BootstrapJavaBank

Alunos:

Rodrigo Nogueira
Diego Camargo
Thalys Pacheco

Casos de Uso

Cadastrar Usuario e Conta

Nome: Cadastrar Usuario e Conta
Visão Geral: O admnistrador vai fazer o cadastro de um usuario na plataforma e cadastrar a conta do usuario
Pré-condições: O administrador está autenticado no sistema.
Lista de atores: Administrador
Cenário típico:
1 - O adminstrador acessa a página de Login.
2 - O adminstrador insere o cpf e a senha
3 - O adminstrador clica no botão de “Entrar”
4 - O sistema abre a tela da home da plataforma.
5 - O adminstrador clica em Cadastrar Usuario
6 - O sistema abre a tela Cadastrar Usuario.
7 - O adminstrador insere o cpf, nome e email.
8 - O adminstrador seleciona o tipo de cliente
9 - O adminstrador clica no botão de “Cadastrar”
10 - O sistema retorna para tela da home.
11 - O adminstrador clica em Cadastrar Conta
12 - O sistema abre a tela Cadastrar Conta.
13 - O adminstrador insere o cpf do usuario cadastrado
14 - O adminstrador seleciona o tipo de conta
15 - O adminstrador clica em “Cadastrar”
16 - O sistema retorna para tela da home.
Pós-condições:
Um novo usuário foi cadastrado no sistema.
Uma nova conta foi associada ao usuário.

Executar deposito e saque

Nome: Executar deposito e saque
Visão Geral: O cliente vai se logar e vai fazer o deposito no banco e vai fazer o saque
Pré-condições: O cliente ter um usuario e uma conta cadastrado
Lista de atores: Cliente
Cenário típico:
1 - O cliente acessa a página de Login.
2 - O cliente insere o cpf e a senha
3 - O cliente clica no botão de “Entrar”
4 - O sistema abre a tela da home da plataforma.
5 - O cliente clica em “Deposito”
6 - O sistema abre a tela de Deposito.
7 - O cliente insere o valor de deposito
8 - O cliente clica em “Enviar”
9 - O sistema retorna para tela da home.
10 - O cliente clica em “Saque”
11 - O sistema abre a tela de Saque
12 - O cliente insere o valor de saque
13 - O cliente clica em “Enviar”
14 - O sistema abre a tela da home da plataforma.
Pós-condições:
As operações de depósito e saque foram realizadas com sucesso.
O saldo do cliente foi atualizado de acordo.

UsuarioDAO

```
package model;
import entidade.UsuarioEntidade;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.util.ArrayList;

public class UsuarioDAO implements DAO<UsuarioEntidade>{
    @Override
    public void  insert(UsuarioEntidade usuario){
        Conexao conexao = new Conexao();
        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("INSERT INTO usuarios (nome, tipo, senha, cpf, email)
VALUES(?, ?, ?, ?, ?)");
            sql.setString(1, usuario.getNome());
            sql.setString(2, usuario.getTipo());
            sql.setString(3, usuario.getSenha());
            sql.setString(4, usuario.getCpf());
            sql.setString(5, usuario.getEmail());
            sql.executeUpdate();
            } catch (SQLException e) {
                throw new RuntimeException("Query (inserir) incorreto " + e.getMessage());
            }finally{
                conexao.closeConexao();
        }
    }
```

```
@Override
    public void delete(int id) {
        Conexao conexao = new Conexao();
        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("DELETE FROM usuarios WHERE ID = ? ");
            sql.setInt(1, id);
            sql.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Query de delete (excluir) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
    }
```

```
@Override
    public void update(UsuarioEntidade usuario)  {
        Conexao conexao = new Conexao();

        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("UPDATE usuarios SET nome = ?, cpf = ?, tipo = ?, senha = ?,
email = ?  WHERE ID = ? ");
            sql.setString(1, usuario.getNome());
            sql.setString(2, usuario.getCpf());
            sql.setString(3, usuario.getTipo());
            sql.setString(4, usuario.getSenha());
            sql.setString(5, usuario.getEmail());
            sql.setInt(6, usuario.getId());
            sql.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Query de update (alterar) incorreta: "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
    }
```

```
@Override
    public ArrayList<UsuarioEntidade> getAll() {
        ArrayList<UsuarioEntidade> meusUsuarios = new ArrayList();
        Conexao conexao = new Conexao();
        try {
            String selectSQL = "SELECT * FROM usuarios order by nome";
            PreparedStatement preparedStatement =  conexao.getConexao().prepareStatement(selectSQL);
            ResultSet resultado = preparedStatement.executeQuery();
            if (resultado != null) {
                while (resultado.next()) {
                    int id = Integer.parseInt(resultado.getString("id"));
                    String nome = (resultado.getString("NOME"));
                    String tipo = (resultado.getString("TIPO"));
                    String cpf = (resultado.getString("CPF"));
                    String email = (resultado.getString("EMAIL"));
                    UsuarioEntidade usuario = new UsuarioEntidade( nome, tipo, cpf, email);
                    usuario.setId(id);
                   meusUsuarios.add(usuario);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException("Query de select (ListaDeUsuarios) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
        return meusUsuarios;
    }
```

```
@Override
    public UsuarioEntidade get(int id) {
        Conexao conexao = new Conexao();
        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM Usuarios WHERE ID = ?");
            sql.setInt(1, id);
            ResultSet resultado = sql.executeQuery();
            if(resultado!=null){
                while(resultado.next()){
                   String nome = (resultado.getString("NOME"));
                    String tipo = (resultado.getString("TIPO"));
                    String cpf = (resultado.getString("CPF"));
                    String email = (resultado.getString("EMAIL"));
                    UsuarioEntidade usuario = new UsuarioEntidade( nome, tipo, cpf, email);
                    usuario.setId(id);
                    return usuario;
                }
            }
            return null;

        } catch (SQLException e) {
            throw new RuntimeException("Query de select (get usuario) incorreto "  + e.getMessage());
        } finally{
            conexao.closeConexao();
        }
    }
```

```
public UsuarioEntidade getByCpf(String cpf) {
        Conexao conexao = new Conexao();

        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM usuarios WHERE cpf = ?");
             sql.setString(1, cpf);
            ResultSet resultado = sql.executeQuery();
            if(resultado!=null){
                while(resultado.next()){
                    int id = (resultado.getInt("ID"));
                    String nome = (resultado.getString("NOME"));
                    String tipo = (resultado.getString("TIPO"));
                    String email = (resultado.getString("EMAIL"));
                    UsuarioEntidade usuario = new UsuarioEntidade( nome, tipo, cpf, email);
                    usuario.setId(id);
                    return usuario;
                }
            }
            return null;

        } catch (SQLException e) {
            throw new RuntimeException("Query (Logar) incorreto "  + e.getMessage());
        } finally{
            conexao.closeConexao();
        }
    }
```

```
public UsuarioEntidade logar(String cpf, String password) {
        Conexao conexao = new Conexao();
        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM Usuarios WHERE cpf = ? and senha = ?");
             sql.setString(1, cpf);
            sql.setString(2, password);
            ResultSet resultado = sql.executeQuery();
            if(resultado!=null){
                while(resultado.next()){
                    int id = (resultado.getInt("ID"));
                    String nome = (resultado.getString("NOME"));
                    String tipo = (resultado.getString("TIPO"));
                    String email = (resultado.getString("EMAIL"));
                    UsuarioEntidade usuario = new UsuarioEntidade( nome, tipo, cpf, email);
                    usuario.setId(id);
                    return usuario;
                }
            }
            return null;

        } catch (SQLException e) {
            throw new RuntimeException("Query (Logar) incorreto "  + e.getMessage());
        } finally{
            conexao.closeConexao();
        }
    }
```

Conta DAO

```
package model;
import entidade.ContaEntidade;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class ContaDAO implements DAO<ContaEntidade>{

    @Override
    public void  insert(ContaEntidade conta){
        Conexao conexao = new Conexao();
        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("INSERT INTO contas (idUsuario, tipo, saldo) VALUES(?, ?, ?)");
            sql.setInt(1, conta.getIdUsuario());
            sql.setString(2, conta.getTipo());
            sql.setDouble(3, conta.getSaldo());
            sql.executeUpdate();

        }catch (SQLException e) {
            throw new RuntimeException("Query (inserir) incorreto: " + e.getMessage());
        }finally{
            conexao.closeConexao();
        }
    }
}
```

```
@Override
    public void delete(int id) {
        Conexao conexao = new Conexao();
        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("DELETE FROM contas WHERE ID = ? ");
            sql.setInt(1, id);
            sql.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Query de delete (excluir) incorreta");
        } finally {
            conexao.closeConexao();
        }
    }
```

```
@Override
    public void update(ContaEntidade conta)  {
        Conexao conexao = new Conexao();

        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("UPDATE contas SET idUsuario = ?, tipo = ?, saldo = ?  WHERE
ID = ? ");
            sql.setInt(1, conta.getIdUsuario());
            sql.setString(2, conta.getTipo());
            sql.setDouble(3, conta.getSaldo());
            sql.setInt(4, conta.getId());
            sql.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Query de update (alterar) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
    }
```

```
@Override
    public ArrayList<ContaEntidade> getAll() {
        ArrayList<ContaEntidade> minhasContas = new ArrayList();
        Conexao conexao = new Conexao();¡
        try {
            String selectSQL = "SELECT * FROM contas order by id";
            PreparedStatement preparedStatement =  conexao.getConexao().prepareStatement(selectSQL);
            ResultSet resultado = preparedStatement.executeQuery();¡
            if (resultado != null) {
                while (resultado.next()) {
                    int id = (resultado.getInt("ID"));
                    int idUsuario = (resultado.getInt("IDUSUARIO"));
                    String tipo = (resultado.getString("TIPO"));
                    double saldo = (resultado.getDouble("SALDO"));
                    ContaEntidade conta = new ContaEntidade(idUsuario, tipo, saldo);
                    conta.setId(id);
                    minhasContas.add(conta);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException("Query de select (ListaDeUsuarios) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
        return minhasContas;
    }
```

```
@Override
    public ContaEntidade get(int id) {
        Conexao conexao = new Conexao();

        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM contas WHERE ID = ?");
            sql.setInt(1, id);
            ResultSet resultado = sql.executeQuery();
            if(resultado!=null){
                while(resultado.next()){
                    int idUsuario = (resultado.getInt("IDUSUARIO"));
                    String tipo = (resultado.getString("TIPO"));
                    double saldo = (resultado.getDouble("SALDO"));
                    ContaEntidade conta = new ContaEntidade(idUsuario, tipo, saldo);
                    conta.setId(id);
                    return conta;
                }
            }
            return null;

        } catch (SQLException e) {
            throw new RuntimeException("Query de select (get usuario) incorreto "  + e.getMessage());
        } finally{
            conexao.closeConexao();
        }
    }
```

```
public ArrayList<ContaEntidade> getContasByUser(int idUsuario) {
        ArrayList<ContaEntidade> minhasContas = new ArrayList();
        Conexao conexao = new Conexao();

        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM contas where idUsuario = ? Order By id");
            sql.setInt(1, idUsuario);
            ResultSet resultado = sql.executeQuery();

            if (resultado != null) {
                while (resultado.next()) {
                    int id = (resultado.getInt("ID"));
                    String tipo = (resultado.getString("TIPO"));
                    double saldo = (resultado.getDouble("SALDO"));
                    ContaEntidade conta = new ContaEntidade(idUsuario, tipo, saldo);
                    conta.setId(id);
                    minhasContas.add(conta);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException("Query de select (ListaDeUsuarios) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
        return minhasContas;
    }
```

TransaçaoDAO

```
package model;

import entidade.TransacaoEntidade;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;

public class TransacaoDAO implements DAO<TransacaoEntidade>{
    @Override
    public void  insert(TransacaoEntidade transacoes){
        Conexao conexao = new Conexao();
        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("INSERT INTO transacoes (idEmissor, idDestinatario, valor,
dataTransacao) VALUES(?, ?, ?, NOW())");
            sql.setInt(1, transacoes.getIdEmissor());
            sql.setInt(2, transacoes.getIdDestinatario());
            sql.setDouble(3, transacoes.getValor());
            sql.executeUpdate();
        }catch (SQLException e) {
            throw new RuntimeException("Query (inserir) incorreto: " + e.getMessage());
        }finally{
            conexao.closeConexao();
        }
    }
```

```
@Override
    public void delete(int id) {
        Conexao conexao = new Conexao();
        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("DELETE FROM transacoes WHERE ID = ? ");
            sql.setInt(1, id);
            sql.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Query de delete (excluir) incorreta");
        } finally {
            conexao.closeConexao();
        }
    }
```

```
@Override
    public void update(TransacaoEntidade transacoes)  {
        Conexao conexao = new Conexao();

        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("UPDATE contas SET idEmissor = ?, idDestinatario = ?, valor =
?  WHERE ID = ? ");
            sql.setInt(1, transacoes.getIdEmissor());
            sql.setInt(2, transacoes.getIdDestinatario());
            sql.setDouble(3, transacoes.getValor());
            sql.executeUpdate();

        } catch (SQLException e) {
            throw new RuntimeException("Query de update (alterar) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
    }
```

```
@Override
    public ArrayList<TransacaoEntidade> getAll() {
        ArrayList<TransacaoEntidade> minhasTransacoes = new ArrayList();
        Conexao conexao = new Conexao();
        try {
            String selectSQL = "SELECT * FROM transacoes order by id";
            PreparedStatement preparedStatement =  conexao.getConexao().prepareStatement(selectSQL);
            ResultSet resultado = preparedStatement.executeQuery();
            if (resultado != null) {
                while (resultado.next()) {
                    int id = (resultado.getInt("ID"));
                    int idEmissor = (resultado.getInt("IDEMISSOR"));
                    int idDestinatario = (resultado.getInt("IDDESTINATARIO"));
                    double valor = (resultado.getDouble("VALOR"));
                    Date dataTransacao = (resultado.getDate("DATATRANSACAO"));
                    TransacaoEntidade transacoes = new TransacaoEntidade(idEmissor, idDestinatario, valor);
                    transacoes.setId(id);
                    transacoes.setDataTransacao(dataTransacao);
                   minhasTransacoes.add(transacoes);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException("Query de select (ListaDeUsuarios) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
        return minhasTransacoes;
    }
```

```
@Override
    public TransacaoEntidade get(int id) {
        Conexao conexao = new Conexao();

        try{
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM transacoes WHERE ID = ?");
            sql.setInt(1, id);
            ResultSet resultado = sql.executeQuery();
            if(resultado!=null){
                while(resultado.next()){
                    int idEmissor = (resultado.getInt("IDEMISSOR"));
                    int idDestinatario = (resultado.getInt("IDDESTINATARIO"));
                    double valor = (resultado.getDouble("VALOR"));
                    Date dataTransacao = (resultado.getDate("DATATRANSACAO"));
                    TransacaoEntidade transacoes = new TransacaoEntidade(idEmissor, idDestinatario, valor);
                    transacoes.setId(id);
                    transacoes.setDataTransacao(dataTransacao);
                   return transacoes;
                }
            }
            return null;

        } catch (SQLException e) {
            throw new RuntimeException("Query de select (get usuario) incorreto "  + e.getMessage());
        } finally{
            conexao.closeConexao();
        }
    }
```

```
public ArrayList<TransacaoEntidade> getTransacoesByConta(int idConta) {
        ArrayList<TransacaoEntidade> minhasTransacoes = new ArrayList();
        Conexao conexao = new Conexao();
        try {
            PreparedStatement sql = conexao.getConexao().prepareStatement("SELECT * FROM transacoes where idEmissor = ? || idDestinatario = ? Order By
id");
            sql.setInt(1, idConta);
            sql.setInt(2, idConta);
            ResultSet resultado = sql.executeQuery();
            if (resultado != null) {
                while (resultado.next()) {
                    int id = (resultado.getInt("ID"));
                    int idEmissor = (resultado.getInt("IDEMISSOR"));
                    int idDestinatario = (resultado.getInt("IDDESTINATARIO"));
                    double valor = (resultado.getDouble("VALOR"));
                    Date dataTransacao = (resultado.getDate("DATATRANSACAO"));
                    TransacaoEntidade transacoes = new TransacaoEntidade(idEmissor, idDestinatario, valor);
                    transacoes.setId(id);
                    transacoes.setDataTransacao(dataTransacao);
                   minhasTransacoes.add(transacoes);
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException("Query de select (ListaDeUsuarios) incorreta "  + e.getMessage());
        } finally {
            conexao.closeConexao();
        }
        return minhasTransacoes;
    }
```

Login Controller

```
import entidade.UsuarioEntidade;
import entidade.ContaEntidade;
import model.ContaDAO;
import model.UsuarioDAO;

import java.io.IOException;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet(urlPatterns = {"/login"})
public class login extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.getSession().invalidate();
        RequestDispatcher rd = request.getRequestDispatcher("/views/login/index.jsp");
        rd.forward(request, response);
    }
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String cpf = request.getParameter("cpf");
        String password = request.getParameter("password");
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        if(cpf.isEmpty() || password.isEmpty()){
            RequestDispatcher rd = request.getRequestDispatcher("/views/login/index.jsp");
            rd.forward(request, response);
        }else{
            UsuarioEntidade usuario = usuarioDAO.logar(cpf, password);
            if(usuario!=null){
              HttpSession session = request.getSession();
               session.setAttribute("usuario",usuario);
               if("admin".equals(usuario.getTipo())){
                   RequestDispatcher rd = request.getRequestDispatcher("/views/admin/index.jsp");
                   rd.forward(request, response);
               }else{
                   ContaDAO contaDAO = new ContaDAO();
                   List<ContaEntidade> contas = contaDAO.getContasByUser(usuario.getId());
                   session.setAttribute("contas",contas);
                   RequestDispatcher rd = request.getRequestDispatcher("/views/home/index.jsp");
                   rd.forward(request, response);
              }
          }else{
               RequestDispatcher rd = request.getRequestDispatcher("/views/login/index.jsp");
               rd.forward(request, response);
            }
        }
    }
}
```

Transaçao Controller

```
import entidade.ContaEntidade;
import entidade.TransacaoEntidade;
import java.io.IOException;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import model.ContaDAO;
import model.TransacaoDAO;
@WebServlet(urlPatterns = {"/transacao"})
public class transacao extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        int idConta = Integer.parseInt(request.getParameter("idConta"));
        TransacaoDAO transacaoDAO = new TransacaoDAO();
        List<TransacaoEntidade> transacoes = transacaoDAO.getTransacoesByConta(idConta);
        if(!transacoes.isEmpty()){
            request.setAttribute("transacoes", transacoes);
            RequestDispatcher rd = request.getRequestDispatcher("/views/extrato/index.jsp");
            rd.forward(request, response);
        }else{
            RequestDispatcher rd = request.getRequestDispatcher("/views/home/index.jsp");
            rd.forward(request, response);
        }
    }
```

```
@Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String tipo = request.getParameter("tipo");
        int idConta = Integer.parseInt(request.getParameter("idConta"));
        int idUsuario = Integer.parseInt(request.getParameter("idUsuario"));
        double valor = Double.parseDouble(request.getParameter("valor"));
        TransacaoDAO transacaoDAO = new TransacaoDAO();
        ContaDAO contaDAO = new ContaDAO();
        switch(tipo){
            case "Deposito":
                TransacaoEntidade transacaoDeposito = new TransacaoEntidade(-1, idConta, valor);
                transacaoDAO.insert(transacaoDeposito);
                ContaEntidade contaDeposito =  contaDAO.get(idConta);
                double novoSaldoDeposito = contaDeposito.getSaldo() + valor;
                contaDeposito.setSaldo(novoSaldoDeposito);
                contaDAO.update(contaDeposito);
                break;
            case "Saque":
                ContaEntidade contaSaque =  contaDAO.get(idConta);
                double novoSaldoSaque = contaSaque.getSaldo() - valor;
                if(novoSaldoSaque < 0){
                    RequestDispatcher rd = request.getRequestDispatcher("/views/transacao/index.jsp?tipo=Saque&idConta=" + idConta);
                    rd.forward(request, response);
                    break;
                }
                TransacaoEntidade transacaoSaque = new TransacaoEntidade(idConta, -1, valor);
                transacaoDAO.insert(transacaoSaque);
                contaSaque.setSaldo(novoSaldoSaque);
                contaDAO.update(contaSaque);
                break;
```

```
            case "Transferencia":
                int idDestinatario = Integer.parseInt(request.getParameter("idDestinatario"));

                TransacaoEntidade transacaoTransferencia = new TransacaoEntidade(idConta, idDestinatario, valor);
                transacaoDAO.insert(transacaoTransferencia);

                ContaEntidade contaEmissor =  contaDAO.get(idConta);
                ContaEntidade contaDestinatario =  contaDAO.get(idDestinatario);

                double novoSaldoEmissor = contaEmissor.getSaldo() - valor;
                double novoSaldoDestinatario = contaDestinatario.getSaldo() + valor;

                contaEmissor.setSaldo(novoSaldoEmissor);
                contaDestinatario.setSaldo(novoSaldoDestinatario);

                contaDAO.update(contaEmissor);
                contaDAO.update(contaDestinatario);
                break;
        }
        HttpSession session = request.getSession();
        List<ContaEntidade> contas = contaDAO.getContasByUser(idUsuario);
        session.setAttribute("contas",contas);
        RequestDispatcher rd = request.getRequestDispatcher("/views/home/index.jsp");
        rd.forward(request, response);

    }
}
```