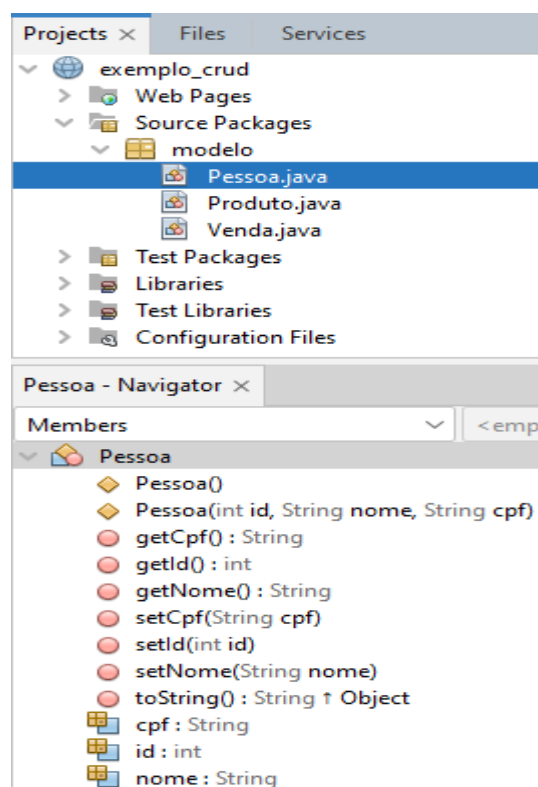
CREATE TABLE `tbproduto` (
  `id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  `nome` varchar(80) NOT NULL,
  `marca` varchar(30) NOT NULL,
  `modelo` varchar(50) NOT NULL,
  `descricao` varchar(300) NOT NULL,
```

```
`valor` double NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `tbvenda` (
  `id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
  `idPessoa` int(11) NOT NULL,
  `idProduto` int(11) NOT NULL,
  `data` timestamp NOT NULL DEFAULT current_timestamp(),
  CONSTRAINT `fkpessoa` FOREIGN KEY (`idPessoa`) REFERENCES `tbpessoa` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fkproduto` FOREIGN KEY (`idProduto`) REFERENCES `tbproduto`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Após a criação do banco de dados, um projeto em um IDE deverá ser criado para a codificação Java necessária. O projeto Java deverá ter o pacote modelo e neste pacote as classes Pessoa, Produto e Venda serão criadas, observando os atributos das tabelas do banco de dados, conforme o pormenoriza-se.

-  Pacote
-  Arquivo contendo o código
-  Construtores
-  Métodos
-  Atributos



O arquivo chamado Pessoa.java terá o seguinte código:

```
package modelo; //pacote da classe
public class Pessoa { //início da classe
  //início da declaração dos atributos
  private String nome, cpf; int id;
  //fim da declaração dos atributos
  //início do construtor vazio da classe
  public Pessoa() { }
  //fim do construtor vazio
```

```
//início do construtor cheio da classe
public Pessoa(int id, String nome, String cpf) {
    this.id = id;
    this.nome = nome;
    this.cpf = cpf;
}
//fim do construtor cheio
//início do método toString(transforma os objetos em String)
@Override //indica que o método é herdado de uma superclasse e será
sobrescrito
public String toString() {
    return "Código:" + id + ", nome:" + nome + "e CPF:" + cpf ;
}
//fim do método toString
//início do método getId (é sem parâmetro e com retorno do tipo int)
public int getId() { //permite visualizar o valor do atributo id do objeto
    return id;
}
//fim do método getId
//início do método setId (é com parâmetro e sem retorno)
public void setId(int id) { //permite alterar o valor do atributo id do
objeto
    this.id = id;
}
//fim do método setId
//início do método getNome (é sem parâmetro e com retorno do tipo String)
public String getNome() { //permite visualizar o valor do atributo nome do
objeto
    return nome;
}
//fim do método getNome
//início do método setNome (é com parâmetro e sem retorno)
public void setNome(String nome) { //permite alterar o valor do atributo
nome do objeto
    this.nome = nome;
}
//fim do método setNome
//início do método getCpf (é sem parâmetro e com retorno do tipo String)
public String getCpf() { //permite visualizar o valor do atributo cpf do
objeto
    return cpf;
}
//fim do método getCpf
//início do método setCpf (é com parâmetro e sem retorno)
public void setCpf(String cpf) { //permite alterar o valor do atributo cpf
do objeto
    this.cpf = cpf;
}
```

```
//fim do método setCpf
} //fim da classe
```

O arquivo chamado Produto.java terá o seguinte código:

```
package modelo; //pacote da classe
public class Produto {
    //início da declaração dos atributos
    private int id;
    private String nome, marca, modelo, descricao;
    private double valor;
    //fim da declaração dos atributos
    //início da declaração do construtor vazio
    public Produto() {
    }
    //fim da declaração do construtor vazio
    //início da declaração do construtor cheio
    public Produto(int id, String nome, String marca, String modelo, String
descricao, double valor) {
        this.id = id;
        this.nome = nome;
        this.marca = marca;
        this.modelo = modelo;
        this.descricao = descricao;
        this.valor = valor;
    }
    //fim da declaração do construtor cheio
    //início do método toString (transforma os objetos em String)
    @Override //indica que o método é herdado de uma superclasse e será
sobrescrito
    public String toString() {
        return "Código:" + id + ", produto:" + nome + ", marca:" + marca + ",
modelo:" + modelo + ", descrição:" + descricao + ", custa R$" + valor;
    }
    //fim do método toString
    //início do método getId (é sem parâmetro e com retorno do tipo int)
    public int getId() { //permite visualizar o valor do atributo id do objeto
        return id;
    }
    //fim do método getId
    //início do método setId (é com parâmetro e sem retorno)
    public void setId(int id) { //permite alterar o valor do atributo id do
objeto
        this.id = id;
    }
    //fim do método setId
    //início do método getNome (é sem parâmetro e com retorno do tipo String)
```

```

    public String getNome() { //permite visualizar o valor do atributo nome do
objeto
        return nome;
    }
    //fim do método getNome
    //início do método setNome (é com parâmetro e sem retorno)
    public void setNome(String nome) { //permite alterar o valor do atributo
nome do objeto
        this.nome = nome;
    }
    //fim do método setNome
    //início do método getMarca (é sem parâmetro e com retorno do tipo String)
    public String getMarca() { //permite visualizar o valor do atributo marca
do objeto
        return marca;
    }
    //fim do método getMarca
    //início do método setMarca (é com parâmetro e sem retorno)
    public void setMarca(String marca) { //permite alterar o valor do atributo
marca do objeto
        this.marca = marca;
    }
    //fim do método setMarca
    //início do método getModelo (é sem parâmetro e com retorno do tipo
String)
    public String getModelo() { //permite visualizar o valor do atributo modelo
do objeto
        return modelo;
    }
    //fim do método getModelo
    //início do método setModelo (é com parâmetro e sem retorno)
    public void setModelo(String modelo) { //permite alterar o valor do
atributo modelo do objeto
        this.modelo = modelo;
    }
    //fim do método setModelo
    //início do método getDescricao (é sem parâmetro e com retorno do tipo
String)
    public String getDescricao() { //permite visualizar o valor do atributo
descricao do objeto
        return descricao;
    }
    //fim do método getDescricao
    //início do método setDescricao (é com parâmetro e sem retorno)
    public void setDescricao(String descricao) { //permite alterar o valor do
atributo descricao do objeto
        this.descricao = descricao;
    }
}

```

```
//fim do método setDescricao
//início do método getValor (é sem parâmetro e com retorno do tipo double)
public double getValor() { //permite visualizar o valor do atributo valor
do objeto
    return valor;
}
//fim do método getValor
//início do método setValor (é com parâmetro e sem retorno)
public void setValor(double valor) { //permite alterar o valor do atributo
valor do objeto
    this.valor = valor;
}
//fim do método setValor
} //fim da classe
```

O arquivo chamado Venda.java terá o seguinte código:

```
package modelo;
public class Venda {
    //início da declaração dos atributos
    private int id, idPessoa, idProduto;
    private String data;
    //fim da declaração dos atributos
    //início da declaração do construtor vazio
    public Venda() { }
    //fim da declaração do construtor vazio
    //início da declaração do construtor cheio
    public Venda(int id, int idPessoa, int idProduto, String data) {
        this.id = id;
        this.idPessoa = idPessoa;
        this.idProduto = idProduto;
        this.data = data;
    }
    //fim da declaração do construtor cheio
    //início do método toString(transforma os objetos em String)
    @Override
    public String toString() {
        return "Código:" + id + ", código da pessoa:" + idPessoa + ", código do
produto:" + idProduto + ", data:" + data;
    }
    //fim do método toString
    //início do método getId (é sem parâmetro e com retorno do tipo int)
    public int getId() { //permite visualizar o valor do atributo id do objeto
        return id;
    }
    //fim do método getId
    //início do método setId (é com parâmetro e sem retorno)
```

```

    public void setId(int id) { //permite alterar o valor do atributo id do
objeto
        this.id = id;
    }
    //fim do método setId
    //início do método getIdPessoa (é sem parâmetro e com retorno do tipo int)
    public int getIdPessoa() { //permite visualizar o valor do atributo
idPessoa do objeto
        return idPessoa;
    }
    //fim do método getIdPessoa
    //início do método setIdPessoa (é com parâmetro e sem retorno)
    public void setIdPessoa(int idPessoa) { //permite alterar o valor do
atributo idPessoa do objeto
        this.idPessoa = idPessoa;
    }
    //fim do método setIdPessoa
    //início do método getIdProduto (é sem parâmetro e com retorno do tipo
int)
    public int getIdProduto() { //permite visualizar o valor do atributo
idProduto do objeto
        return idProduto;
    }
    //fim do método getIdProduto
    //início do método setIdProduto (é com parâmetro e sem retorno)
    public void setIdProduto(int idProduto) { //permite alterar o valor do
atributo idProduto do objeto
        this.idProduto = idProduto;
    }
    //fim do método setIdProduto
    //início do método getData (é sem parâmetro e com retorno do tipo String)
    public String getData() { //permite visualizar o valor do atributo data do
objeto
        return data;
    }
    //fim do método getData
    //início do método setData (é com parâmetro e sem retorno)
    public void setData(String data) { //permite alterar o valor do atributo
data do objeto
        this.data = data;
    }
    //fim do método setData
} //fim da classe

```

Realizadas as codificações das classes correspondentes as tabelas do banco de dados, serão criadas as telas de inserções de dados para cumprimento da primeira da letra C do acrônimo CRUD.

A tela para cadastrar pessoas terá o visual abaixo, considerando que no banco de dados o atributo id é autoincremento:











O código da tela para cadastrar pessoas foi desenvolvido no arquivo cadpessoa.jsp, com o seguinte conteúdo:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></scrip
t>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></sc
ript>
    <title>Exemplo CRUD</title>
  </head>
  <body>
    <h2> Cadastrar pessoa</h2>
    <form class="col-md-8" method="GET" action="controle.jsp" >
      <div class="form-row">
        <div class="col-md-8 col-sm-12 col-lg-8 mb-3">
          <label for="nome">Nome:</label>
          <input type="text" class="form-control" id="nome"
placeholder="digite o nome" name="nome" required>
        </div>
        <div class="col-md-4 col-sm-6 col-lg-4 mb-3">
          <label for="cpf">CPF:</label>
          <input type="text" class="form-control" id="cpf"
placeholder="digite o CPF" name="cpf" required>
        </div>
      </div>
    </form>
  </body>
</html>
```

```
<button style="float:right;margin-top:10px;"class="btn btn-primary"
name="cadpessoa" type="submit">Cadastrar</button>
</form>
</body>
</html>
```

As partes do código destacadas de amarelo são fundamentais no processo de inserção dos dados, uma vez que a parte `action="controle.jsp"` define para qual página os dados contidos nos inputs identificados pela parte do código `name="nome"` e `name="cpf"` serão enviados quando o botão cadastrar for clicado.

A tela para consultar pessoas terá o visual abaixo, considerando que nesta mesma tela haverá a possibilidade de iniciar o processo de alteração/atualização/edição dos dados, bem como exclusão:

Código	Nome	CPF	#
1	Maria	111111	 
2	José	55555	 
3	Laura	44444	 
4	Marcos	654321	 
5	Josefina	12313	 

A tela acima está codificada com o seguinte código:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="modelo.Conexao" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></scrip
t>
```

```

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></sc
ript>


<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<title>Exemplo CRUD</title>
</head>
<body>
<div class="container">
<h2>Consultar pessoa</h2>
<table class="table">
<thead>
<tr>
<th>Código</th>
<th>Nome</th>
<th>CPF</th>
<th>#</th>
</tr>
</thead>
<tbody>
<%
Conexao c = new Conexao();
for(int i=0;i<c.consultaPessoa().size();i++){
%>
<tr>
<td><%=c.consultaPessoa().get(i).getId() %></td>
<td><%=c.consultaPessoa().get(i).getNome() %></td>
<td><%=c.consultaPessoa().get(i).getCpf() %></td>
<td>
<form method="GET" action="controle.jsp">
<button name="altbutao" class="btn btn-warning"
value="<%=c.consultaPessoa().get(i).getId()%>">
<i class="fa fa-pencil-square-o "></i>
</button>


<button name="excbutao" class="btn btn-danger"
value="<%=c.consultaPessoa().get(i).getId()%>">
<i class="fa fa-trash " ></i>
</button>
</form>
</td>
</tr>
<% } %>
</tbody>
</table>
</div>
</body>

```




```
</html>
```

Para realizar alterações nos registros listados, basta clicar no botão  e a tela a seguir será apresentada:



Para realizar exclusões nos registros listados, basta clicar no botão  e a tela a seguir será apresentada:



O mesmo arquivo detém o código das duas telas acima, portanto serve para exclusão e atualização/edição dos registros. O código apresentado abaixo, refere-se ao arquivo `altpeessoa.jsp`, sendo relevante mencionar que a referência `excbutao` identifica que o botão  da tela de consulta foi clicado, acionando a tela de confirmação de exclusão de um registro da tabela `tbpeessoa`. A referência `excpessoa` identifica que a confirmação da exclusão foi realizada, ou seja, o botão  foi clicado, determinando a exclusão do registro. Ainda no mesmo tema, menciona-se a referência `altbutao` identifica que o botão  da tela de consulta foi clicado, acionando a tela de edição/atualização de um registro da tabela `tbpeessoa`. A referência `altpeessoa` identifica que a confirmação da edição/atualização foi realizada, ou seja, o botão

Alterar

foi clicado, determinando a edição/atualização do registro. O código do arquivo `altpessoa.jsp` é o seguinte:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></scrip
t>
        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></sc
ript>
        <title>Exemplo CRUD</title>
    </head>
    <body>
        <%
            String titulo="",tituloBotao="",editarCampo="",tituloAcao="",id="0";
            if(request.getParameter("excbutao")!=null ||
request.getParameter("excpessoa")!=null){
                id = request.getParameter("excbutao");
                titulo="Excluir pessoa";
                tituloBotao="Excluir";
                editarCampo="readonly='readonly'";
                tituloAcao = "excpessoa";
            }else if(request.getParameter("altbutao")!=null ||
request.getParameter("altpessoa")!=null){
                id =
request.getParameter("altbutao")!=null?request.getParameter("altbutao"):reques
t.getParameter("altpessoa");
                titulo="Alterar pessoa";
                tituloBotao="Alterar";
                tituloAcao = "altpessoa";
            }
        %>
        <h2><%=titulo%> </h2>
        <form class="col-md-8" method="GET" action="controle.jsp" >
            <div class="form-row">
                <input type="hidden" id="nome" value="<%=id %>" name="id"
>
                <div class="col-md-8 col-sm-12 col-lg-8 mb-3">
                    <label for="nome">Nome:</label>
```

```

        <input type="text" class="form-control" id="nome"
value="<%=request.getParameter("nome")%>" name="nome" required
<%=editarCampo%>
    </div>
    <div class="col-md-4 col-sm-6 col-lg-4 mb-3">
        <label for="cpf">CPF:</label>
        <input type="text" class="form-control" id="cpf"
value="<%=request.getParameter("cpf")%>" name="cpf" required
<%=editarCampo%>
    </div>
</div>
    <button style="float:right;margin-top:10px;" class="btn btn-danger"
name="<%=tituloAcao%>" type="submit"><%=tituloBotao%></button>
</form>
</body>
</html>

```

O arquivo controle.jsp é responsável por relacionar as requisições do usuário com os métodos da classe Conexão do arquivo Conexao.java, permitindo informar os resultados das operações ao usuário, bem como implementá-las no banco de dados.

O código utilizado no arquivo controle.jsp é o descrito abaixo, valendo informar que uma estrutura condicional foi criada com o objetivo de identificar qual ação o usuário realizou e com base nesta redirecionar para algum método do arquivo Conexao.java:

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="modelo.Conexao" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            Conexao c = new Conexao();
            if(request.getParameter("cadpessoa")!=null &&
request.getParameter("nome")!=null &&
request.getParameter("cpf")!=null){
                c.inserirPessoasBD(request.getParameter("nome"),request.getPar
ameter("cpf"));
                out.print("<h2>Pessoa inserida no banco de dados.</h2>");
                response.setHeader("Refresh","3;URL=cadpessoa.jsp");
            }else if(request.getParameter("altbutao")!=null){

```

```

        String id = request.getParameter("altbutao");
        String nome = request.getParameter("nome");
        String cpf = request.getParameter("cpf");
        response.setHeader("Refresh", "0;URL=altpeessoa.jsp?nome="+nome
+"&cpf="+cpf+"&altbutao="+id+"");
    }else if(request.getParameter("excbutao")!=null){
        String id = request.getParameter("excbutao");
        String nome = request.getParameter("nome");
        String cpf = request.getParameter("cpf");
        response.setHeader("Refresh", "0;URL=altpeessoa.jsp?nome="+nome
+"&cpf="+cpf+"&excbutao="+id+"");
    }else if(request.getParameter("altpeessoa")!=null){
        c.alterarPessoasBD(request.getParameter("id"),request.getParam
eter("nome"),request.getParameter("cpf"));
        out.print("<h2>Pessoa atualizada no banco de dados.</h2>");
        response.setHeader("Refresh", "3;URL=conpeessoa.jsp");
    }else if(request.getParameter("excpessoa")!=null){
        c.excluirPessoasBD(request.getParameter("id"),request.getParam
eter("nome"),request.getParameter("cpf"));
        out.print("<h2>Pessoa excluída no banco de dados.</h2>");
        response.setHeader("Refresh", "3;URL=conpeessoa.jsp");
    }else{
        out.print("<h2>Operação não realizada</h2>");
        response.setHeader("Refresh", "13;URL=cadpeessoa.jsp");
    }

    %>
</body>
</html>

```

O arquivo Conexao.java irá promover as conexões com o banco de dados, por meio da classe Conexao que detém métodos para cada letra do acrônimo CRUD, conforme código abaixo:

```

package modelo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
public class Conexao {
    private final String driver = "com.mysql.cj.jdbc.Driver";
    private final String servidor = "jdbc:mysql://localhost/dbaula";
    private final String usuario = "root";
    private final String senha = "";

```

```
private Connection conectar(){
    try{
        Class.forName(driver);
        return DriverManager.getConnection(servidor, usuario, senha);
    }catch(ClassNotFoundException | SQLException ex){
        System.err.println(ex);
        return null;
    }
}

public ArrayList<Pessoa> consultaPessoa(){
    ArrayList<Pessoa> arrayPessoa = new ArrayList<>();
    try{
        ResultSet listaPessoas =
conectar().createStatement().executeQuery("select * from tbpessoa");
        while(listaPessoas.next())
            arrayPessoa.add(new Pessoa(listaPessoas.getInt("id"),
                listaPessoas.getString("nome"),
                listaPessoas.getString("cpf")));
    }catch(Exception ex){
        System.out.println(ex);
    }
    return arrayPessoa;
}

public void inserirPessoasBD(String nome, String cpf){
    try{
        PreparedStatement inserirPessoa =
conectar().prepareStatement("insert into tbpessoa(nome,cpf)
values('"+nome+"','"+cpf+"')");
        inserirPessoa.execute();
        conectar().commit();
    }catch(Exception ex){
        System.out.println(ex);
    }
}

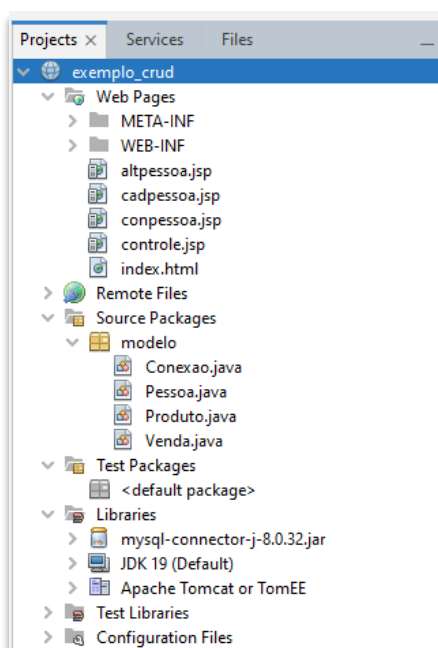
public void alterarPessoasBD(String id, String nome, String cpf){
    try{
        PreparedStatement alterarPessoa =
conectar().prepareStatement("update tbpessoa set nome='"+nome+"',
cpf='"+cpf+"' where id='"+id+"'");
        alterarPessoa.execute();
        conectar().commit();
    }catch(Exception ex){
        System.out.println(ex);
    }
}

public void excluirPessoasBD(String id, String nome, String cpf){
```



```
try{
    PreparedStatement alterarPessoa =
conectar().prepareStatement("delete from tbpessoa where id='"+id+"' and
cpf='"+cpf+"'");
    alterarPessoa.execute();
    conectar().commit();
}catch(Exception ex){
    System.out.println(ex);
}
}
```

Vale registrar que com as explicações e códigos deste material é possível reproduzir o CRUD para as tabelas/entidades tbproduto e tbvenda, finalizando a projeto, que por ora detém a seguinte estrutura:



Referências

- ★ https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_basic&stacked=h
- ★ https://repositorio.ufsm.br/bitstream/handle/1/25212/TG328_Felipe%20Gabriel%20Arend.pdf?sequence=1
- ★ <https://mariadb.org/documentation/>
- ★ https://www.w3schools.com/bootstrap/bootstrap_tables.asp
- ★ https://www.cin.ufpe.br/~wsr/tutorial_jsp.pdf
- ★ <http://www.dsc.ufcg.edu.br/~jacques/cursos/j2ee/html/jsp/livros.htm>