



WORKSHOP

JAVA WEB

www.3way.com.br

Seja um profissional aprendendo com profissionais

INTRODUÇÃO

O objetivo deste workshop é consolidar o conhecimento adquirido durante o curso. Continuaremos o software de gestão bancária, desenvolveremos o cadastro de cliente, vínculo de conta bancária ao cliente cadastrado e rotinas de um banco como saques, depósitos e transferência.

Utilizaremos o padrão de desenvolvimento MVC:

Banco-Modelo – as classes de entidade e persistência.

Banco-Servico – as classes de regra de negocio e relacionamento com a camada DAO.

Banco-WebView

- **Visão** – as JSP's.
- **Servlets** – as classes de controle

Exercícios

Exercício 1: Criar o CRUD de cliente.

Exercício 2: Criar Conta bancária.

Exercício 3: Rotinas bancárias (saques, depósitos e transferências).

Antes de começarmos, por questões de boas práticas, é recomendado colocar o .jar do postgres na pasta **lib** de *Banco-WebView* e adicioná-lo ao seu *build path*, já que esse é o nosso projeto empacotador. Portanto, ao colocar o .jar do postgres em *WebView*, *Banco-Servico* consegue reconhecê-lo.

Exercício 1: Criar o CRUD de cliente.

Altere o servlet Login. Crie os atributos conforme abaixo:

```
@WebServlet("/ManterCliente")
public class ManterCliente extends ApplicationController {

    private ClienteService service;

    private Cliente cliente;
```

1. Vamos criar uma instâncias para os atributos acima, iremos utilizar o método **init()**, para inicializar as variáveis.

```
@Override
public void init() throws ServletException {
    // TODO Auto-generated method stub
    super.init();

    //verifica se o objeto cliente é igual a null, se sim cria uma instancia para ele
    if (cliente == null){

        cliente = new Cliente();
    }

    //verifica se o objeto service é igual a null, se sim cria uma instancia para ele
```

```
        if (service == null){  
            service = new ClienteService();  
        }  
    }  
}
```

2. Altere o método doGet() do servlet **ManterCliente**, para que fique conforme abaixo.

```
//Remova o código anterior  
  
//Recupera a ação de acordo com o botão selecionado  
String action = request.getParameter("action");  
  
cliente.setNome(request.getParameter("nome"));  
cliente.setEndereco(request.getParameter("endereco"));  
cliente.setRg(request.getParameter("rg"));  
cliente.setTelefone(request.getParameter("telefone"));  
cliente.setCpf(request.getParameter("cpf"));  
  
if ("salvar".equals(action)){  
    salvarCliente(cliente);  
  
    this.gerarLog("Cadastro do Cliente: " + cliente.getNome() + ", Rg:" + cliente.getRg());  
  
    request.getSession().setAttribute("cliente", null);  
}  
else if ("alterar".equals(action)){  
    alterarCliente(cliente);  
  
    this.gerarLog("Alteração do Cliente: " + cliente.getNome() + ", Rg:" + cliente.getRg());  
  
    request.getSession().setAttribute("cliente", null);  
}  
else if ("carregar".equals(action)){  
    cliente = service.buscarClientePorCPF(request.getParameter("cpf"));  
  
    //joga os dados do cliente selecionado na sessão para carregar no formulario  
    request.getSession().setAttribute("cliente", cliente);  
}  
else if ("excluir".equals(action)){  
    excluirCliente(cliente);  
}  
else{  
    request.getSession().setAttribute("cliente", null);  
    request.getSession().setAttribute("msg", null);  
    request.getSession().setAttribute("conta", null);  
}  
  
//joga na sessão a lista de Clientes Cadastrados  
request.getSession().setAttribute("listaClientes", service.listarTodosClientes());  
  
response.sendRedirect("manterCliente.jsp");  
}
```

3. Altere a classe **AppController** adicionando o método responsável por exibir mensagens de aviso ou erro ao usuário, adicione o método abaixo do método gerarLog().

```
protected void exibirMensagem(final String mensagem){
```

```
}  
this.request.getSession().setAttribute("msg", mensagem);  
}
```

4. O método **salvarCliente**, persiste os dados no banco, usando o mesmo service do sistema desktop criado no modulo de O.O, veja que persiste os dados no banco e lança uma mensagem, também existe um tratamento de exceção caso os dados obrigatórios não sejam preenchidos.

```
private void salvarCliente(){  
    try {  
        service.salvar(cliente);  
        this.exibirMensagem("Ação realizada com sucesso!");  
    } catch (CamposObrigatoriosException e) {  
        this.exibirMensagem("Preencha os campos obrigatórios!");  
    }  
}
```

5. Crie o método alterar cliente, note que o método possui quase as mesmas particularidades do método salvar.

```
private void alterarCliente(){  
    try {  
        service.atualizar(cliente);  
        this.exibirMensagem("Ação realizada com sucesso!");  
    } catch (CamposObrigatoriosException e) {  
        this.exibirMensagem("Preencha os campos obrigatórios!");  
    }  
}
```

6. Crie abaixo de o método alterar o método excluir.

```
private void excluirCliente(){  
    service.excluir(cliente);  
    this.exibirMensagem("Ação realizada com sucesso!");  
}
```

7. Refatore a página **manterCliente.jsp**, adicione o código abaixo referente a máscara para os campos telefone e cpf no final do bloco `<head>`.

```
<script type="text/javascript">  
    JQuery(document).ready(function() {  
        JQuery("input[name='telefone']").mask("(99)9999-9999");  
        JQuery("input[name='cpf']").mask("999.999.999-99");  
    });  
</script>
```

8. Altere os arquivos de **properties** mapeando os atributos do formulário cliente, para o funcionamento da internacionalização.

```
message_pt_BR.properties
form.cliente.nome = Nome
form.cliente.telefone = Telefone
form.cliente.rg = Registro Geral
form.cliente.cpf = CPF
form.cliente.endereco = Endereço
btn.cliente.salvar = Salvar
btn.cliente.atualizar = Atualizar
btn.cliente.excluir = Excluir
btn.cliente.limpar = Limpar
btn.cliente.cancelar = Cancelar
title.modal.abrirConta = Abertura de conta
```

```
message_en_US.properties
form.cliente.nome = Name
form.cliente.telefone = Phone
form.cliente.rg = General record
form.cliente.cpf = CPF
form.cliente.endereco = Address
btn.cliente.salvar = Save
btn.cliente.atualizar = Update
btn.cliente.excluir = Remove
btn.cliente.limpar = Clean
btn.cliente.cancelar = Cancel
title.modal.abrirConta = Account opening
```

9. Refatore o formulário de cliente na página **manterCliente.jsp** para que fique conforme o código abaixo.

```
<div class="container">
    <div class="page-header">
        <h4>Banco: <span style="font-size: 14px; color: red;">
            ${agencia.nome}</span>
        </h4>
        <h4>Agência: <span style="font-size: 14px; color: red;">
            ${agencia.banco.nome}</span>
        </h4>
    </div>

    <div class="panel panel-default">
        <jsp:include page="/templates/menu.jsp"/>
        <c:if test="${msg != null}">
            <div class="alert alert-danger fade in" id="btnIndividual">
                <button type="button" class="close" data-dismiss="alert"
                    aria-hidden="true">x</button>
                <strong>${msg}</strong>
            </div>
        </c:if>

        <div class="panel-body">
            <div class="panel panel-primary">
                <div class="panel-heading">
```

```

<!-- <h3 class="panel-title">Cadastrar Cliente</h3> -->
</div>
<div class="panel-body">
    <form class="form-horizontal" action="ManterCliente" method="post"
        role="form">
        <div class="form-group">
            <label class="col-sm-2 control-label">
                <fmt:message key="form.cliente.nome"/>:
                <a style="color: red;">*</a>
            </label>
            <div class="col-sm-4">
                <input type="text" name="nome"
                    class="form-control" value="${cliente.nome}"
                    placeholder="<fmt:message
                        key="form.cliente.nome"/>"
                </div>
            <label class="col-sm-1 control-label">
                <fmt:message key="form.cliente.telefone"/>:
            </label>
            <div class="col-sm-4">
                <input type="text" name="telefone"
                    class="form-control"
                    placeholder="<fmt:message
                        key="form.cliente.telefone"/>"
                    value="${cliente.telefone}"
                </div>
        </div>
        <div class="form-group">
            <label class="col-sm-2 control-label">
                <fmt:message key="form.cliente.endereco"/>:
            </label>
            <div class="col-sm-10">
                <input type="text" name="endereco"
                    class="form-control"
                    value="${cliente.endereco}"
                    placeholder="<fmt:message
                        key="form.cliente.endereco"/>"
                </div>
        </div>
        <div class="form-group">
            <label class="col-sm-2 control-label">
                <fmt:message key="form.cliente.rg"/>:
            </label>
            <div class="col-sm-10">
                <input type="text" name="rg"
                    class="form-control"
                    value="${cliente.rg}"
                    placeholder="<fmt:message
                        key="form.cliente.rg"/>"
                </div>
        </div>
        <div class="form-group">
            <label class="col-sm-2 control-label">
                <fmt:message key="form.cliente.cpf"/>:
                <a style="color: red;">*</a>
            </label>
            <div class="col-sm-10">
                <input type="text" name="cpf"
                    value="${cliente.cpf}"
                    class="form-control"
                    placeholder="<fmt:message
                        key="form.cliente.cpf"/>"
                </div>
        </div>
        <span>Campos Obrigatórios</span>
        <div class="row">
            <p class="text-right" style="padding-right: 5%">
                <button type="reset" class="btn btn-primary">
                    <fmt:message key="btn.cliente.limpar"/>
                </button>
                <c:if test="${cliente == null}">
                    <button type="submit"
                        name="action"
                        value="salvar"
                        class="btn btn-primary">

```

```

        <fmt:message
            key="btn.cliente.salvar"/>
        </button>
    </c:if>
    <c:if test="${cliente != null}">
        <button type="submit" name="action"
            value="alterar"
            class="btn btn-primary">
            <fmt:message
                key="btn.cliente.salvar"/>
        </button>
        <button type="submit" name="action"
            value="excluir"
            class="btn btn-primary">
            <fmt:message
                key="btn.cliente.excluir"/>
        </button>
    </c:if>
</p>
</div>
</form>
</div>
</div>
<div class="panel panel-primary">
    <div class="panel-heading">
        <h3 class="panel-title">Clientes Cadastrados</h3> -->
    </div>
    <div class="table-responsive">
        <table class="table table-bordered table-hover">
            <thead>
                <tr>
                    <th>#</th>
                    <th>
                        <fmt:message key="form.cliente.nome"/>
                    </th>
                    <th>
                        <fmt:message key="form.cliente.endereco"/>
                    </th>
                    <th>
                        <fmt:message key="form.cliente.telefone"/>
                    </th>
                    <th>
                        <fmt:message key="form.cliente.endereco"/>
                    </th>
                    <th>
                        <fmt:message key="form.cliente.cpf"/>
                    </th>
                </tr>
            </thead>
            <tbody>
                <c:forEach items="${listaClientes}" var="cliente"
                    varStatus="index">
                    <tr>
                        <td>${index.count}</td>
                        <td>${cliente.nome}</td>
                        <td>${cliente.endereco}</td>
                        <td>${cliente.telefone}</td>
                        <td>${cliente.rg}</td>
                        <td>${cliente.cpf}</td>
                        <td style="width: 125px;">
                            [ <a href="/ManterCliente?action=carregar&cpf=${
                                cliente.cpf}"
                                style="font-size: 11px;">Carregar</a> ]
                        </td>
                    </tr>
                </c:forEach>
            </tbody>
        </table>
    </div>
</div>
</div>

```

```
</div>
<!-- /container -->
```

10. Reinicie o web container e acesse a aplicação, realize um cadastro de cliente.

Banco: 3way NetWorks!

Agência: Banco Java Brasil

Manter Cliente

Operações Bancárias

Nome:*

Teste 01

Telefone:

(62)9999-9999

Endereço:

Rua Teste, Setor Teste Goiania

Registro Geral:

9999999

CPF:*

999.999.999-01

Campos obrigatorios

Limpar

Salvar

#	Nome	Endereço	Telefone	Registro Geral	CPF
---	------	----------	----------	----------------	-----

Banco: 3way NetWorks!

Agência: Banco Java Brasil

Manter Cliente

Operações Bancárias

Ação realizada com sucesso!

Nome:*

Nome

Telefone:

Telefone

Endereço:

Endereço

Registro Geral:

Registro Geral

CPF:*

CPF

Campos obrigatorios

Limpar

Salvar

#	Nome	Endereço	Telefone	Registro Geral	CPF	
1	Teste 01	Rua Teste, Setor Teste Goiania	(62)9999-9999	9999999	999.999.999-01	[Carregar]

11. Alteração de clientes, clique no botão carregar.

Banco: 3way NetWorks!
Agência: Banco Java Brasil

[Manter Cliente](#) [Operações Bancárias](#)

Nome:* Telefone:

Endereço:

Registro Geral:

CPF:*

Campos obrigatorios

[Limpar](#) [Salvar](#) [Excluir](#)

#	Nome	Endereço	Telefone	Registro Geral	CPF	
1	Teste 01	Rua Teste, Setor Teste Goiania	(62)9999-9999	9999999	999.999.999-01	[Carregar]

12. Dados alterados.

Banco: 3way NetWorks!
Agência: Banco Java Brasil

[Manter Cliente](#) [Operações Bancárias](#)

Ação realizada com sucesso!

Nome:* Telefone:

Endereço:

Registro Geral:

CPF:*

Campos obrigatorios

[Limpar](#) [Salvar](#)

#	Nome	Endereço	Telefone	Registro Geral	CPF	
1	Teste 01 Alterado	Rua Teste, Setor Teste Goiania	(62)9999-9999	9999999	999.999.999-01	[Carregar]

Exercício 2: Gerenciar conta bancaria.

1. Crie um servlet com o nome **OperacoesBancarias**.

```
package threeway.projeto.servlets;  
@WebServlet("/OperacoesBancarias")  
public class OperacoesBancarias extends ApplicationController {
```

2. Crie os atributos conforme abaixo.

```
private Conta conta;  
private ContaService contaService;  
private TransacaoService transacaoService;
```

3. Inicialize os objetos do método **init()** da classe **HttpServlet**.

```
@Override  
public void init() throws ServletException {  
    // TODO Auto-generated method stub  
    super.init();  
  
    if(conta == null) {  
        conta = new Conta();  
    }  
  
    if(contaService == null) {  
        contaService = new ContaService();  
    }  
  
    if(transacaoService == null) {  
        transacaoService = new TransacaoService();  
    }  
}
```

4. Vamos implementar a funcionalidade “abrir conta”, adicione o código abaixo no servlet **OperacoesBancarias**.

```
protected void doGet(HttpServletRequest request,  
    HttpServletResponse response) throws ServletException, IOException {  
    //Recupera a conta que foi criada pelo usuário vinculado a sessão  
    Cliente clienteSession = (Cliente) request.getSession().getAttribute("cliente");  
    Conta contaSession = (Conta) request.getSession().getAttribute("conta");  
  
    if (contaSession == null) {  
        /*Recupera a conta que foi criada pelo usuario vinculado  
        a sessão e lança o objeto conta na sessão para ser  
        recuperado e montar os dados na página*/  
  
        request.getSession().setAttribute("conta",  
            contaService.buscaContaPorCliente(clienteSession));  
    }  
  
    Double valor = null;
```

```
String action = request.getParameter("action");

if ("abrirConta".equals(action)) {

    conta.setDataAbertura(new Date());
    conta.setSaldo(Double.valueOf(request.getParameter("saldoInicial")));
    conta.setNumero(Integer.valueOf(request.getParameter("numero")));
    conta.setTitular(clienteSession);
    conta.setAgencia((Agencia)request.getSession().getAttribute("agencia"));
    conta.setTipoConta(EnumTipoConta.CONTA_PESSOAL);
    abrirConta();
    request.getSession().setAttribute("conta",
        contaService.buscaContaPorCliente(clienteSession));
}

//Campo de data formatada no padrao dd/m/yyyy HH:mm, campo será
//exibido na tela de criar conta bancaria
request.getSession().setAttribute("dataFormatada",
    UtilData.formataData(new Date()));
response.sendRedirect("operacoesBancarias.jsp");
}
```

5. Agora adicionando o método de abertura de conta ainda no servlet **OperacoesBancarias**.

```
private void abrirConta() {

    try {
        contaService.salvar(conta);
        this.exibirMensagem("Ação Realizada com sucesso!");
    } catch (CamposObrigatoriosException e) {
        this.exibirMensagem("Preencha os dados obrigatorios");
    }
}
```

6. Altere o método **doPost()**, conforme abaixo.

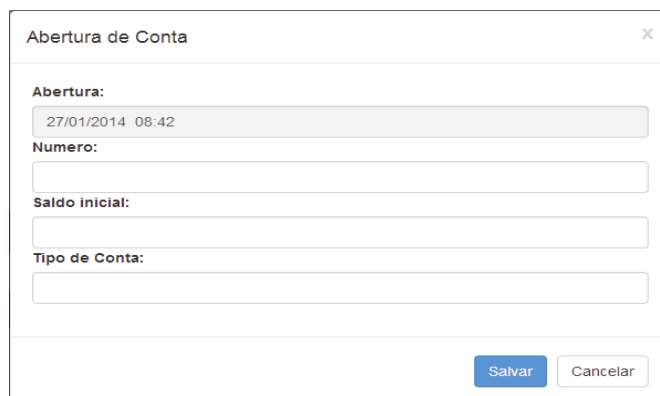
```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    doGet(request, response);
}
```

7. Agora no arquivo **operacoesBancarias.jsp** vamos criar um modal para implementação da funcionalidade abertura de conta conforme a imagem abaixo.

Adicione o botão “Abrir conta” ao lado dos botões de saque, depósito e transferência, conforme o código abaixo.

```
<button class="btn btn-sn btn-primary" data-toggle="modal"
    data-target="#modalConta" ${conta.identificador != null ? "disabled='disabled'" : "" }>
    Abrir Conta
</button>
```



Adicione o seguinte código abaixo da tag `<!-- /container -->`.

```
<!-- Modal Abrir Conta-->
<div class="modal fade" id="modalConta" tabindex="-1" role="dialog" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal"
          aria-hidden="true">x
        </button>
        <h4 class="modal-title" id="myModalLabel">
          <fmt:message key="title.modal.abrirConta"/>
        </h4>
      </div>
      <form action="OperacoesBancarias" method="post">
        <div class="modal-body">
          <label>Abertura</label>
          <div>
            <input type="text" disabled="disabled"
              class="form-control" value="${dataFormatada}">
          </div>
          <label>Numero:</label>
          <div>
            <input type="text" name="numero" class="form-control">
          </div>
          <label>Saldo inicial:</label>
          <div>
            <input type="text" name="saldoInicial"
              class="form-control">
          </div>
          <label>Tipo de Conta:</label>
          <div>
            <input type="text" name="tipoConta"
              class="form-control">
          </div>
        </div>
        <div class="modal-footer">
          <button type="submit" name="action" value="abrirConta"
            class="btn btn-primary">
            <fmt:message key="btn.cliente.salvar"/>
          </button>
          <button type="button" class="btn btn-default"
            data-dismiss="modal">
            <fmt:message key="btn.cliente.cancelar"/>
          </button>
        </div>
      </form>
    </div>
  </div>
</div>
<!-- /Modal abrir conta -->
```

8. Acesse a aba de operações bancárias. A aba fica habilitada após os dados do cliente serem carregados na tela “Manter Cliente”.



Manter Cliente Operações Bancárias

Ação realizada com sucesso!

Operações Bancárias

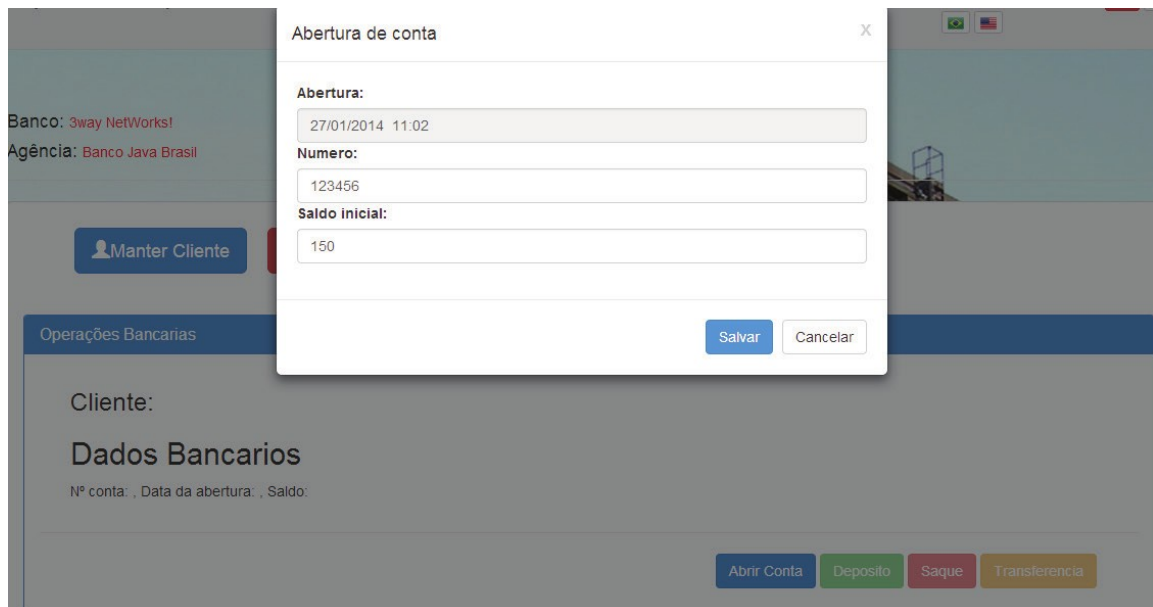
Cliente:

Dados Bancarios

Nº conta: , Data da abertura: , Saldo:

Abrir Conta Deposito Saque Transferencia

9. Clique no botão abrir conta, preencha os dados e clique em salvar.



Abertura de conta

Abertura:

27/01/2014 11:02

Numero:

123456

Saldo inicial:

150

Salvar Cancelar

Banco: 3way NetWorks!

Agência: Banco Java Brasil

Manter Cliente

Operações Bancárias

Cliente:

Dados Bancarios

Nº conta: , Data da abertura: , Saldo:

Abrir Conta Deposito Saque Transferencia

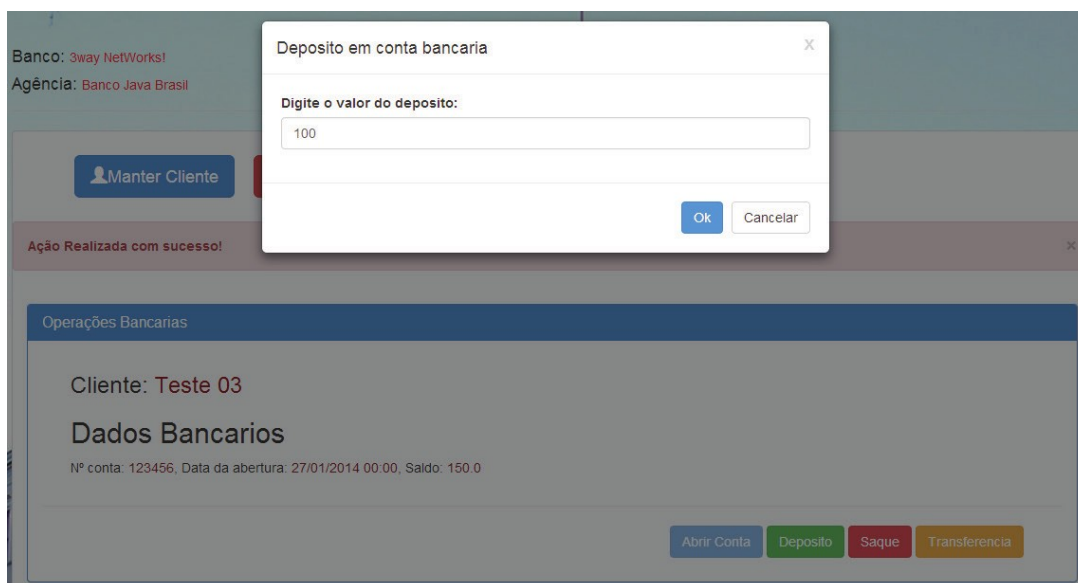
10. Veja abaixo os dados da conta que foram salvos na sessão e recuperados no jsp.



Exercício 3: Rotinas bancarias (saques, depósitos e transferências).


Use o conhecimento adquirido nos exercícios anteriores e implemente as demais funções bancarias. Utilize os mesmo métodos de depósito, transferência e saque da camada de serviço que foram utilizados no módulo de O.O para desenvolvimento da aplicação Desktop.

Deposito



Transações				
	Tipo Transação	Titular/ Conta Afetada	Data	Valor
1	DEPOSITO	Teste 03 / 123456	27/01/2014 00:00	100.0

Saque



Banco: 3way NetWorks!
Agência: Banco Java Brasil

Manter Cliente

Operação de depósito realizada com sucesso

Operações Bancárias

Saque em conta bancária

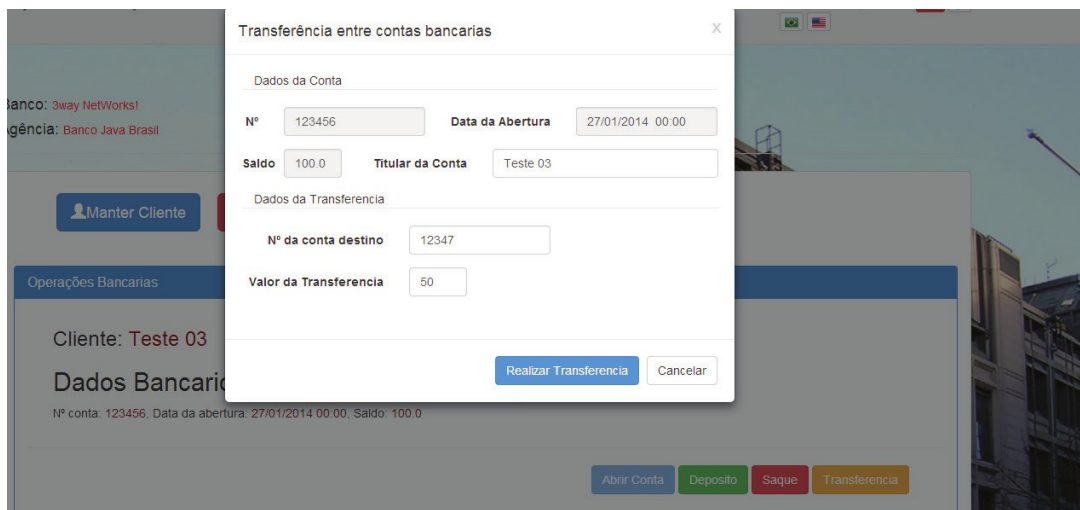
Digite o valor do saque:

150

Ok Cancelar

Transações				
	Tipo Transação	Titular/ Conta Afetada	Data	Valor
1	DEPOSITO	Teste 03 / 123456	27/01/2014 00:00	100.0
2	SAQUE	Teste 03 / 123456	27/01/2014 00:00	150.0

Transferência



Banco: 3way NetWorks!
Agência: Banco Java Brasil

Manter Cliente

Operações Bancárias

Cliente: Teste 03

Dados Bancários

Nº conta: 123456, Data da abertura: 27/01/2014 00:00, Saldo: 100.0

Abrir Conta Depósito Saque Transferência

Transferência entre contas bancárias

Dados da Conta

Nº 123456 Data da Abertura 27/01/2014 00:00

Saldo 100.0 Titular da Conta Teste 03

Dados da Transferencia

Nº da conta destino 12347

Valor da Transferencia 50

Realizar Transferencia Cancelar

Transações				
	Tipo Transação	Titular/ Conta Afetada	Data	Valor
1	DEPOSITO	Teste 03 / 123456	27/01/2014 00:00	100.0
2	SAQUE	Teste 03 / 123456	27/01/2014 00:00	150.0
3	TRANSFERENCIA	Teste 02 / 12347	27/01/2014 00:00	50.0