

Workshop 02 – Livraria Frameworks

O objetivo deste workshop é consolidar o conhecimento adquirido durante o curso. Será desenvolvido uma Livraria virtual. Nesse workshop, iremos trabalhar na parte de cadastro de clientes, no carrinho de compras e na verificação dos pedidos feitos por cada cliente.

Exercícios

Exercício 1: Criando a tabela pedido e itemPedido

Exercício 2: Criando a classe ItemCarrinho

Exercício 3: Criando a classe Carrinho e CarrinhoBean

Exercício 4: Desenvolvendo a página do carrinho

Exercício 5: Criando as classes Cliente, ClienteDao e ClienteBean

Exercício 6: Desenvolvendo a página de cadastro e Login

Exercício 7: Criando as classes Pedido e PedidoDao

Exercício 8: Página VerificarPedido e VerificarItem

Exercício 1 – Criando as tabelas

1. Vamos complementar nosso banco de dados com as tabelas que iremos usar nesse laboratório, as tabelas de cliente, pedido e itemPedido. Crie as tabelas no nosso banco de dados Livraria criado anteriormente assim como representado abaixo:

TABELA CLIENTE:

cod_cliente [PK] serial	nome character varying(50)	login character	senha character	endereco character	cidade character	bairro character	estado character	cep character
----------------------------	-------------------------------	--------------------	--------------------	-----------------------	---------------------	---------------------	---------------------	------------------

TABELA PEDIDO:

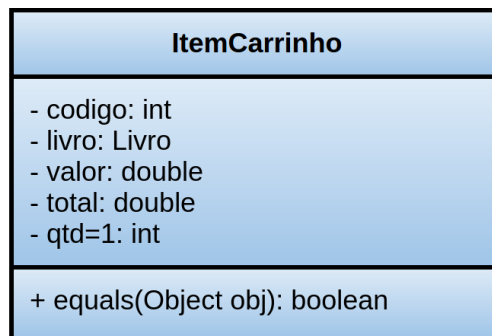
cod_pedido [PK] serial	data_pedido date	status character varying(50)	pagamento character varying(10)	cod_cliente bigint	total numeric(6,2)
---------------------------	---------------------	---------------------------------	------------------------------------	-----------------------	-----------------------

TABELA ITEMPELIDO:

cod_item [PK] serial	qtd integer	cod_livro bigint	cod_pedido bigint
-------------------------	----------------	---------------------	----------------------

Exercício 2 – Criando a classe ItemCarrinho

1. No nosso projeto, crie a classe ItemCarrinho. Ela representará a entidade ItemCarrinho em nosso projeto, seguindo o padrão VO (Value Object) e conterá os dados de um item do carrinho. Não se esqueça de acrescentar os getters e setters na classe.



```

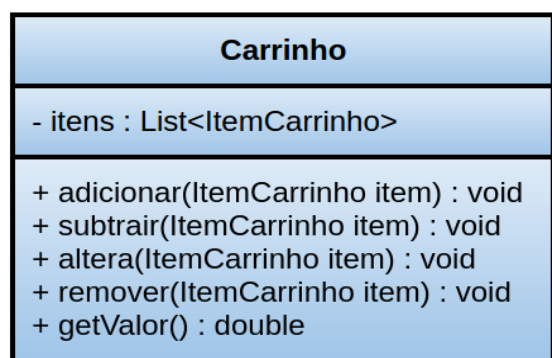
equals(){
    ItemCarrinho item = (ItemCarrinho) obj;
    if(item.getLivro().getCodigo()==this.getLivro().getCodigo()){
        return true;
    }
    else{
        return false;
    }
}

getValor(){
    return livro.getPreco()*qtd;
}

```

Exercício 3 – Criando a classe Carrinho e CarrinhoBean

1. A classe Carrinho será apenas um Bean. Esta classe representa o Carrinho de compras contendo os itens e métodos para o controle do carrinho. Não se esqueça dos getters e setters.



```

Adicionar(){
    for(ItemCarrinho itm: itens){
        if(itm.equals(item)){
            itm.setQtd(itm.getQtd()+1);
            return;
        }
    }
    itens.add(item);
}

```

```

    }

    Subtrair(){
        for(ItemCarrinho itm: itens){
            if(itm.equals(item)){
                itm.setQtd(itm.getQtd()-1);
                return;
            }
        }
        itens.remove(item);
    }

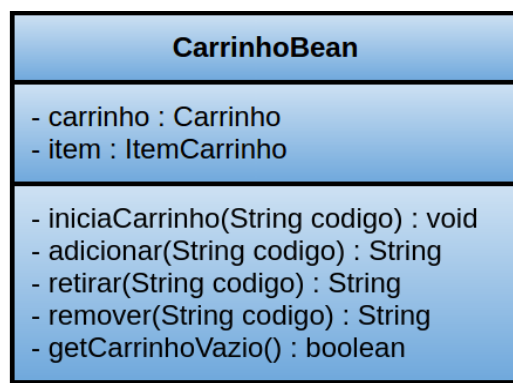
    Remover(){
        itens.remove(item);
    }

    Altera(){
        itens.remove(item);
        itens.add(item);
    }

    getValor(){
        double result = 0;
        for(ItemCarrinho item : itens){
            result += item.getValor();
        }
        return result;
    }

```

2. Crie também a classe CarrinhoBean. Essa vai ser nosso bean funcional, que vai fazer as associações necessárias entre a entidade e a nossa página.



```

iniciaCarrinho(){
    int codigo;
    LivroDao dao = null;
    if(carrinho == null){
        carrinho = new Carrinho();
    }
    try{
        codigo = Integer.parseInt(cod);
    }catch(NumberFormatException e){
        codigo = 0;
    }
}

```

```
try{
    dao = new LivroDao();
}
catch(SQLException e1){
    e1.printStackTrace();
}

Livro livro = dao.consultar(codigo);
ItemCarrinho item2 = new ItemCarrinho();
item2.setLivro(livro);
item2.setCodigo(livro.getCodigo());
item = item2;
}

adicionar(){

    iniciaCarrinho(codigoS);
    carrinho.adicionar(item);

    return "Carrinho";
}

retirar(){

    iniciaCarrinho(codigoS);
    carrinho.subtrair(item);
}

remover(){

    iniciaCarrinho(codigoS);
    carrinho.remover(item);
}

getCarrinhoVazio(){

    if(carrinho == null){
        carrinho = new Carrinho();
    }
    if(carrinho.getItems().size() == 0){
        return false;
    }
    return true;
}
```

Exercício 4 – Desenvolvendo a página carrinho

1. Na página do carrinho, usaremos um layout diferente do resto do site. Então, crie um novo template **LayoutCarrinho.xhtml** e nele, vamos alterar apenas o cabeçalho, iremos criar um novo chamado **carrinhoHeader.xhtml**.

```
<ui:composition>
    <header>
        <nav class=" navbar navbar-primary navbar-static-top">
            <div class="navbar-header">
                <h:link value="TRIWAY Livraria" outcome="Home"
                    class="navbar-brand"/>
                <button class="navbar-toggle" type="button">
```

```

        data-target="navbar-collapse" data-
        toggle="collapse">menu</button>
    </div>
    <h:form>
        <ul class="nav navbar-nav collapse navbar-collapse
            navbar-right">
            <li><h:link value="Olá, cliente" /></li>
            <li><h:commandLink value="Verificar Pedidos"/></li>
            <li><h:link value="Sair" outcome="Home"/></li>
        </ul>
    </h:form>
</nav>
</header>
</ui:composition>

```

LayoutCarrinho.xhtml

```

<h:body>
    <div class="page">
        <div class="header container">
            <ui:insert name="header">
                <ui:include src="/WEB-INF/template/carrinhoHeader.xhtml" />
            </ui:insert>
        </div>
    </div>
</h:body>

```

- Feito isso, crie o arquivo **Carrinho.xhtml** e nele adicione um painel onde ficarão todos os livros selecionados.

```

<table class="table cart-items">
    <thead>
        <tr>
            <th class="produto">Produto</th>
            <th class="produto-preco">Preço</th>
            <th class="quantidade">Quantidade</th>
            <th class="preco-total">Total</th>
            <th class="item-remove"></th>
        </tr>
    </thead>
    <ui:repeat value="#{carrinhoBean.carrinho.itens}" var="item">
        <tbody id="tabela">
            <tr class="produto-item">
                <td class="produto-image"></td>
                <td class="produto-nome">
                    <h:form>
                        <h:commandLink action="#{pesquisaBean.verLivro(item.codigo)}"
                            value="#{item.livro.titulo}" />
                    </h:form>
                    <div class="autor">
                        <h:outputLabel value="#{item.livro.autor}" />
                    </div>
                </td>
                <td class="produto-preco">
                    <h:outputText value="#{item.livro.preco}">
                        <f:convertNumber currencyCode="BRL" type="currency" />
                    </h:outputText>
                </td>
                <td class="quantidade">

```

```

        <h:form>
            <p:commandLink value=" + " />
            <h:outputText id="qtd" value="#{item.qtd}" />
            <p:commandLink value=" - " />
        </h:form>
    </td>
    <h:form id="frm2">
        <td class="preco-total" id="precoOut">
            <h:outputText value="#{item.valor}" id="valor">
            <f:convertNumber currencyCode="BRL" type="currency" />
            </h:outputText>
        </td>
    </h:form>
    <td class="item-remove">
        <h:form>
            <p:commandLink value="X"
                action="#{carrinhoBean.remover(item.codigo)}"
                update="@all" />
        </h:form>
    </td>
</tr>
</tbody>
</ui:repeat>
</table>

<div class="forma-pagamento">
<div class="row">
<div class="col-md-8">
    <div class="panel panel-default">
        <div class="panel-body">
            <h:form>
                <legend>Forma de Pagamento</legend>
                <div class="radio">
                    <label>
                        <input value="cartao" type="radio" name=" optradio"
                            checked="checked" />
                        Cartão de Crédito
                    </label>
                    <label>
                        <input value="boleto" type="radio" name="optradio" />
                        Boleto Bancário (10% desc.)
                    </label>
                </div>
            </h:form>
            <div class="cartao box">
                <h:panelGrid class="menu-bandeira">
                    <p:outputLabel value="Número - CVV" for="num-cartao" />
                    <p:inputText id="num-cartao" />
                    <p:outputLabel value="Validade" for="val-cartao" />
                    <p:calendar id="val-cartao" />
                    <p:outputLabel value="Bandeira" for="bandeira-cartao" />
                    <p:selectOneButton id="bandeira-cartao">
                        <f:selectItem itemLabel="Visa" itemValue="visa" />
                        <f:selectItem itemLabel="MasterCard" itemValue="master" />
                        <f:selectItem itemLabel="American Express" itemValue="amex" />
                    </p:selectOneButton>
                </h:panelGrid>
                <p:commandButton value="Finalizar compra"/>
            </div>
            <div class="boleto box">
                <h:panelGrid>
                    <p:outputLabel value="Número do CPF" for="num-cpf" />
                    <p:inputText id="num-cpf" placeholder="000.000.000-00" />
                </h:panelGrid>
                <p:commandButton value="Finalizar Compra"/>
            </div>
            <p:commandButton value="Continuar Comprando"
                action="#{pesquisaBean.pesquisar}" />
        </div>
    </div>
</div>

```

```

        </h:form>
    </div>
</div>
<div class="col-md-4">
    <table class="table total-compra">
        <tbody>
            <tr>
                <td>Total dos Produtos</td>
                <td>
                    <h:form id="frm3">
                        <h:outputText value="#{carrinhoBean.carrinho.valor}"
                            id="total1">
                            <f:convertNumber currencyCode="BRL" type="currency" />
                        </h:outputText>
                    </h:form>
                </td>
            </tr>
            <tr>
                <td>Descontos</td>
                <td>R$ 00,00</td>
            </tr>
        </tbody>
    <tfoot>
        <tr>
            <td>TOTAL:</td>
            <td>
                <h:form id="frm4">
                    <h:outputText value="#{carrinhoBean.carrinho.valor}"
                        id="total2">
                        <f:convertNumber currencyCode="BRL" type="currency" />
                    </h:outputText>
                </h:form>
            </td>
        </tr>
    </tfoot>
</table>
</div>
</div>
</div>

```

- Para acrescentar um livro no seu carrinho, vamos voltar à página **Livro.xhtml** e adicionar uma ação de adicionar livro ao botão “Adicionar Livro” criado no workshop passado.

```

<p:commandButton value="Adicionar ao Carrinho"
    action="#{carrinhoBean.adicionar(pesquisaBean.livro.codigo)}"/>

```

Essa ação irá fazer com que o livro selecionado seja adicionado ao carrinho de compras e já te envia para a página do carrinho.

#DESAFIO: Faça com que o cliente seja capaz de adicionar o livro selecionado diretamente da página **Resultado.xhtml**, sem precisar entrar no Livro.xhtml toda vez que quiser colocar mais um livro no carrinho.

- Agora vamos fazer com que a página nos apresente uma mensagem caso o carrinho esteja vazio. Isso será possível através do atributo **rendered** de algumas tags. Preste atenção na composição das tags e do código.

```

<div class="jumbotron">
    <h:form rendered="#{carrinhoBean.carrinhoVazio}">
        <h1>Ótima Escolha</h1>
        <p>Obrigado por comprar na TriWay Livraria! Por favor, confirme
            seus dados antes de efetivar a compra!</p>
    </h:form>

```

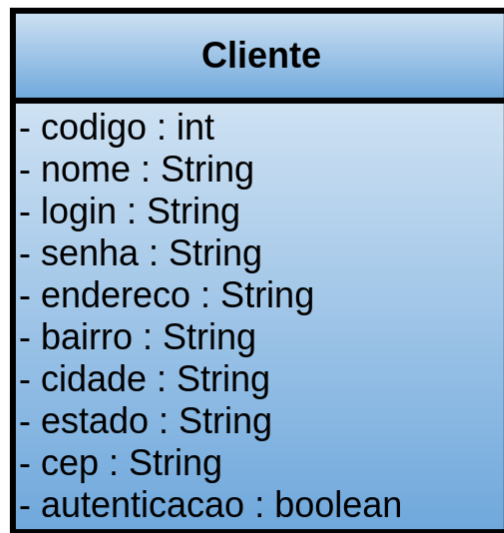
```
<h:form rendered="#{carrinhoBean.carrinhoVazio == false}">
  <h1>Carrinho Vazio</h1>
  <p>Adicione produtos ao seu carrinho de compras para continuar!</p>
  <p:commandButton value="Continuar Comprando"
    action="#{pesquisaBean.pesquisar}" />
</h:form>
</div>
```

5. Faça o teste, adicione um livro ao seu carrinho de compras e veja a página em funcionamento.

#DESAFIO: Até o momento, você só consegue acessar seu carrinho adicionando um livro ou escrevendo o html da página. Faça com que o cliente consiga acessar o carrinho de compras diretamente da página inicial, através do cabeçalho.

Exercício 5 – Criando as classes Cliente, ClienteDao e ClienteBean

1. Esta classe representará um cliente do sistema. Crie-a a partir do UML abaixo, não esquecendo dos getters e setters.



2. Vamos criar agora o ClienteDao, de acordo com o código abaixo:

```
public class ClienteDao {
    Connection conexao;

    public ClienteDao() throws SQLException{
        this.conexao= FabricaConexao.getConexao();
    }

    public Cliente inserir(Cliente cliente){
        try{
            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO CLIENTE
(NOME,LOGIN,SENHA,ENDereco,CIDADE,BAIRRO,ESTADO,CEP) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
            PreparedStatement consulta = conexao.prepareStatement(sql.toString());
            consulta.setString(1, cliente.getNome());
            consulta.setString(2, cliente.getLogin());
            consulta.setString(3, cliente.getSenha());
            consulta.setString(4, cliente.getEndereco());
            consulta.setString(5, cliente.getCidade());
            consulta.setString(6, cliente.getBairro());
            consulta.setString(7, cliente.getEstado());
            consulta.setString(8, cliente.getCep());
            consulta.execute();
        }
    }
}
```



```
    }catch(SQLException e){
        e.printStackTrace();
    }
    return cliente;
}

public Cliente autenticar(Cliente cliente){
    try{
        StringBuffer sql = new StringBuffer();
        sql.append("SELECT NOME, SENHA, LOGIN, ENDERECO, CIDADE, BAIRRO, ESTADO,
        CEP, COD_CLIENTE FROM CLIENTE where SENHA = ? and LOGIN = ?");
        this.conexao= FabricaConexao.getConexao();
        PreparedStatement consulta = conexao.prepareStatement(sql.toString());
        consulta.setString(1, cliente.getSenha());
        consulta.setString(2, cliente.getLogin());
        ResultSet resultado = consulta.executeQuery();
        if(resultado.next()){
            cliente.setNome(resultado.getString("NOME"));
            cliente.setSenha(resultado.getString("SENHA"));
            cliente.setLogin(resultado.getString("LOGIN"));
            cliente.setEndereco(resultado.getString("ENDERECO"));
            cliente.setCidade(resultado.getString("CIDADE"));
            cliente.setBairro(resultado.getString("BAIRRO"));
            cliente.setEstado(resultado.getString("ESTADO"));
            cliente.setCep(resultado.getString("CEP"));
            cliente.setCodigo(resultado.getInt("COD_CLIENTE"));
            cliente.setAutenticacao(true);
        }
        else{
            cliente=null;
        }
    }catch(SQLException e){
        e.printStackTrace();
    }
    return cliente;
}

public void alterar(Cliente cliente){
    try{
        StringBuffer sql = new StringBuffer();
        sql.append("update CLIENTE SET SENHA = ?, LOGIN = ?, ENDERECO = ?,
        CIDADE = ?, BAIRRO = ?, ESTADO = ?, CEP = ? where COD_CLIENTE = ?");

        this.conexao= FabricaConexao.getConexao();
        PreparedStatement consulta = conexao.prepareStatement(sql.toString());

        consulta.setString(1, cliente.getSenha());
        consulta.setString(2, cliente.getLogin());
        consulta.setString(3, cliente.getEndereco());
        consulta.setString(4, cliente.getCidade());
        consulta.setString(5, cliente.getBairro());
        consulta.setString(6, cliente.getEstado());
        consulta.setString(7, cliente.getCep());
        consulta.setInt(8, cliente.getCodigo());
        consulta.execute();
    }
    catch(SQLException e){
        e.printStackTrace();
    }
}

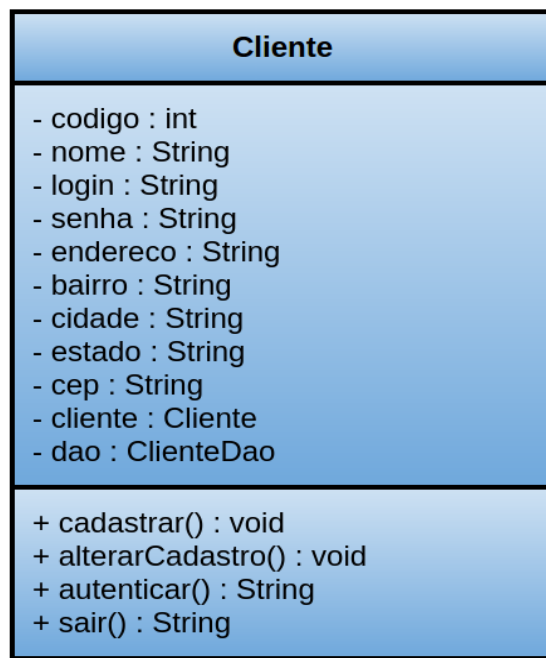
public boolean existeCliente(String login){
    boolean result = true;
    try{
        StringBuffer sql = new StringBuffer();
        sql.append("SELECT (LOGIN, COD_CLIENTE) FROM CLIENTE WHERE LOGIN = ?");
        this.conexao= FabricaConexao.getConexao();
```

```

        PreparedStatement consulta = conexao.prepareStatement(sql.toString());
        consulta.setString(1, login);
        ResultSet resultado = consulta.executeQuery();
        if(!resultado.next()){
            result = false;
        }
    }
    catch(SQLException e){
        e.printStackTrace();
    }
    return result;
}
}

```

3. Finalizado o Dao, crie agora a classe ClienteBean e desenvolva assim como mostrado na UML e no código abaixo, sem esquecer dos getters e setters:



```

cadastrar(){
    FacesMessage msg = null;
    if(cliente == null){
        cliente = new Cliente();
    }
    try{
        dao = new ClienteDao();
    }catch(SQLException e){
        e.printStackTrace();
    }
    cliente.setSenha(senha);
    cliente.setNome(nome);
    cliente.setEndereco("endereco");
    cliente.setBairro("bairro");
    cliente.setCidade("cidade");
    cliente.setCep("cep");
    cliente.setEstado("estado");

    System.out.println("Usuário: "+usuario);
    System.out.println("Senha: "+senha);
}

```

```
        if(dao.existeCliente(usuario)){
            msg = new FacesMessage("Já existe esse usuário em nosso sistema");
            msg.setSeverity(FacesMessage.SEVERITY_WARN);
        }else{
            cliente.setLogin(usuario);
            dao.inserir(cliente);
            msg = new FacesMessage("Cliente cadastrado com sucesso!");
        }
        FacesContext.getCurrentInstance().addMessage(null, msg);
    }

    alterarCadastro(){
        FacesMessage msg = null;
        if(cliente == null){
            cliente = new Cliente();
        }
        try{
            dao = new ClienteDao();
        }catch(SQLException e){
        }

        cliente.setSenha(senha);
        cliente.setNome(nome);
        cliente.setEndereco(endereco);
        cliente.setBairro(bairro);
        cliente.setCidade(cidade);
        cliente.setCep(cep);
        cliente.setEstado(estado);

        msg = new FacesMessage("Cadastro alterado com Sucesso!");
        FacesContext.getCurrentInstance().addMessage(null, msg);
        System.out.println("Cadastro alterado com sucesso");
    }

    autenticar(){
        FacesMessage msg = null;
        HttpSession session = (HttpSession)
        FacesContext.getCurrentInstance().getExternalContext().getSession(false);

        if(cliente == null){
            cliente = new Cliente();
        }
        try{
            dao = new ClienteDao();
        }catch(SQLException e){
            e.printStackTrace();
        }

        cliente.setLogin(usuario);
        cliente.setSenha(senha);

        cliente = dao.autenticar(cliente);
        if(cliente == null){
            msg = new FacesMessage("Login incorreto");
            msg.setSeverity(FacesMessage.SEVERITY_ERROR);
            System.out.println("Senha ou Usuario incorretos");
            FacesContext.getCurrentInstance().addMessage(null, msg);
            return "Login";
        }else{
```

```

        System.out.println("Cliente "+cliente.getNome()+ " Autenticado!");
        cliente.setAutenticacao(true);
        FacesContext.getCurrentInstance().getExternalContext()
            .getSessionMap().put("user", cliente.getLogin());
        session.setAttribute("cliente", cliente);
        return "Home";
    }

}

sair(){
    cliente.setAutenticacao(false);
    FacesContext.getCurrentInstance().getExternalContext().invalidateSession();
    System.out.println("Cliente "+cliente.getNome()+ " saiu da sessão!");
    cliente = null;
    return "Home";
}

```

Exercício 6 – Criando o arquivo Cadastro.xhtml e Login.xhtml

1. Nesse exercício, faremos a página onde funcionará um formulário de cadastro de clientes e vamos tentar logar esse cliente na nossa aplicação. Primeiro, crie o arquivo **Cadastro.xhtml** e adicione o formulário abaixo.

```

<div class="container">
    <h:form>
    <div class="panel panel-primary painel-cadastro">
        <div class="panel-heading">Cadastro</div>
        <div class="panel-body">
            <p:panelGrid>
                <p:messages/>
                <div class="form-group">
                    <p:outputLabel value="Nome" for="nome"/>
                    <p:inputText id="nome" required="true" class="form-control"
                        value="#{clienteBean.nome}"/>
                </div>
                <div class="form-group">
                    <p:outputLabel value="Login" for="login"/>
                    <p:inputText id="login" required="true" class="form-control"
                        value="#{clienteBean.usuario}"/>
                </div>
                <div class="form-group">
                    <p:outputLabel value="Senha" for="senha"/>
                    <p:password id="senha" required="true" class="form-control"
                        value="#{clienteBean.senha}" match="ConfSenha" >
                        <f:validateLength minimum="6" />
                    </p:password>
                </div>
                <div class="form-group">
                    <p:outputLabel value="Confirme sua Senha" for="ConfSenha"/>
                    <p:password id="ConfSenha" required="true"
                        class="form-control"/>
                </div>
            </p:panelGrid>
            <p:commandButton value="Cadastrar" icon="ui-icon-disk"
                action="#{clienteBean.cadastrar}" update="@form"/>
        </div>
    </div>

```

```
</div>
</h:form>
```

2. Teste sua página. Crie um cliente, faça seu cadastro e veja na sua tabela SQL se foi realmente adicionado.

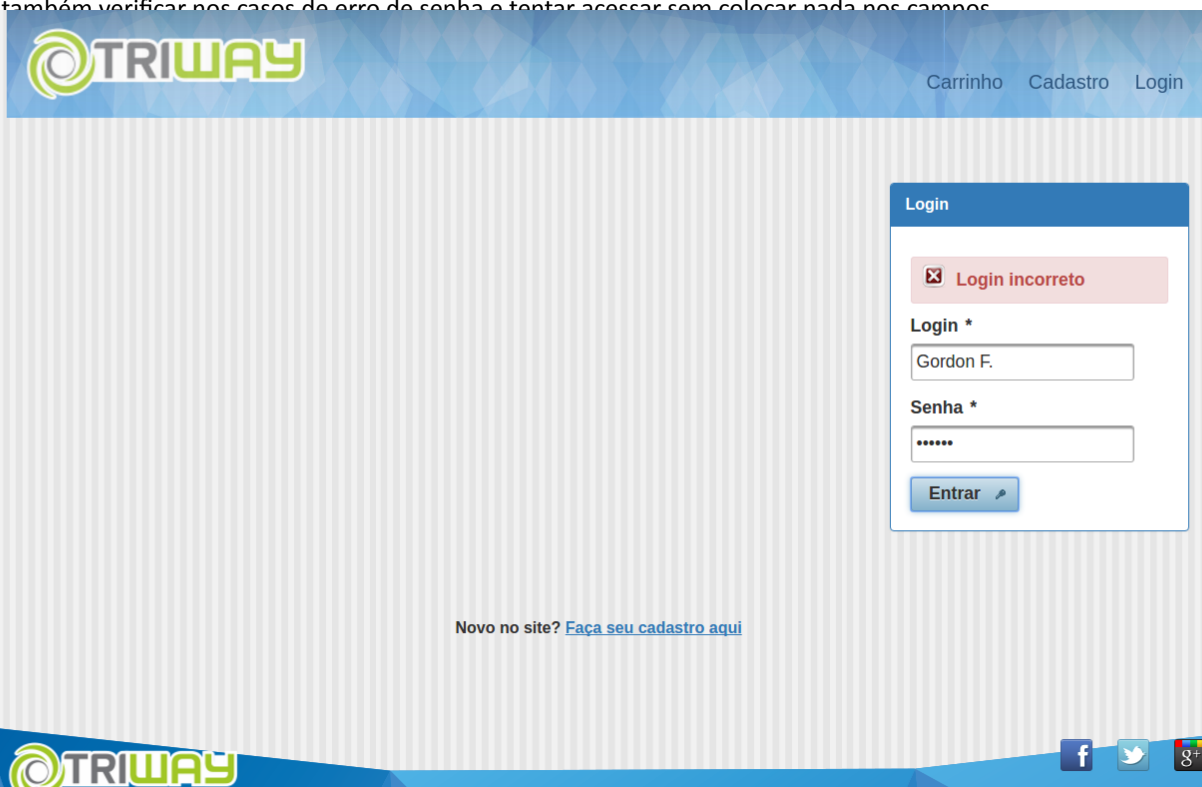
cod_cliente [PK] serial	nome character	login character	senha character	endereço character	cidade character	bairro character	estado character	cep character
1	Lucas	lucas	123	rua	cidade	bairro	UF	000000

3. Crie agora o arquivo **Login.xhtml** e adicione o seguinte formulário.

```
<div class="container">
    <div class="panel panel-primary login">
        <div class="panel-heading">Login</div>
        <div class="panel-body">
            <p:messages autoUpdate="true"/>
            <h:form>
                <p:panelGrid>
                    <p:outputLabel value="Login" for="login"/><br/>
                    <p:inputText id="login" required="true"
                        value="#{clienteBean.usuario}"/>
                    <p:outputLabel value="Senha" for="senha"/><br/>
                    <p:password id="senha" required="true"
                        value="#{clienteBean.senha}"/>
                </p:panelGrid>
                <p:commandButton value="Entrar" action="#{clienteBean.autenticar}"
                    icon="ui-icon-key" iconPos="right"/>
            </h:form>
        </div>
    </div>
</div>
```

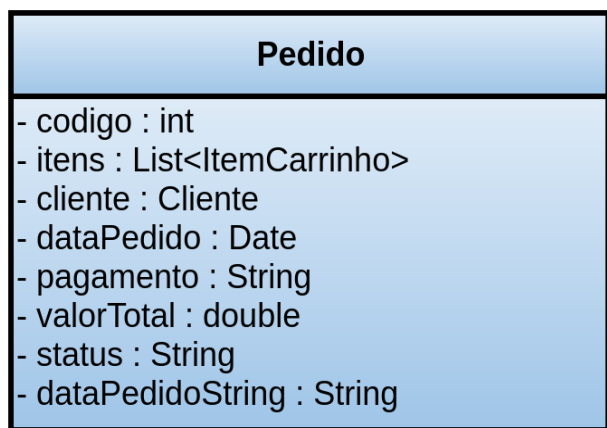
#DESAFIO: Faça uma referencia à página de cadastro para o caso do cliente não ter cadastro no site. Lembre-se também de fazer referencia à página de cadastro a partir do menu do cabeçalho.

4. Tente fazer o login no site com o cliente cadastrado anteriormente e veja se está funcionando corretamente. Tente também verificar nos casos de erro de senha e tentar acessar sem colocar nada nos campos.

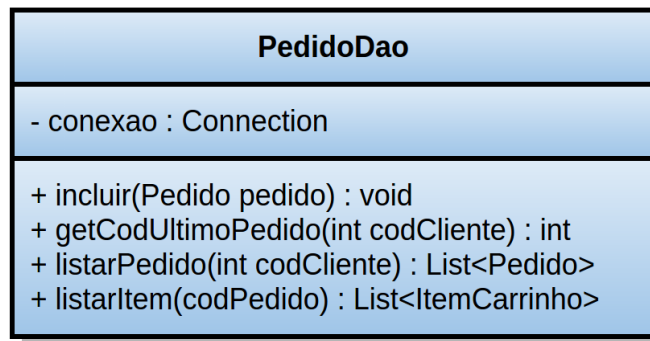


Exercício 7 – Criando as classes Pedido e PedidoDao

1. Para termos um serviço de armazenamento de pedidos, vamos criar as classes Pedido e PedidoDao. Crie agora o arquivo **Pedido.class** que será a entidade responsável por salvar temporariamente todas as informações de pedidos necessárias. Siga a tabela UML abaixo, e não se esqueça dos getters e setters.



2. Para salvarmos esses dados em nossa tabela SQL, vamos criar a classe PedidoDao conforme o UML e o código abaixo.



```
incluir(){
```

```
    Date dataPedido = new Date(pedido.getDataPedido().getTime());
```

```
    StringBuffer sql = new StringBuffer();
```

```
    sql.append("INSERT INTO PEDIDO (DATA_PEDIDO, COD_CLIENTE, STATUS, PAGAMENTO, TOTAL)  
              VALUES (?, ?, ?, ?, ?)");
```

```
    StringBuffer sqlItem = new StringBuffer();
```

```
    sqlItem.append("INSERT INTO ITEM_PEDIDO (COD_LIVRO, QTD, COD_PEDIDO) VALUES (?, ?, ?)");
```

```
    try{
```

```
        this.conexao = FabricaConexao.getConexao();
```

```
        PreparedStatement consulta = conexao.prepareStatement(sql.toString());
```

```
        consulta.setDate(1, dataPedido);
```

```
        consulta.setInt(2, pedido.getCliente().getCodigo());
```

```
        consulta.setString(3, pedido.getStatus());
```

```
        consulta.setString(4, pedido.getPagamento());
```

```
        consulta.setDouble(5, pedido.getValorTotal());
```

```
        consulta.execute();
```

```
        PreparedStatement consultaItem =
```

```
        conexao.prepareStatement(sqlItem.toString());
```

```
        for(ItemCarrinho item : pedido.getItens()){
```

```
            consultaItem.setInt(1, item.getLivro().getCodigo());
```

```
            consultaItem.setInt(2, item.getQtd());
```

```
            consultaItem.setInt(3,
```

```
            getCodUltimoPedido(pedido.getCliente().getCodigo());
```

```
            consultaItem.execute();
```

```
        }
```

```
    } catch(SQLException e){
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
getCodUltimoPedido(){
```

```
    int cod=1;
```

```
    try{
```

```
        PreparedStatement consulta = conexao.prepareStatement("SELECT MAX(COD_PEDIDO)  
                      AS COD_PEDIDO FROM PEDIDO WHERE COD_CLIENTE = ?");
```

```
        consulta.setInt(1, codCliente);
```

```
        ResultSet resultado = consulta.executeQuery();
```

```
        if(resultado.next()){
```

```
            cod=resultado.getInt("COD_PEDIDO");
```

```
        }
```

```
    } catch(SQLException e){
```

```
        e.printStackTrace();
```

```
    }
```

```
    return cod;
```

```
}
```

```
listarPedido() {
    ArrayList<Pedido> pedidos = new ArrayList<Pedido>();
    StringBuffer sql = new StringBuffer();
    sql.append("SELECT pedido.COD_PEDIDO, pedido.DATA_PEDIDO, pedido.PAGAMENTO,
pedido.STATUS ");
    sql.append("FROM pedido WHERE pedido.COD_CLIENTE = ?");

    try {
        PreparedStatement consulta = conexao.prepareStatement(sql.toString());
        consulta.setInt(1, codCliente);
        ResultSet resultado = consulta.executeQuery();
        while (resultado.next()) {
            Pedido pedido = new Pedido();

            pedido.setCodigo(resultado.getInt("COD_PEDIDO"));
            pedido.setDataPedido(resultado.getDate("DATA_PEDIDO"));
            pedido.setPagamento(resultado.getString("PAGAMENTO"));
            pedido.setStatus(resultado.getString("STATUS"));

            pedidos.add(pedido);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return pedidos;
}

listarItem() {
    ArrayList<ItemCarrinho> itens = new ArrayList<ItemCarrinho>();
    StringBuffer sql = new StringBuffer();
    sql.append("SELECT estoque.IMAGEM, estoque.TITULO, estoque.preco ,
item_pedido.QLD, pedido.TOTAL FROM cliente INNER JOIN pedido ON
cliente.COD_CLIENTE = pedido.COD_CLIENTE INNER JOIN item_pedido ON
pedido.COD_PEDIDO = item_pedido.COD_PEDIDO INNER JOIN estoque ON
item_pedido.COD_LIVRO = estoque.COD_LIVRO WHERE pedido.COD_PEDIDO = ?");
    try {
        PreparedStatement consulta = conexao.prepareStatement(sql.toString());
        consulta.setInt(1, codPedido);
        ResultSet resultado = consulta.executeQuery();
        while (resultado.next()) {
            ItemCarrinho item = new ItemCarrinho();
            Livro livro = new Livro();
            livro.setImagem(resultado.getString("IMAGEM"));
            livro.setTitulo(resultado.getString("TITULO"));
            livro.setPreco(resultado.getDouble("PRECO"));
            item.setTotal(resultado.getDouble("TOTAL"));
            item.setLivro(livro);
            item.setQtd(resultado.getInt("QLD"));
            itens.add(item);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return itens;
}
```

3. Pronto, temos um método eficiente de organizar pedidos por cliente. Mas para funcionar, precisamos garantir de que o carrinho de compras seja finalizado somente se houver um cliente autenticado na página. Vamos voltar a classe CarrinhoBean e à página Carrinho para fazer algumas alterações. Vá em CarrinhoBean e adicione os seguintes métodos.


```
public void finalizarCompra(){
    FacesMessage mensagem = null;

    HttpSession session = (HttpSession)FacesContext.getCurrentInstance()
        .getExternalContext().getSession(false);

    cliente = (Cliente) session.getAttribute("cliente");

    if(cliente == null){
        mensagem = new FacesMessage("Faça seu login antes de concluir a compra");
        mensagem.setSeverity(FacesMessage.SEVERITY_WARN);
        FacesContext.getCurrentInstance().addMessage(null, mensagem);
        cad = false;
        return;
    }
    pedido = new Pedido();
    pedido.setCliente(cliente);
    pedido.setItens(carrinho.getItens());
    pedido.setStatus("Pedido Registrado");
    pedido.setValorTotal(carrinho.getValor());
    cad = true;
}

public String compraCartao(){
    finalizarCompra();
    if(cad == false){
        return "Carrinho";
    }
    pedido.setPagamento("Cartão");
    System.out.println("Compra feita com cartão");
    PedidoDao dao = null;
    try{
        dao = new PedidoDao();
    } catch(SQLException e){
        e.printStackTrace();
    }
    dao.incluir(pedido);
    carrinho = null;
    return "VerificarPedido";
}

public String compraBoleto(){
    finalizarCompra();
    if(cad == false){
        return "Carrinho";
    }
    pedido.setPagamento("Boleto");
    System.out.println("Compra feita no Boleto");
    PedidoDao dao = null;
    try{
        dao = new PedidoDao();
    } catch(SQLException e){
        e.printStackTrace();
    }
    dao.incluir(pedido);
    carrinho = null;
    return "VerificarPedido";
}
```

5. Agora volte ao **Carrinho.xhtml** e altere assim como mostrado abaixo:

```
<ui:define name="content">
    <script type="text/javascript"
        src="http://code.jquery.com/jquery.min.js"></script>
    <script type="text/javascript" src="resources/js/Pgto.js"></script>
    <div class="jumbotron">
        .
        .
        .
    </div>
    <div class="container">
        <p:panel rendered="#{carrinhoBean.carrinhoVazio}">
            <table class="table cart-items">
                .
                .
                .
            </table>
            <div class="forma-pagamento">
                <div class="cartao box">
                    <p:commandButton value="Finalizar compra"
                        action="#{pedidoBean.verificarPedido}"
                        actionListener="#{carrinhoBean.compraCartao()}" />
                </div>
                <div class="boleto box">
                    <p:commandButton value="Finalizar Compra"
                        action="#{pedidoBean.verificarPedido}"
                        actionListener="#{carrinhoBean.compraBoleto()}" />
                </div>
            </div>
            <table class="table total-compra">
                .
                .
                .
            </table>
        </p:panel>
    </div>
</ui:define>
```

6. Teste novamente o carrinho de compras, veja se é possível completar uma compra sem estar logado. Veja também se sua compra foi salva na sua tabela, no banco de dados.

cod_item [PK] serial	qtd inte	cod_livro bigint	cod_pedido bigint	cod [PK] date	data_pedido date	status character varying(50)	pagamento character	cod bigint	total numeri
66	2	17	31	31	2016-06-08	Pedido Registrado	Cartão	1	25.00

Exercício 8 – Página VerificarPedido e VerificarItem

1. Para o cliente ter uma visibilidade de todos os pedidos feitos, datas, forma de pagamento e status do pedido, vamos criar umas páginas para isso. Crie o arquivo **VerificarPedido.xhtml** e implemente assim como mostrado abaixo.

```
<div class="container">
    <div class="verifica-pedidos">
        Pedidos Realizados
    </div>
    <div class="table-pedidos">
        <h:form rendered="#{not empty user}">
            <p:dataTable value="pedidos" var="pedido" rows="15">
```

```

paginator="true">
<p:column headerText="Código do pedido" width="180" style="text-align:center">
    <h:commandLink value="codigo"
        action="#{verificarItens(pedido.codigo)}"/>
</p:column>
<p:column headerText="Data do pedido" width="180" style="text-align:center">
    <h:outputText value="dataPedido">
        <f:convertDateTime pattern="dd/MM/yyyy" />
    </h:outputText>
</p:column>
<p:column headerText="Pagamento" style="text-align:center">
    <h:outputText value="pagamento" />
</p:column>
<p:column headerText="Status" style="text-align:center">
    <h:outputText value="status" />
</p:column>
</p:dataTable>
</h:form>
</div>
</div>

```



Código do pedido	Data do pedido	Pagamento	Status
31	08/06/2016	Cartão	Pedido Registrado
32	08/06/2016	Boleto	Pedido Registrado
33	09/06/2016	Boleto	Pedido Registrado
34	09/06/2016	Cartão	Pedido Registrado
35	09/06/2016	Cartão	Pedido Registrado
36	09/06/2016	Cartão	Pedido Registrado
37	09/06/2016	Cartão	Pedido Registrado
38	09/06/2016	Cartão	Pedido Registrado
39	09/06/2016	Cartão	Pedido Registrado
40	09/06/2016	Cartão	Pedido Registrado
41	09/06/2016	Cartão	Pedido Registrado
42	09/06/2016	Cartão	Pedido Registrado
43	10/06/2016	Cartão	Pedido Registrado
44	10/06/2016	Cartão	Pedido Registrado
45	10/06/2016	Cartão	Pedido Registrado

detalhes. Segue um exemplo do código abaixo.

```

<div class="container">
    <p:panel header="Lista de Livros" id="painel-final">
        <table class="table cart-items">
            <thead>
                <tr>
                    <th class="produto">Produto</th>

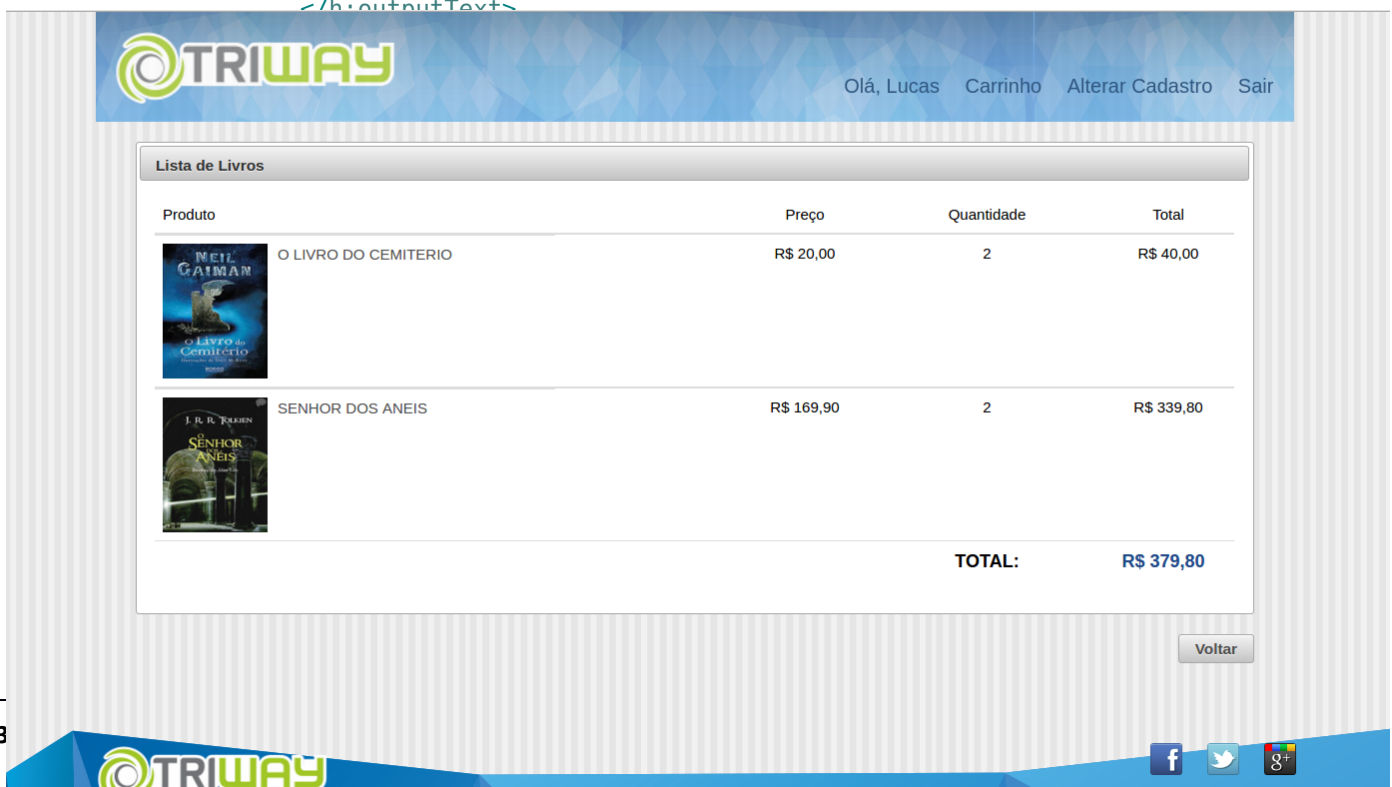
```

```



<th class="produto-preco">Preço</th>
<th class="quantidade">Quantidade</th>
<th class="preco-total">Total</th>
</tr>
</thead>
<ui:repeat value="#{pedidoBean.itens}" var="item">
<tbody id="tabela">
<tr class="produto-item">
<td class="produto-image">

</td>
<td class="produto-nome">
<h:form>
<h:commandLink
action="#{pesquisaBean.verLivro(item.codigo)}"
value="#{item.livro.titulo}" />
</h:form>
</td>
<td class="produto-preco">
<h:outputText value="#{item.livro.preco}">
<f:convertNumber currencyCode="BRL" type="currency" />
</h:outputText>
</td>
<td class="quantidade">
<h:outputText id="qtd" value="#{item.qtd}" />
</td>
<td class="preco-total" id="precoOut">
<h:outputText value="#{item.valor}" id="valor">
<f:convertNumber currencyCode="BRL" type="currency" />
</h:outputText>
</td>
</tr>
</tbody>
</ui:repeat>
<tfoot>
<tr>
<td class="total-fim">
TOTAL:
</td>
<td class="total-fim">
<h:outputText value="#{pedidoBean.item.total}" class="preco" >
<f:convertNumber type="currency" currencyCode="BRL" />
</h:outputText>

```



The screenshot shows the TRIWAY web application interface. At the top, there is a navigation bar with the TRIWAY logo and user links: "Olá, Lucas", "Carrinho", "Alterar Cadastro", and "Sair". Below the navigation bar, a section titled "Lista de Livros" displays a table of books. The table has four columns: "Produto", "Preço", "Quantidade", and "Total". Two books are listed: "O LIVRO DO CEMITERIO" by Neil Gaiman and "SENHOR DOS ANEIS" by J.R.R. Tolkien. The total price for the items is R\$ 379,80. A "Voltar" button is located at the bottom right of the table.

Produto	Preço	Quantidade	Total
 O LIVRO DO CEMITERIO	R\$ 20,00	2	R\$ 40,00
 SENHOR DOS ANEIS	R\$ 169,90	2	R\$ 339,80
TOTAL:			R\$ 379,80

Voltar