**UNIVERSIDADE ESTADUAL DE MARINGÁ**

**DEPARTAMENTO DE INFORMÁTICA**

**CIÊNCIA DA COMPUTAÇÃO**

**PROJETO DE SISTEMA DE SOFTWARE**

**PROFESSOR: DONIZETE CARLOS BRUZAROSCO**

**TRABALHO DE PROJETO DE IMPLEMENTAÇÃO DE SISTEMAS DE SOFTWARE (ISS)**

JOÃO PEDRO BALESTRA ZACARKIN RA: 83967

MATHEUS ALBERTO VAZ CECILIO RA: 85582

THIAGO DE OLIVEIRA LAVEZZO RA: 84753

THIAGO VIDAL DE MORAES RA: 82349

**MARINGÁ**

**01 DE FEVEREIRO DE 2016**

Sumário

[1 ª SPRINT: 3](#_Toc442218423)

[2ª SPRINT: 3](#_Toc442218424)

[Telas do sistema + Cronogramas: 4](#_Toc442218425)

[Thiago Vidal de Moraes: 4](#_Toc442218426)

[Matheus Alberto Vaz Cecilio: 5](#_Toc442218427)

[Thiago de Oliveira Lavezzo: 6](#_Toc442218428)

[João Pedro Balestra Zacarkin: 9](#_Toc442218429)

[Frameworks: 10](#_Toc442218430)

[Diagrama MVC – Primeira Sprint: 11](#_Toc442218431)

[Diagrama MVC – Segunda Sprint: 12](#_Toc442218432)

[Composto 12](#_Toc442218433)

[PRODUTO/RELATÓRIO 13](#_Toc442218434)

[PEDIDO/RELATÓRIO 14](#_Toc442218435)

[PAGAMENTO 15](#_Toc442218436)

[COMPRA 16](#_Toc442218437)

[Planejamento dos testes: 17](#_Toc442218438)

[Prazo – Cronograma: 17](#_Toc442218439)

[Recursos: 17](#_Toc442218440)

[Estratégia de testes: 17](#_Toc442218441)

[Teste Caixa Preta: 18](#_Toc442218442)

[Matheus Alberto Vaz Cecilio 18](#_Toc442218443)

[Thiago Vidal de Moraes 20](#_Toc442218444)

[João Pedro Balestra Zacarkin 21](#_Toc442218445)

[Thiago de Oliveira Lavezzo 24](#_Toc442218446)

[Teste Caixa Branca: 26](#_Toc442218447)

[Matheus Alberto Vaz Cecilio 26](#_Toc442218448)

[Thiago Vidal de Moraes: 28](#_Toc442218449)

[João Pedro Balestra Zacarkin 30](#_Toc442218450)

[Thiago de Oliveira Lavezzo 32](#_Toc442218451)

[Padrão de Projeto: 35](#_Toc442218452)

[Ajuda Online: 35](#_Toc442218453)

[Funcionalidade Web: 35](#_Toc442218454)

# 1 ª SPRINT:

Gerenciamento de Usuários e Autenticar Usuário (Thiago Moraes)

Gerenciamento de Clientes(Matheus Alberto)

Gerenciamento de Produtos (Thiago Lavezzo)

Gerenciamento de Fornecedor(João Pedro)

# 2ª SPRINT:

Gerenciamento de Pagamento (Thiago Moraes)

Gerar Relatórios, Gerenciamento de Compra de Produtos (Thiago Lavezzo)

Gerenciamento de Vendas - Pedidos (Matheus Alberto)

Simular Manipulação (João Pedro)

# Telas do sistema + Cronogramas:

## Thiago Vidal de Moraes:

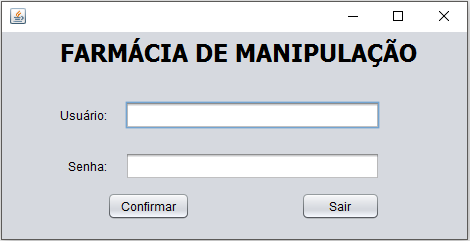
1. Sprint 1: RF002 Gerenciamento de Usuários

* 20h
* Telas:

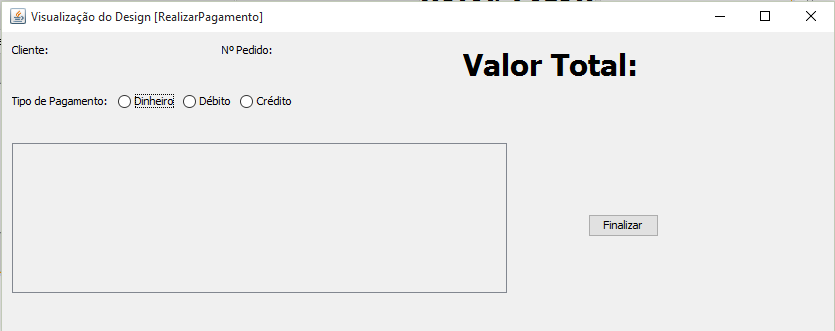
1. Sprint 1: RF003 Autenticar Usuário

* 4h
* Telas:



1. Sprint 2: RF009 Gerenciamento de Pagamento

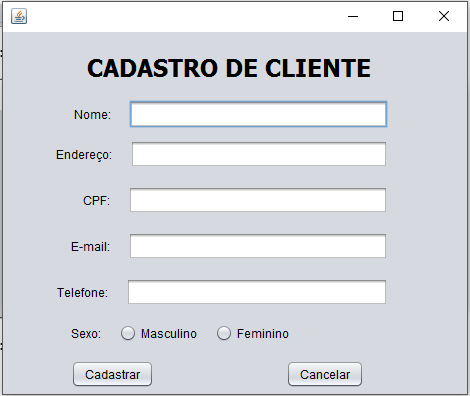
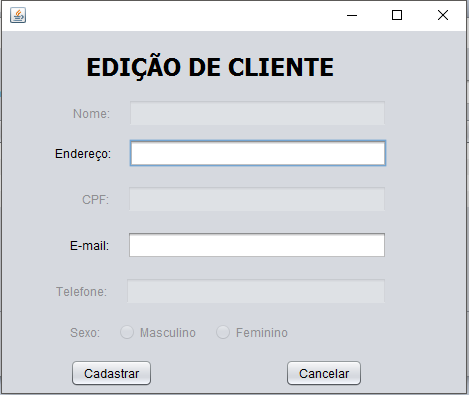
* 30h
* Telas:



## Matheus Alberto Vaz Cecilio:

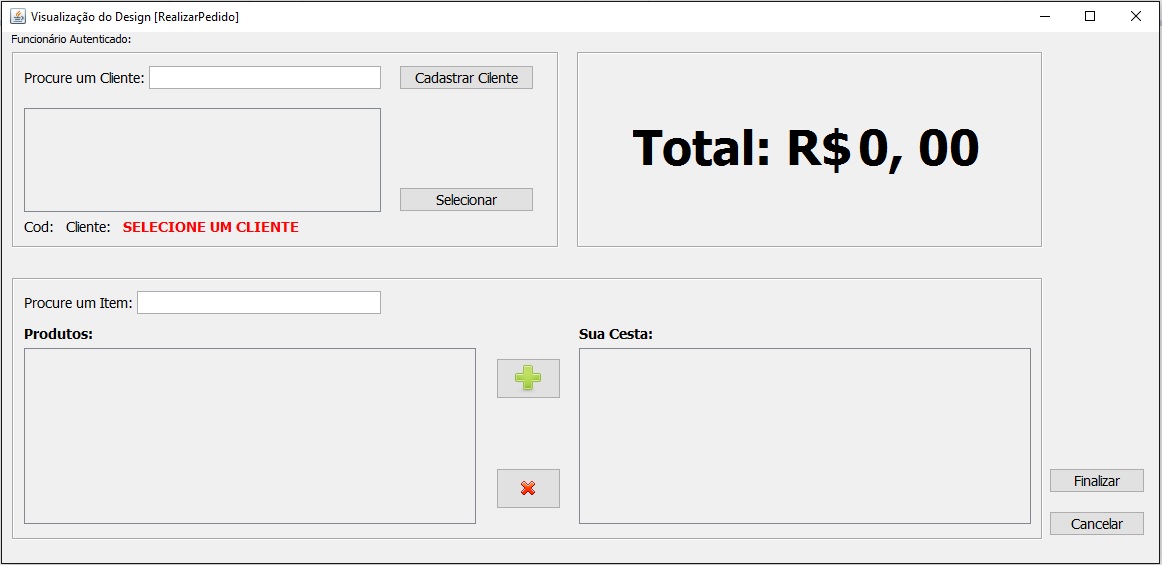
1. Sprint 1: RF001 Gerenciamento de Clientes

* 20h
* Telas:

1. Sprint 2: RF006 Gerenciamento de Vendas

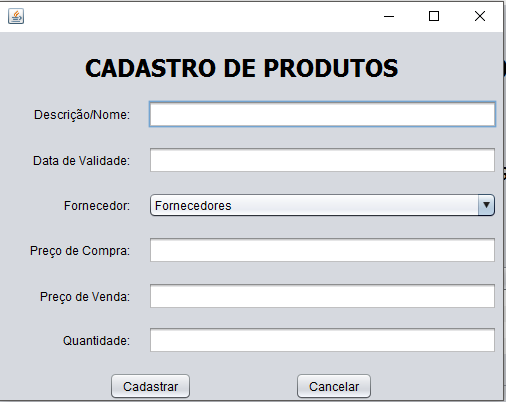
* 35h
* Telas:



## Thiago de Oliveira Lavezzo:

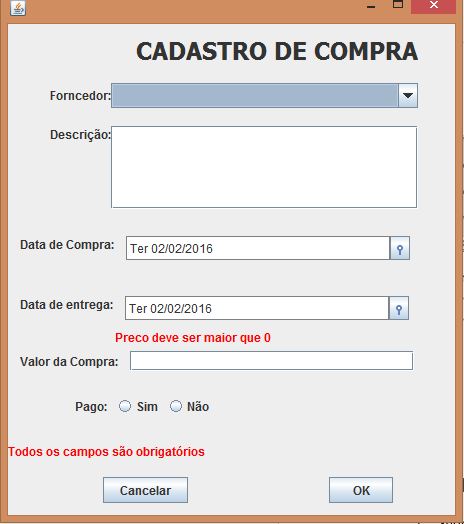
1. Sprint 1: RF004 Gerenciar Estoque

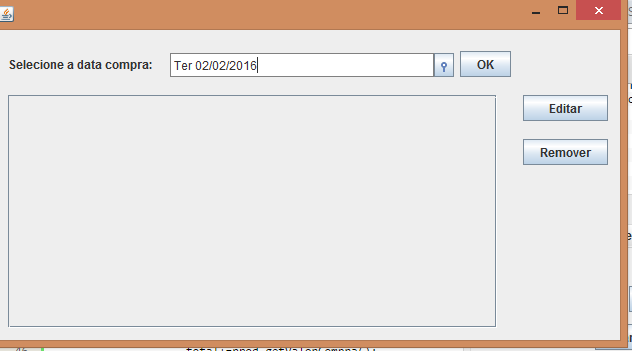
* 16h
* Telas:

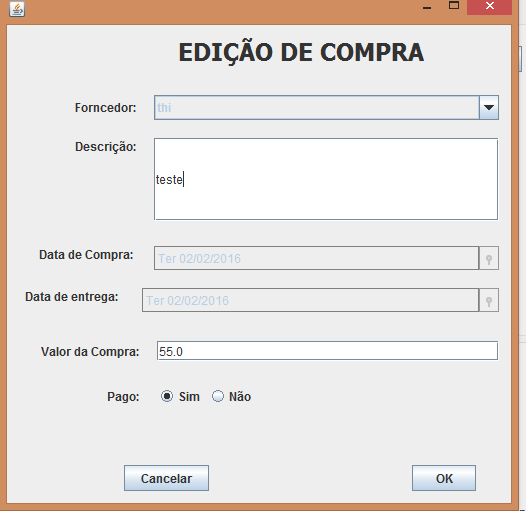
 

1. Sprint 2: Gerenciamento de Compra e Relatórios

* 30h
* Telas:

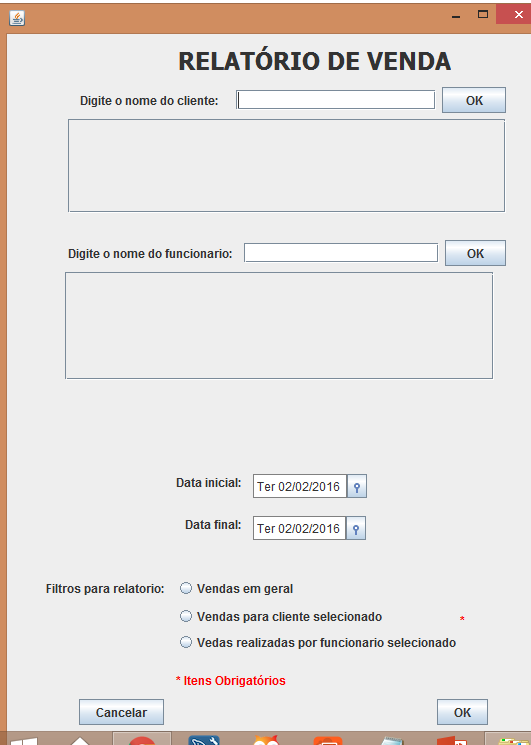
****

****

****







## João Pedro Balestra Zacarkin:

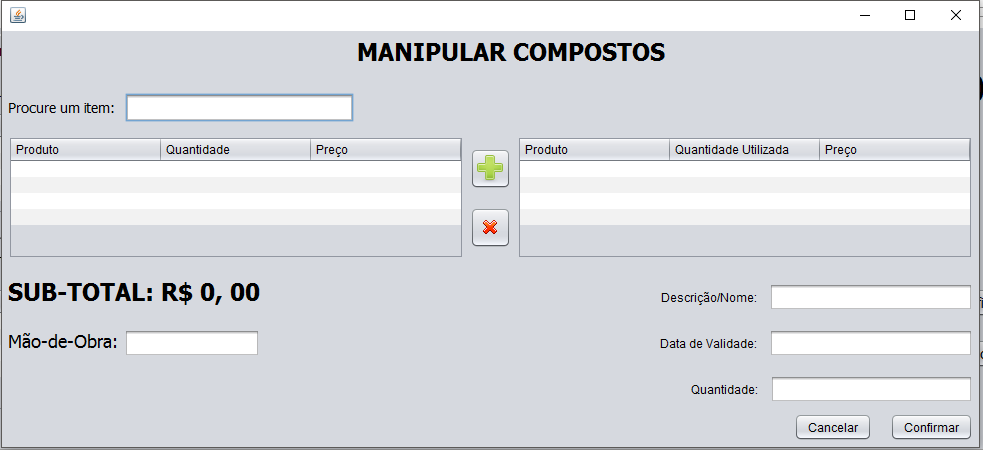
1. Sprint 1: RF007 Gerenciamento de Fornecedor

* 20h
* Telas:

1. Sprint 2: RF005 Simular Manipulação:

* 30h
* Telas:

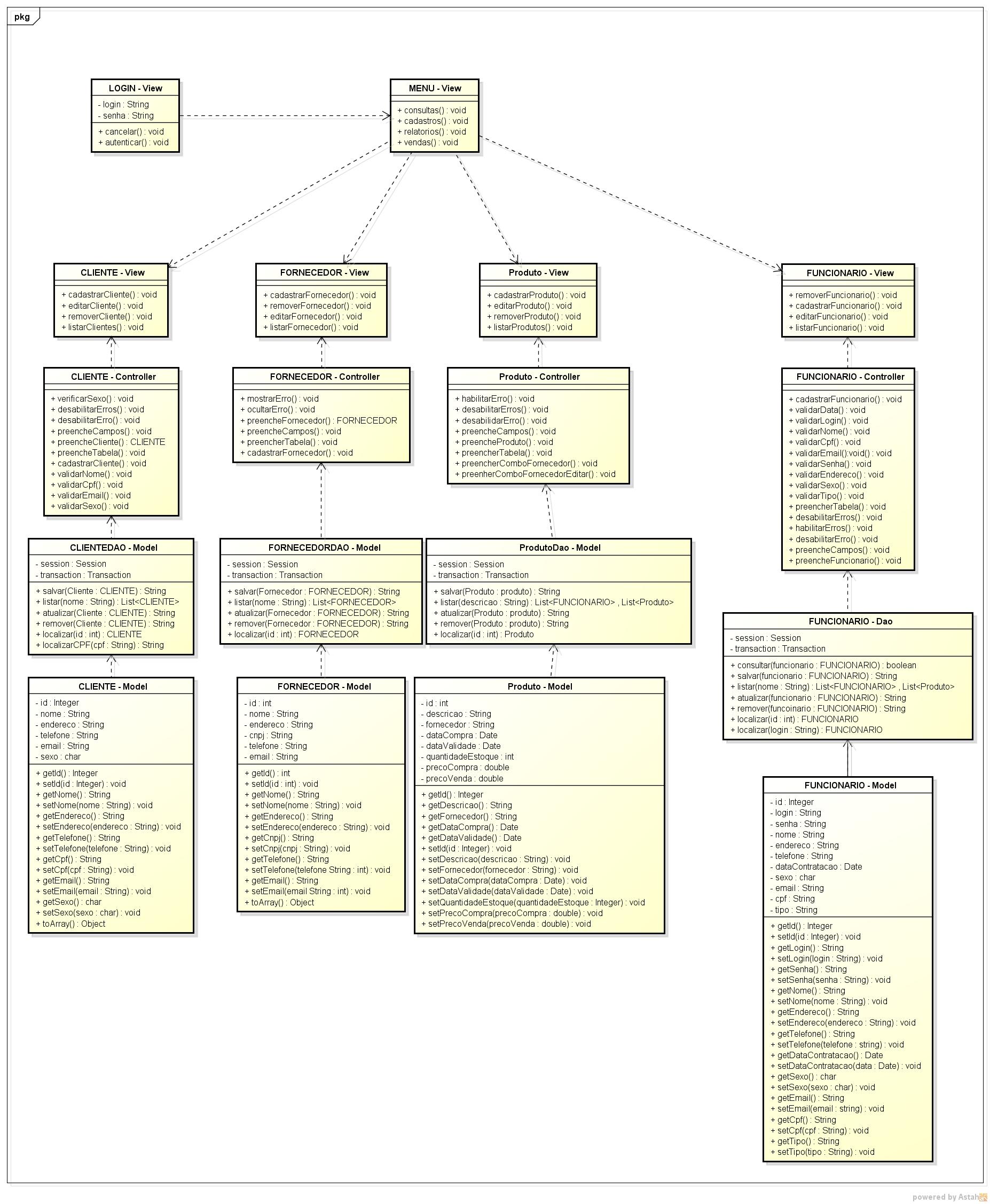


# Frameworks:

- Hibernate: “É um software livre de código aberto distribuído com licença LGPL. O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional (presente na maioria dos SGBDs). ”

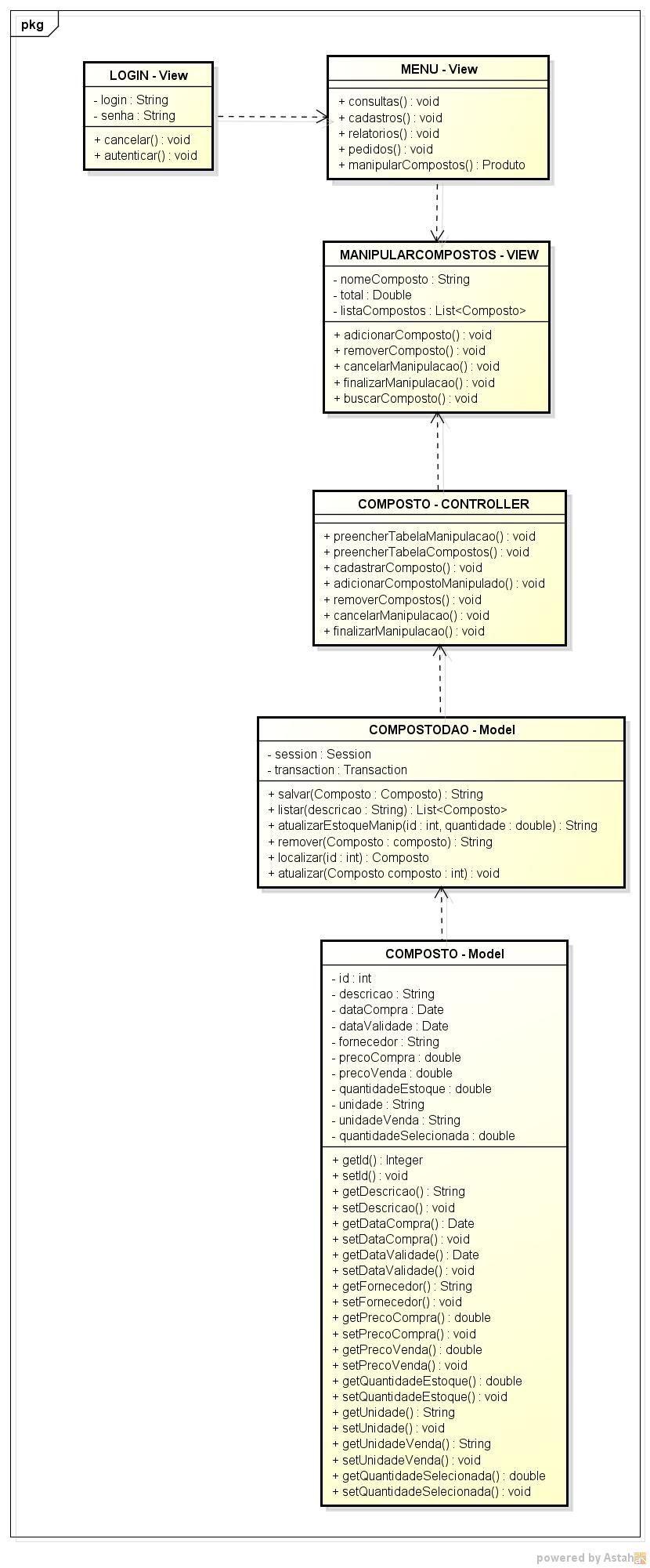
- Swing Java: “**Swing** é um *[widget toolkit](https://pt.wikipedia.org/wiki/Toolkit" \o "Toolkit)* para uso com o [Java](https://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o)). Ele é compatível com o [Abstract Window Toolkit (AWT)](https://pt.wikipedia.org/wiki/Abstract_Window_Toolkit), mas trabalha de uma maneira totalmente diferente. A API Swing procura renderizar/desenhar por conta própria todos os componentes, ao invés de delegar essa tarefa ao sistema operacional, como a maioria das outras APIs de interface gráfica trabalham. ”

# Diagrama MVC – Primeira Sprint:

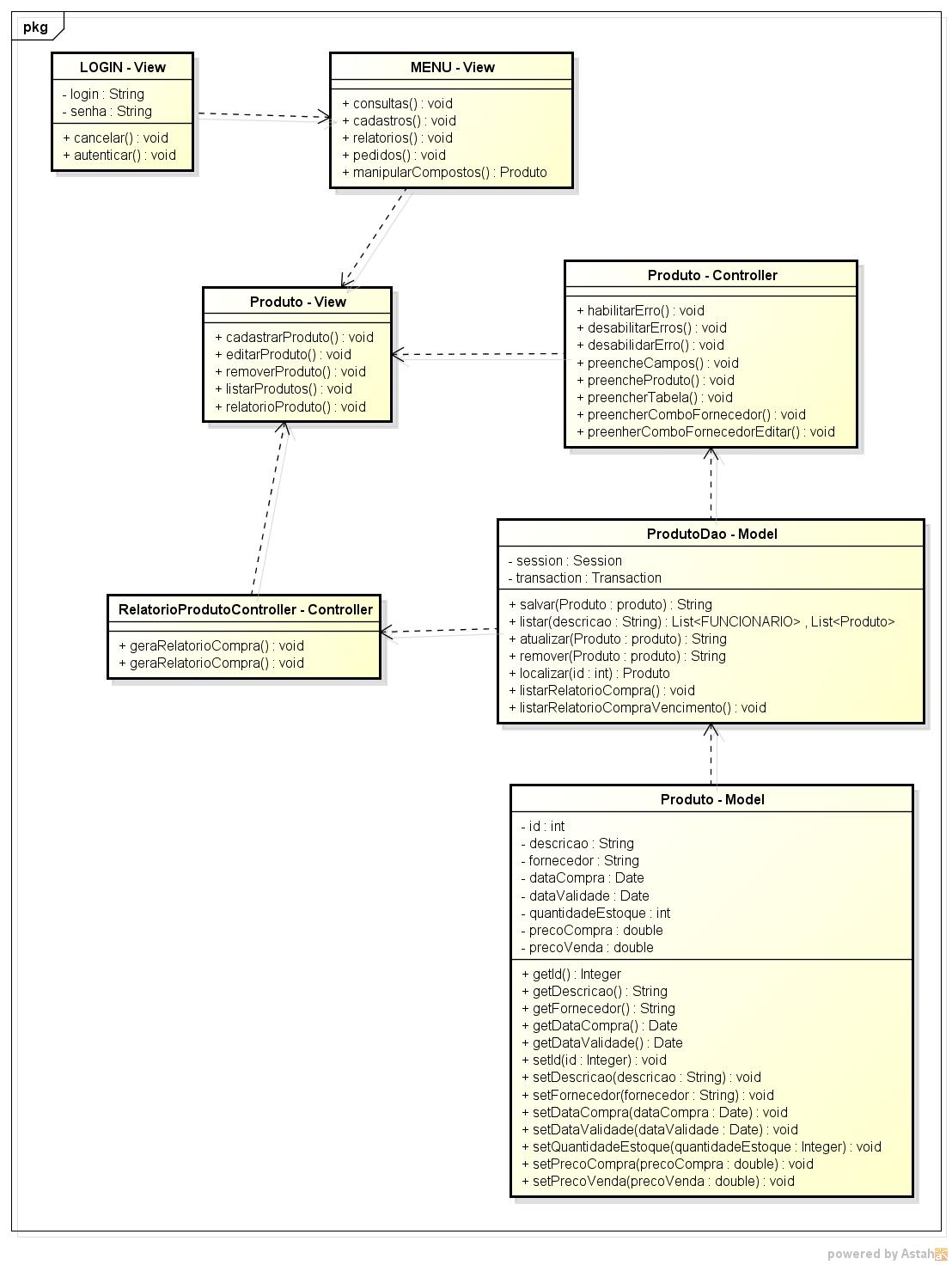


# Diagrama MVC – Segunda Sprint:

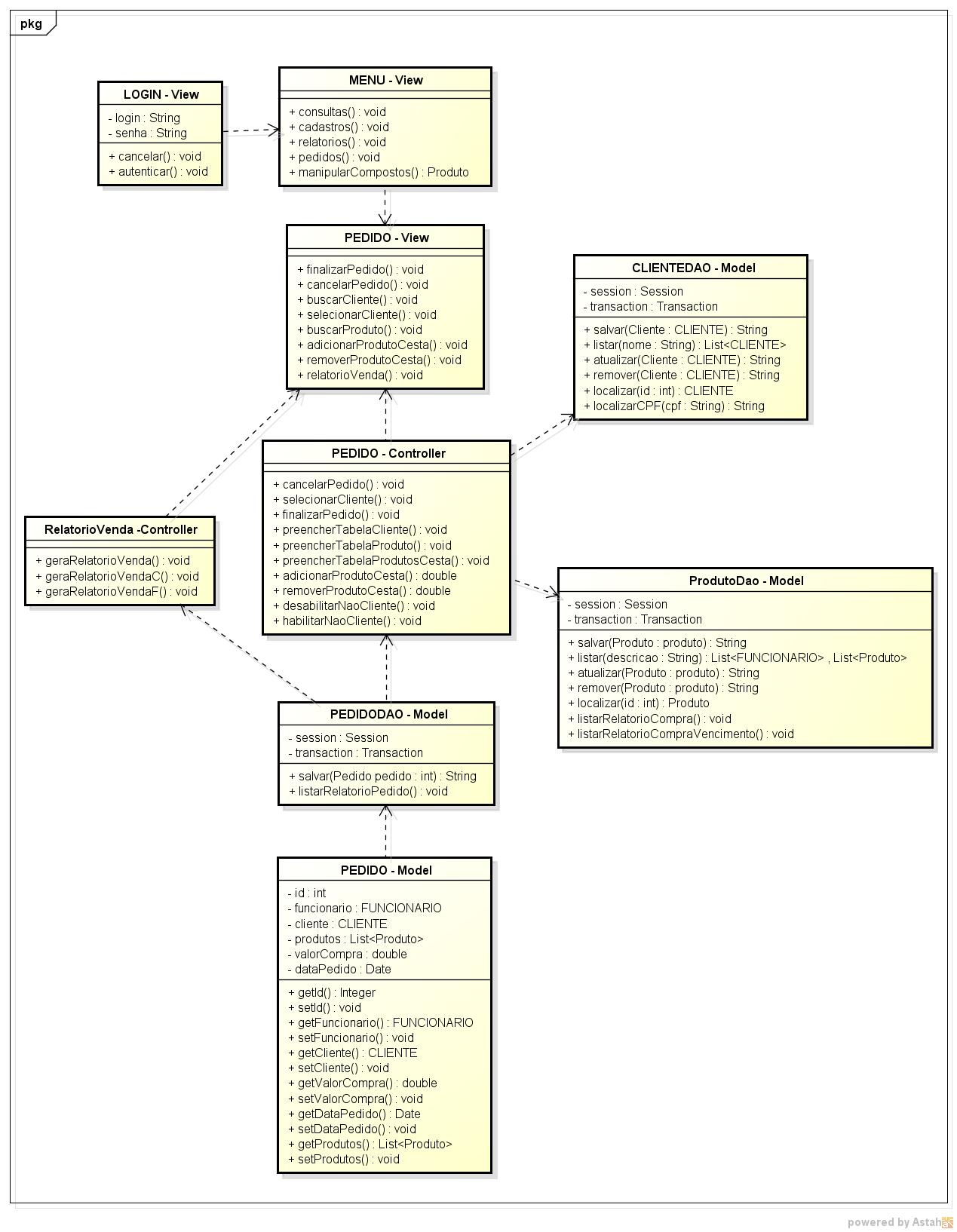
## Composto

****

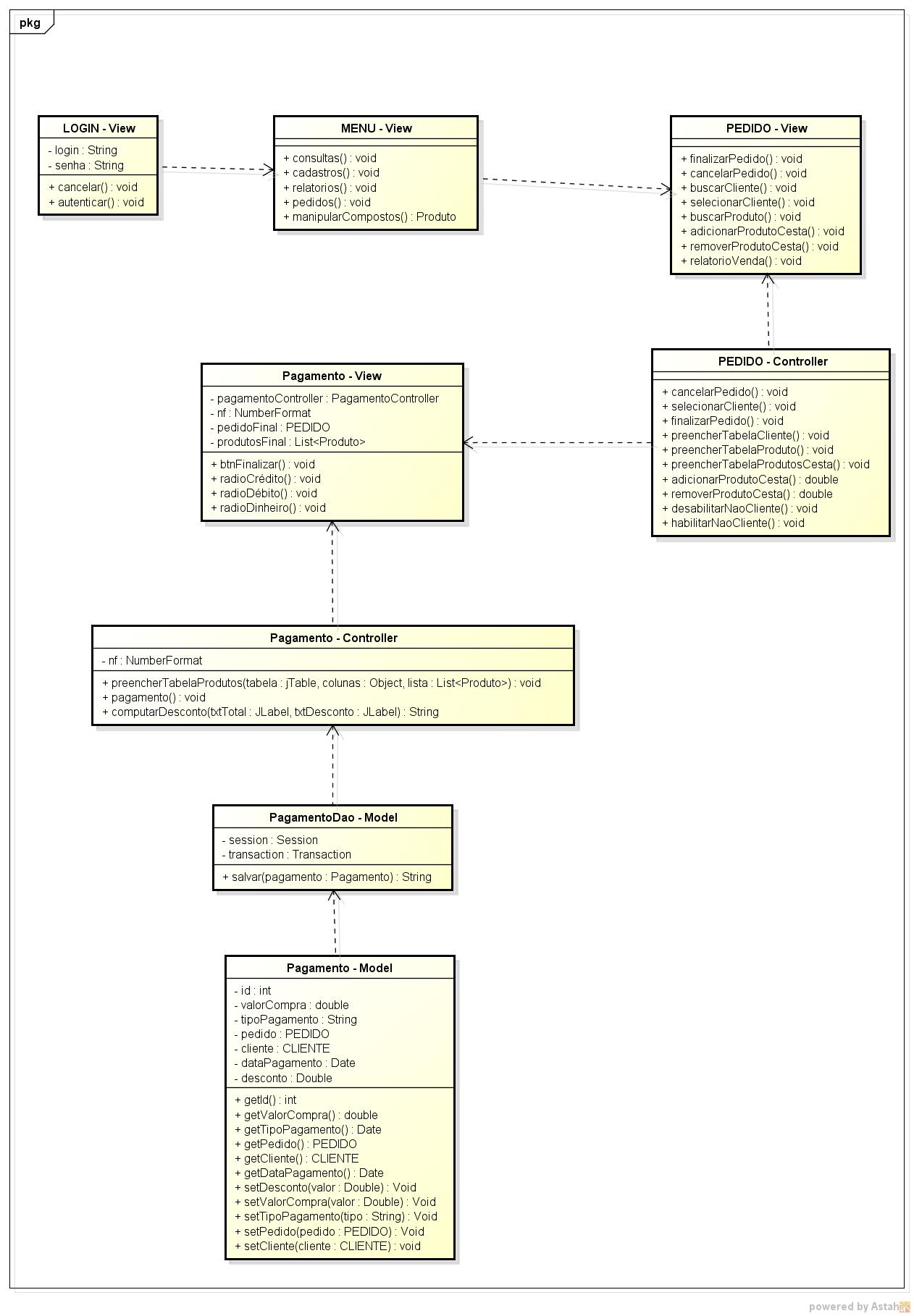
## PRODUTO/RELATÓRIO

****

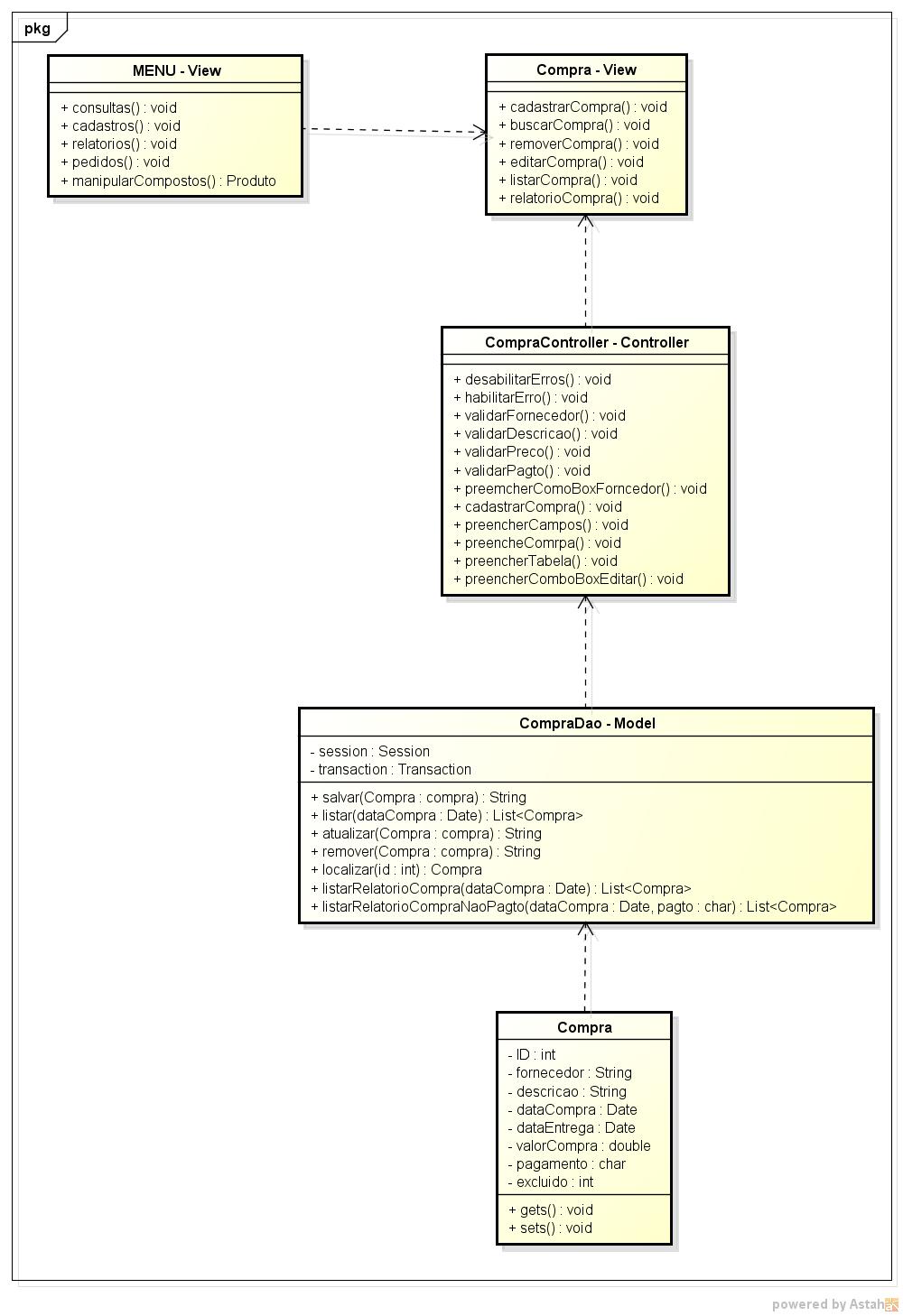
## PEDIDO/RELATÓRIO

****

## PAGAMENTO

****

## COMPRA

****

# Planejamento dos testes:

## Prazo – Cronograma:

Os testes serão estipulados e encerrados em cerca de 5 horas.

## Recursos:

Pessoas: Equipe de desenvolvimento do projeto.

Ferramentas: Softwares especializados e computadores da própria equipe.

## Estratégia de testes:

Os testes foram realizados usando o JUnit, que possui um plug-in na IDE Netbeans, para a realização dos testes.

Os testes foram realizados em alguns métodos de algumas classes, que pode ser observado ao importar o projeto no GitHub no endereço: <https://github.com/matheusalberto/TrabalhoISS>.

Cada aluno criou uma classe com seus testes na “Pacotes de Teste”.

# Teste Caixa Preta:

## Matheus Alberto Vaz Cecilio

* + **PARTICIONAMENTO DE EQUIVALÊNCIA**
  + Funcionalidade: Adicionar produto na cesta
  + Variável de entrada: Produto (idProduto).
  + Variáveis internas: não possui.
  + Variáveis de saída: Mensagem “Valor Inválido”, “Produto não encontrado” e o produto é aceito na cesta, que na tela se dá com a inserção do produto na tabela que corresponde à cesta de produtos para o pedido.
  + Observação: O produto, que é adquirido com base no id do produto, possui um atributo que é o seu preço de venda. Sendo assim, temos também como variável o preço de venda.
  + Classes de equivalência:
    - **Variável Preço de venda do Produto (preçoVenda).**
    - Classe 1: meuProduto.getPreçoVenda() <= 0 (Classe inválida)
    - Classe 2: meuProduto.getPreçoVenda() > 0 (Classe válida)
    - **Variável ID PRODUTO**
    - Classe 3: idProduto ϵ {Conjunto de todos os produtos existentes no banco de dados} (Classe válida)
    - Classe 4: idProduto ∉ {Conjunto de todos os produtos existentes no banco de dados} (Classe inválida)
    - **Variável Mensagem**
    - Classe 5: Valor inválido (Classe válida)
    - Classe 6: Produto não encontrado (Classe válida)
    - **Variável produto inserido na cesta**
    - Classe 7: Produto inserido na cesta (Classe válida).
  + Definição dos casos de teste:
    - Caso de teste 1:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA)** | **RESULTADO ESPERADO** |
| meuProduto.getPreçoVenda() = 25 (2) | Produto inserido na cesta |
| idProduto ϵ CONJUNTO DOS PRODUTOS (3) |  |

* + - Caso de teste 2:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA)** | **RESULTADO ESPERADO** |
| meuProduto.getPreçoVenda() = -1 (1) | Valor inválido |
| idProduto ϵ CONJUNTO DOS PRODUTOS (3) |  |

* + - Caso de teste 3:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA)** | **RESULTADO ESPERADO** |
| meuProduto.getPreçoVenda() = 25 (2) | Produto não encontrado |
| idProduto ∉ CONJUNTO DOS PRODUTOS (4) |  |

* + - Caso de teste 4:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA)** | **RESULTADO ESPERADO** |
| meuProduto.getPreçoVenda() = -3 (1) | Produto não encontrado |
| idProduto ∉ CONJUNTO DOS PRODUTOS (4) |  |

* + **ANÁLISE DE VALOR LIMITE**
  + Observação: Será realizado a análise com base no preço de venda de algum produto.
  + Variável: meuProduto.getPreçoVenda(), adotarei neste teste o nome precoVenda.

Classe de equivalência: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_0\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + Definição dos casos de teste:

|  |  |  |
| --- | --- | --- |
| **CASO DE TESTE** | **precoVenda** | **Resultado esperado** |
| 1 | -1 | “Valor Inválido” |
| 2 | 0 | “Valor Inválido” |
| 3 | 1 | Produto inserido na cesta |

## Thiago Vidal de Moraes

* + **PARTICIONAMENTO DE EQUIVALÊNCIA**
  + Funcionalidade: Finalizar Pagamento
  + Variáveis de Entrada: Desconto e radio button da escolha do tipo de pagamento
  + Variáveis internas: não possui.
  + Variáveis de saída: Mensagens “Selecione um tipo de pagamento”(1), “Pagamento efetuado com sucesso”(2), “Tente Novamente”(3) e “Valor de desconto inválido”(4).
  + Classes de equivalência:
    - **Variável Desconto**
    - Classe 1: txtDesconto.getText() < 0 (Classe inválida)
    - Classe 2: txtDesconto.getText() >= 0 e < txtTotal.getText() (Classe válida)
    - Classe 3: txtDesconto.getText() > txtTotal.getText() (Classe inválida)
    - **Variável Radio Button**
    - Classe 3: nenhuma radio button selecionada (Classe inválida)
    - Classe 4: qualquer radio button selecionada (Classe válida)
  + Definição dos casos de teste:
    - Caso de teste 1:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA = 1)** | **RESULTADO ESPERADO** |
| txtDesconto.getText() = -1.2 | Mensagem 4 |

* + - Caso de teste 2:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA = 2)** | **RESULTADO ESPERADO** |
| txtDesconto.getText() = 0 | Valor válido |

* + - Caso de teste 3:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA 3)** | **RESULTADO ESPERADO** |
| txtDesconto.getText() > txtTotal.getText() | Mensagem 4 |

* + - Caso de teste 4:

|  |  |
| --- | --- |
| **VARIÁVEL (CLASSE COBERTA)** | **RESULTADO ESPERADO** |
| Nenhum radio button selecionado | Mensagem 1 |

* + **ANÁLISE DE VALOR LIMITE**
  + Observação: Será realizado a análise no valor de desconto.
  + Variável: txtDesconto.getText().

Classe de equivalência: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_0\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + Definição dos casos de teste:

|  |  |  |
| --- | --- | --- |
| **CASO DE TESTE** | **txtDesconto.getText()** | **Resultado esperado** |
| 1 | -1 | Mensagem 4 |
| 2 | 0 | Mensagem 1 |
| 3 | 1 | Mensagem 1 |
| 4 | = txtTotal.getText() – 1 | Mensagem 1 |
| 5 | = txtTotal.getText() | Mensagem 1 |
| 6 | = txtTotal.getText() + 1 | Mensagem 4 |

## João Pedro Balestra Zacarkin

* + **PARTICIONAMENTO DE EQUIVALÊNCIA**
  + **Funcionalidade**: Adicionar composto para manipulação
  + **Variável de entrada**: quantidadeRequerida , unidadeSelecionada (combobox onde o usuário seleciona a unidade de medida para a quantidade requerida)
  + **Variáveis internas**: quantidadeEstoque,unidadeEstoque (unidade de medida que foi cadastrada para a quantidade do produto no estoque).
  + **Variáveis de saída**:

Mensagens :“Quantidade Inválida”, “Quantidade insuficiente em estoque”, “Unidade de medida inválida” e o composto é adicionado à lista de manipulação (mostrado graficamente na tela pela passagem de uma tabela para outra).

* **Classes de equivalência**:
  + **Variável quantidadeRequerida**:
    - Classe 1: quantidadeRequerida >= 1 (Classe Válida)
    - Classe 2: quantidadeRequerida está vazia (Classe Inválida)
    - Classe 3: quantidadeRequerida < 1 (Classe Inválida)
  + **Variável quantidade Estoque**:
    - Classe 4: quantidadeEstoque > quantidadeRequerida (Classe válida)
    - Classe 5: quantidadeEstoque < quantidadeRequerida (Classe inválida)
  + **Variável unidadeSelecionada:**
    - unidadeSelecionada ∈ {Kg, g, mg} :
      * Classe 6: unidadeEstoque ∈ {Kg, g, mg} (Classe Válida)
      * Classe 7: unidadeEstoque ∈ {l, ml} (Classe Inválida)
    - unidadeSelecionada ∈ {l, ml}
      * Classe 8: unidadeEstoque ∈ {l, ml} (Classe Válida)
      * Classe 9: unidadeEstoque ∈ {Kg, g, ml} (Classe Inválida)
  + **Variável Mensagem:**
    - Classe 10: “Quantidade Inválida” (Classe válida)
    - Classe 11: “Quantidade insuficiente em estoque” (Classe Válida)
    - Classe 12: “Unidade de medida inválida” (Classe Válida)
  + **Variável Composto adicionado à lista de manipulação:**
    - Classe 13: O composto é adicionado à tabela de manipulação (Classe Válida)
* **Definição dos casos de Teste**:
  + **Caso de teste 1:**

|  |  |
| --- | --- |
| **Classes Cobertas** | **Resultado Esperado** |
| quantidadeRequerida = 100 (1)  unidadeSelecionada = “g” e unidadeEstoque = “Kg” (6)  quantidadeEstoque = 10 (4) | Composto adicionado à lista de manipulação (13) |

* + **Caso de teste 2:**

|  |  |
| --- | --- |
| **Classes Cobertas** | **Resultado Esperado** |
| quantidadeRequerida = 100 (1)  unidadeSelecionada = “ml” e unidadeEstoque = “l” (8)  quantidadeEstoque = 10 (4) | Composto adicionado à lista de manipulação (13) |

* + **Caso de teste 3:**

|  |  |
| --- | --- |
| **Classes Cobertas** | **Resultado Esperado** |
| quantidadeRequerida = 0 (3) | “Quantidade inválida”(10) |

* + **Caso de teste 4:**

|  |  |
| --- | --- |
| **Classes Cobertas** | **Resultado Esperado** |
| quantidadeRequerida = 11 (2)  unidadeSelecionada = “Kg” e unidadeEstoque = “Kg” (6)  quantidadeEstoque = 10 (5) | “Quantidade insuficiente em estoque”(11) |

* + **Caso de teste 5:**

|  |  |
| --- | --- |
| **Classes Cobertas** | **Resultado Esperado** |
| unidadeSelecionada = “ml” e unidadeEstoque = “Kg” (9) | “Unidade de medida inválida”(12) |

Agora que todas as classes válidas foram cobertas, inicia-se os casos de teste para as classes inválidas ainda não cobertas.

* + **Caso de teste 6:**

|  |  |
| --- | --- |
| **Classes Cobertas** | **Resultado Esperado** |
| unidadeSelecionada = “g” e unidadeEstoque = “ml” (9) | “Unidade de medida inválida”(12) |

* **ANÁLISE DE VALOR LIMITE**
  + **Variável quantidadeRequerida:**

- A variável está no seguinte intervalo:

1 <= quantidadeRequerida <= quantidadeEstoque

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Caso de teste** | **quantidadeRequerida** | **quantidadeEstoque** | **unidadeEstoque** | **unidadeSelecionada** | **Saída Esperada** |
| 1 | 0 | 1000 | g | g | Quantidade inválida. |
| 2 | 1 | 1000 | g | g | Produto Adicionado |
| 3 | 2 | 1000 | g | g | Produto adicionado |

* + **Variável unidadeSelecionada:**

A unidade de medida deve pertencer ao mesmo tipo (Sólido ou Líquido). Assim, o usuário deve selecionar uma unidade compatível a unidade cadastrada no banco de dados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Caso de teste** | **quantidadeRequerida** | **quantidadeEstoque** | **unidadeEstoque** | **unidadeSelecionada** | **Saída Esperada** |
| 4 | 10 | 1000 | g | mg | Produto adicionado. |
| 5 | 10 | 1000 | g | ml | Unidade Inválida |
| 6 | 10 | 1000 | l | ml | Produto adicionado |
| 7 | 10 | 1000 | L | G | Unidade Inválida. |

* + **Variável quantidadeEstoque:**

A variável quantidadeEstoque deve ser maior ou igual a quantidadeRequerida.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Caso de teste** | **quantidadeRequerida** | **quantidadeEstoque** | **unidadeEstoque** | **unidadeSelecionada** | **Saída Esperada** |
| 8 | 1000 | 999 | g | g | Quantidade insuficiente em estoque |
| 9 | 1000 | 1000 | g | g | Produto adicionado. |
| 10 | 1000 | 1001 | g | g | Produto adicionado. |

## Thiago de Oliveira Lavezzo

* + **PARTICIONAMENTO DE EQUIVALÊNCIA**
  + **Funcionalidade**: realizar a criação do relatório de produtos.
  + **Variável de entrada**: dataInicial, dataFinal, itemSelecionado.
  + **Variáveis internas**: itemDataCompra, itemDataVencimento.
  + **Variáveis de saída**: msg1(Não existe entrada de produtos nessa data), msg2(Existem campos obrigatorios), msg3(Não existe produtos que vencem nessa data),msg4(Documento gerado com sucesso).
  + **Classe de Equivalencia:**
* **Variavel dataInicial:**
  + - Classe1: sempre valida {valido}
* **Variavel dataFinal:**
  + - Classe3: sempre valida {valida}
* **Variavel itemSelecionado:**
  + - Classe4: um dos itens selecionados (itemDataCompra, ItemDataVencimento) {valida}
    - Classe5: nenhum dos itens selecionados {invalida}
* **Variavel msg1:**
  + - Classe6: Não existe entrada de produtos nessa data {valida}
* **Variavel msg2:**
  + - Classe7: Existem campos obrigatorios {valida}
  + **Variavel msg3:**
    - Classe8: Não existe produtos que vencem nessa data {valida}
  + **Variavel msg4:**
    - Classe9: Documento gerado com sucesso {valida}
  + **Definição de Caso de teste:**
* **Caso de teste 1**

dataInicial = 31/01/2016; (cobre classe 1)

dataFinal = 31/01/2016; (cobre classe 2)

itemSelecionado = nenhum item selecionado(cobre a classe 5)

resultado esperado = msg2 (cobre calsse 7)

* **Caso de teste 2:**

dataInicial = 31/01/2016; (cobre classe 1)

dataFinal = 31/01/2016; (cobre classe 2)

itemSelecionado = itemDataCompra(cobre a classe 4)

resultado esperado = msg4 (cobre calsse 9)

* **Caso de teste 3:**

dataInicial = 31/01/2016; (cobre classe 1)

dataFinal = 31/01/2016; (cobre classe 2)

itemSelecionado = itemDataCompra(cobre a classe 4)

Observação: não existe produtos cadastrados nesse intervalo de tempo

resultado esperado = msg1 (cobre classe 6)

* **Caso de teste 4:**

dataInicial = 31/01/2016; (cobre classe 1)

dataFinal = 31/01/2016; (cobre classe 2)

itemSelecionado = itemDataVencimento(cobre a classe 4)

Observação: não existe produtos que vencem nesse intervalo de tempo

resultado esperado = msg3 (cobre calsse 8)

* + **ANÁLISE DE VALOR LIMITE**

Observação: Será realizado a análise com base na Data de Compra do produto.

Variável: dataInicio, dataFim.

Classe de equivalência:

O intervalo de [a;b], representam o intervalo de tempo onde houveram cadastro de compra de produtos [01/01/2016; 10/01/2016]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_a\_\_\_\_\_\_\_\_\_b\_\_\_\_\_\_\_\_

Definição dos casos de teste:

|  |  |  |  |
| --- | --- | --- | --- |
| **CASO DE TESTE** | **dataInicio** | **dataFim** | **Resultado esperado** |
| 1 | 27/01/2015 | 28/02/2015 | Msg1 |
| 2 | 01/01/2016 | 01/01/2016 | Msg7 |
| 3 | 02/01/2016 | 05/01/2016 | Msg7 |
| 4 | 08/01/2016 | 09/01/2016 | Msg7 |
| 5 | 10/01/2016 | 10/01/2016 | Msg7 |
| 6 | 11/01/2016 | 05/02/2016 | Msg1 |

# Teste Caixa Branca:

## Matheus Alberto Vaz Cecilio

* + Funcionalidade: Clique do botão para finalizar um pedido.
  + Observações: O clique do botão é implementado automaticamente pela IDE Netbeans, não permitindo a passagem de parâmetros, sendo assim as variáveis são globais e estão na própria interface que, as enxergam para realizar algumas funcionalidades.
  + Variáveis globais do código que deverão conter algum valor:

private final List<Produto> listaCesta = new ArrayList<>();

private final int idFuncionario;

private javax.swing.JLabel txtIdCliente;

* + Código:

private void btnFinalizarPedidoActionPerformed(java.awt.event.ActionEvent evt) {

1 if (txtIdCliente.getText().equals("")) { //VAZIO, NÃO SELECIONOU CLIENTE

2 pedidoController.habilitarNaoCliente(labelCod, labelCliente, txtNaoCliente); //HABILITA UMA LABEL QUE DIZ QUE SE DEVE SELECIONAR UM CLIENTE

3 JOptionPane.showMessageDialog(this, "Selecione um cliente para continuar!", "Algo deu errado.", JOptionPane.ERROR\_MESSAGE);

4 } else if (listaCesta.isEmpty()) {

5 JOptionPane.showMessageDialog(this, "Não é possível continuar, sua cesta está vazia, insira algum item!", "Algo deu errado.", JOptionPane.ERROR\_MESSAGE);

6 } else {

7 pedidoController.desabilitarNaoCliente(labelCod, labelCliente, txtNaoCliente);

8 Cliente cliente = new ClienteDao().localizar(Integer.parseInt(txtIdCliente.getText()));

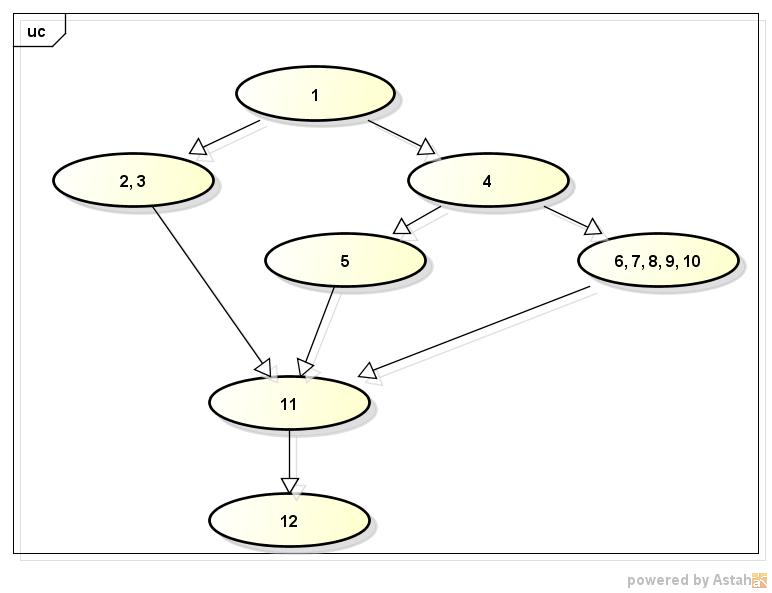
9 Funcionario funcionario = new FuncionarioDao().localizar(idFuncionario);

10 pedidoController.finalizarPedido(this, cliente, funcionario, listaCesta, total);

11 } //FIM SE

12 } //FIM MÉTODO

* + Grafo:



* + Complexidade Ciclomática (CC):

E -> Número de ramos do grafo (arco/aresta);

N -> Número de nós (vértice).

CC = E – N + 2 -> 8 – 7 + 2 = **3** (Três testes)

* + Caminhos independentes:
    - 1-2-3-11-12:
      * Casos de teste:
        + txtIdCliente = “”, listaCesta = [Produto2, Produto5, Produto20] .
      * Resultados:
        + Mensagem "Selecione um cliente para continuar!" na tela, o método é encerrado, continua na tela de pedido.
    - 1-4-5-11-12:
      * Casos de teste:
        + txtIdCliente = “122”, listaCesta = [].
      * Resultados:
        + Mensagem "Não é possível continuar, sua cesta está vazia, insira algum item!" na tela, o método é encerrado, continua na tela de pedido.
    - 1-4-6-7-8-9-10-11-12:
      * Casos de teste:
        + txtIdCliente = “122”, listaCesta = [Produto2, Produto5, Produto20].
      * Resultados:
        + Mensagem "Pedido feito com sucesso!"" na tela, e é feito o pedido e o sistema encaminha o usuário pra outra tela.

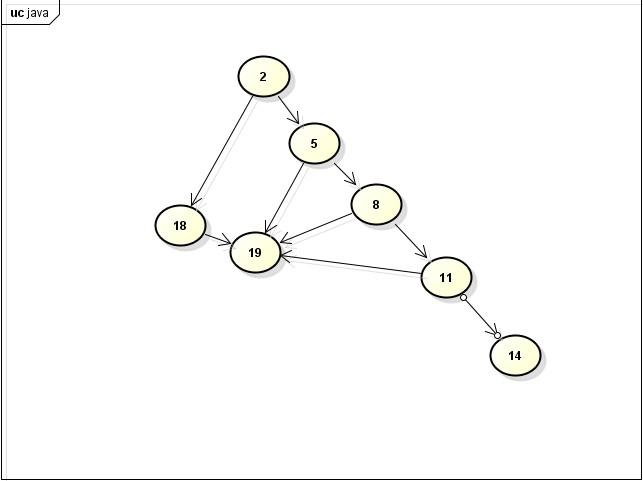
Observação: A mensagem neste caso de teste é mostrada através do controller, que antes de exibir a mensagem ele simplesmente faz uma confirmação do pedido. Por isso a mensagem não se encontra no código, mas sim no controller, através do método “finalizarPedido”.

## Thiago Vidal de Moraes:

* + Funcionalidade: botão para finalizar a compra.
  + Código:

1. String tipo = null;
2. if (Double.parseDouble(txtDesconto.getText().replace(',', '.')) < 0.00 ||
3. Double.parseDouble(txtDesconto.getText().replace(',', '.')) > Double.parseDouble(txtTotal.getText().replace(',', '.'))) {
4. JOptionPane.showMessageDialog(this, "Valor de desconto inválido", "Erro", JOptionPane.ERROR\_MESSAGE);
5. } else if (radioDinheiro.isSelected()) {
6. tipo = "0";
7. pagamentoController.pagamento(this, tipo, pedidoFinal.getValorCompra(), pedidoFinal.getCliente(), pedidoFinal, pedidoFinal.getDataPedido(), produtosFinal);
8. } else if (radioDébito.isSelected()) {
9. tipo = "1";
10. pagamentoController.pagamento(this, tipo, pedidoFinal.getValorCompra(), pedidoFinal.getCliente(), pedidoFinal, pedidoFinal.getDataPedido(), produtosFinal);
11. } else if (radioCrédito.isSelected()) {
12. tipo = "2";
13. pagamentoController.pagamento(this, tipo, pedidoFinal.getValorCompra(), pedidoFinal.getCliente(), pedidoFinal, pedidoFinal.getDataPedido(), produtosFinal);
14. } else {
15. labelPagamento.setVisible(true);
16. labelPagamentoObrigatorio.setVisible(true);
17. JOptionPane.showMessageDialog(this, "Selecione um tipo de pagamento.", "Erro", JOptionPane.ERROR\_MESSAGE);
18. }//FIM IF
19. //FIM IF

* Grafo:



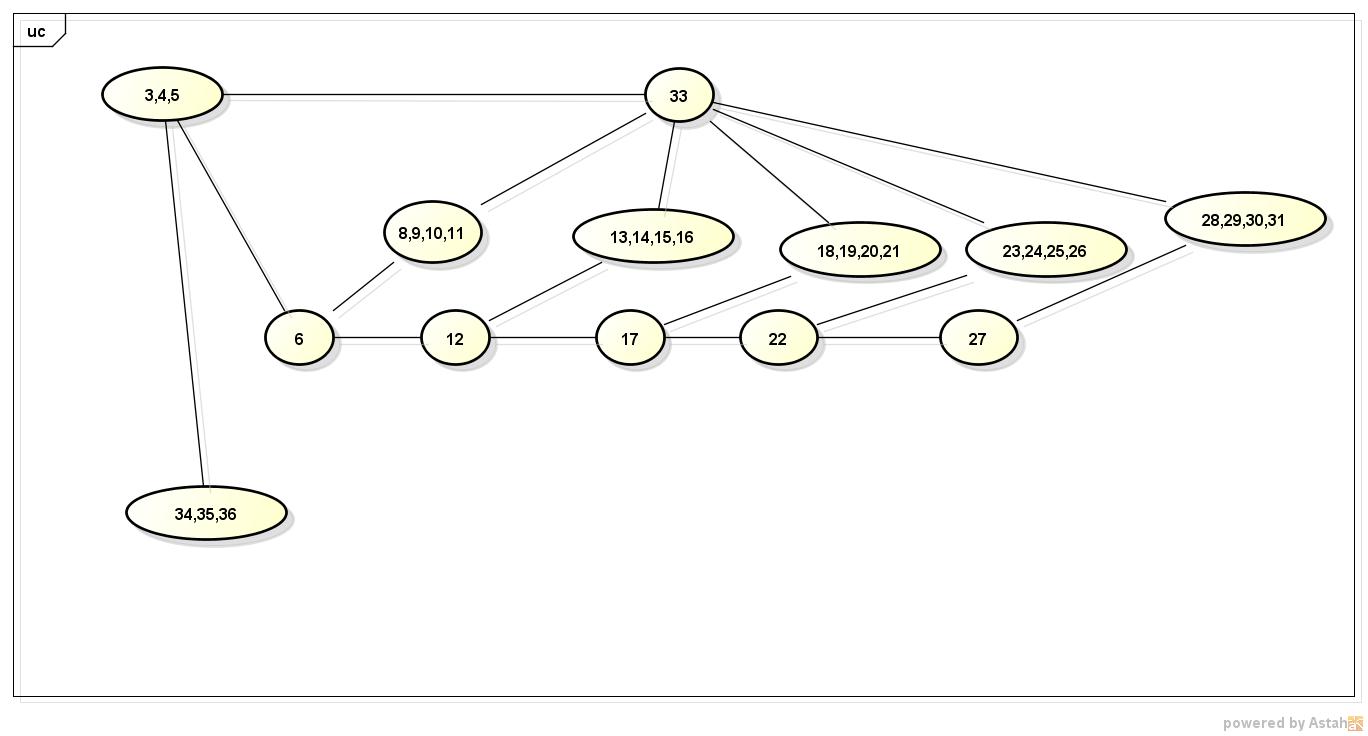
* Observação: O nó “Fim” simboliza que caiu no if e saiu da função.
* Complexidade Aciclomática:
  + 9 (arestas) – 7 (nós) + 2 = 4
* Caminhos Independentes:
  + 2 – 18 – 19
    - Caso de teste : valor do desconto menor que zero ou maior que o valor total
    - Mensagem: “Valor de desconto inválido”
  + 2 – 5 – 19
    - Caso de teste: valor de desconto válido e radioDinheiro selecionado
    - Mensagem: “Pagamento efetuado com sucesso”
  + 2 – 5 – 8 – 19
    - Caso de teste: valor de desconto válido e radioDébito selecionado
    - Mensagem: “Pagamento efetuado com sucesso”
  + 2 – 5 – 8 – 11 – 19
    - Caso de teste: valor de desconto válido e radioCrédito selecionado
    - Mensagem: “Pagamento efetuado com sucesso”
  + 2 – 5 – 8 – 11 – 14
    - Caso de teste: valor de desconto válido e nenhum radioButton selecionado
    - Mensagem: “Selecione um tipo de Pagamento”

## João Pedro Balestra Zacarkin

* + **Funcionalidade: Cancelar Manipulação-** Funcionalidade do controller para o cancelamento da manipulação de compostos.
  + **Variáveis** **principais**: composto.getUnidade() vai devolver a unidade de medida utilizada no estoque
  + **Código**:

1. **public** **void** cancelarManipulacao(List<Composto> listaCompostos) {
3. **double** aux = 0;
5. **for** (Composto composto : listaCompostos) {
7. **if** (composto.getUnidade().equals("Kg")) {
9. aux = composto.getQuantidadeEstoque() + composto.getQuantidadeSelecionada() / 1000;
10. composto.setQuantidadeEstoque(aux);
12. } **else** **if** (composto.getUnidade().equals("g")) {
14. aux = composto.getQuantidadeEstoque() + composto.getQuantidadeSelecionada();
15. composto.setQuantidadeEstoque(aux);
17. } **else** **if** (composto.getUnidade().equals("mg")) {
19. aux = composto.getQuantidadeEstoque() + (composto.getQuantidadeSelecionada() \* 1000);
20. composto.setQuantidadeEstoque(aux);
22. } **else** **if** (composto.getUnidade().equals("l")) {
24. aux = composto.getQuantidadeEstoque() + (composto.getQuantidadeSelecionada() / 1000);
25. composto.setQuantidadeEstoque(aux);
27. } **else** **if** (composto.getUnidade().equals("ml")) {
29. aux = composto.getQuantidadeEstoque() + composto.getQuantidadeSelecionada();
30. composto.setQuantidadeEstoque(aux);
32. }
33. **new** CompostoDao().atualizarQuantidadeEstoqueManip(composto.getId(), aux);
35. }
36. }

* }
* **Grafo de fluxo:**



* **Complexidade Ciclomática:**
  + **P =** Número de nós condicionais
  + **CC = P + 1 = 5 + 1 = 6**

**Ou**

* + - * **N = Arestas**
      * **E = Vértices**
      * **CC = N – E + 2 = 17 – 13 + 2 = 6**

* **Caminhos Independentes:**
  + (3,4,5) – (34,3536)

**Caso de teste**: quando o laço chega ao fim

**Resultado**: cancelamento de manipulação é realizado com sucesso.

* + (3,4,5) – 6 – (8,9,10,11) – 33 – (3,4,5) – (34,3536)
    - **Caso de teste:** Quando a unidade de medida em estoque está em quilogramas (Kg)
    - **Resultado:** O estoque é atualizado corretamente.
  + (3,4,5) – 12 – (13,14,15,16) – 33 – (3,4,5) – (34,3536)
    - **Caso de teste:** Quando a unidade de medida em estoque está em gramas (g)
    - **Resultado:** O estoque é atualizado corretamente.
  + (3,4,5) – 17 – (18,19,20,21) – 33 – (3,4,5) – (34,3536)
    - **Caso de teste:** Quando a unidade de medida em estoque está em miligramas (mg)
    - **Resultado:** O estoque é atualizado corretamente.
  + (3,4,5) – 22 – (23,24,25,26) – 33 – (3,4,5) – (34,3536)
    - **Caso de teste:** Quando a unidade de medida em estoque está em litros (l)
    - **Resultado:** O estoque é atualizado corretamente.
  + (3,4,5) – 27 – (28,29,30,31) – 33 – (3,4,5) – (34,3536)
    - **Caso de teste:** Quando a unidade de medida em estoque está em mililitros (ml)
    - **Resultado:** O estoque é atualizado corretamente.

## Thiago de Oliveira Lavezzo

Funcionalidade: método para geração de relatório de venda

Codigo:

1. **public** **void** geraRelatorioVenda(Frame tela, Date dataIni, Date dataFim) {
2. //throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
3. List<Pedido> lista = **new** PedidoDao().listarRelatorioPedido(dataIni, dataFim);

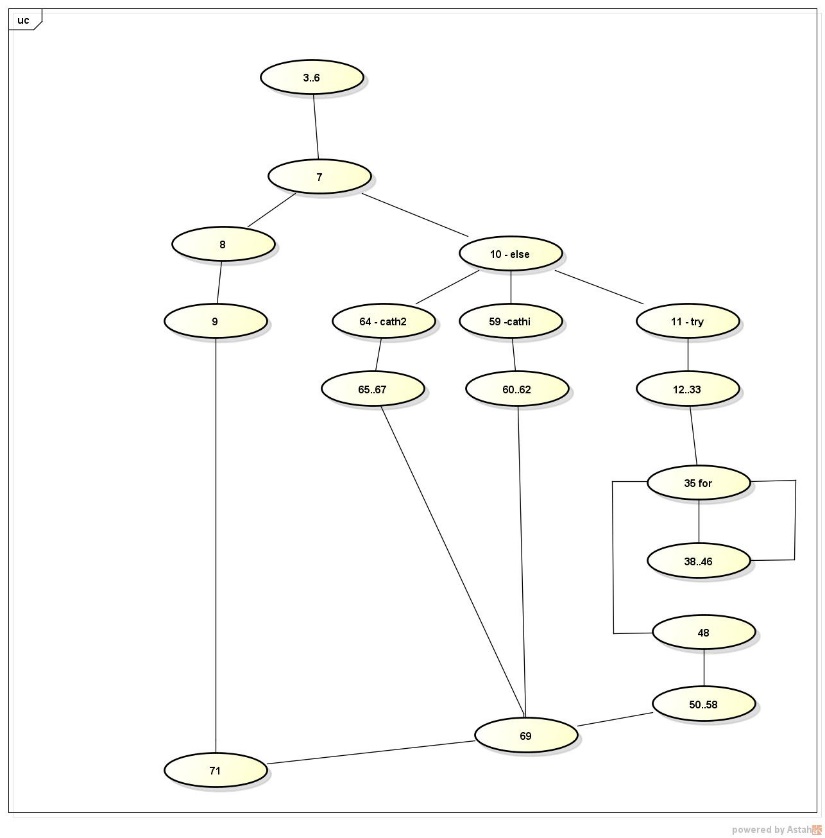
6. Document document = **new** Document();
7. **if** (lista.isEmpty()){
8. JOptionPane.showMessageDialog(tela, "Nao existe vendas nessa data", "Algo deu errado", JOptionPane.DEFAULT\_OPTION);
9. }
10. **else**{
11. **try**{
12. PdfWriter.getInstance(document, **new** FileOutputStream("C:\\Users\\Public\\relatorioVenda.pdf"));
13. document.open();
15. com.itextpdf.text.Font fonteNegrito1 = FontFactory.getFont(FontFactory.TIMES\_ROMAN, 20,Font.BOLD,BaseColor.BLACK);
17. Paragraph preface = **new** Paragraph("Relatorio de Venda de Produtos",fonteNegrito1);
18. preface.setAlignment(Element.ALIGN\_CENTER);// centraliza texto
19. document.add(preface);

22. DataSistema dataAgora =  **new** DataSistema();
23. document.add(**new** Paragraph("\nData e hora de criaÃ§Ã£o de documento: "+dataAgora.getDateTime()));
24. document.add(**new** Paragraph("Busca realizada de "+dataIni.toString()+" ate "+dataFim.toString()));

27. com.itextpdf.text.Font fonteNegrito2 = FontFactory.getFont(FontFactory.TIMES\_ROMAN, 14,Font.BOLD,BaseColor.BLACK);
28. Paragraph subtit = **new** Paragraph("\nResultados Encontrados",fonteNegrito2);
29. subtit.setAlignment(Element.ALIGN\_CENTER);// centraliza texto
30. document.add(subtit);
32. **double** total = 0;
33. **double** aux = 0;
35. **for**(Pedido prod : lista){

38. Paragraph id = **new** Paragraph("\nPedido: "+prod.getId(),fonteNegrito2);
39. id.setAlignment(Element.ALIGN\_LEFT);// centraliza texto
40. document.add(id);
41. document.add(**new** Paragraph("Data do Pedido: "+prod.getDataPedido().toString()));
42. document.add(**new** Paragraph("Funcionario responsavel por venda: "+prod.getFuncionario().getNome()));
43. document.add(**new** Paragraph("Cliente: "+prod.getCliente().getNome()));
44. document.add(**new** Paragraph("Valor da Venda: "+prod.getValorCompra()+" reais"));
46. total+=prod.getValorCompra();
48. }
50. Paragraph tot = **new** Paragraph("\nValor total das vendas: "+total+" reais",fonteNegrito2);
51. tot.setAlignment(Element.ALIGN\_RIGHT);// centraliza texto
52. document.add(tot);
54. document.close();
56. JOptionPane.showMessageDialog(tela, "Documento gerado com sucesso.", "Sucesso", JOptionPane.DEFAULT\_OPTION);
57. tela.dispose();
58. }
59. **catch**(DocumentException de){
60. System.err.println(de.getMessage());
62. }
64. **catch**(IOException ioe){
65. System.err.println(ioe.getMessage());
67. }
69. }
71. }

* Grafo de fluxo



* Complexidade ciclomatica: 5 (número regiões)
* Caminhos independentes:
* (3..6)-7-8-9-71:
  + Caso de teste: não existe vendas realizadas nas datas limites dataIni e dataFim, ex só existem vendas realizadas do dia 01/01/2016 ate 03/02/2016 e minha entrada foi de dataIni(11/02/2016) ate dataFim(15/03/2016)
  + Resultado: “Nao existe vendas nessa data”
* (3..6)-7-10-64-(65..67)-69-71:
  + Caso de teste: existem vendas realizadas no período de dataIni até dataFim, mas quando for salvar o arquivo ele se corrompeu por algum motivo.
  + Resultado:   System.err.println(ioe.getMessage());
* (3..6)-7-10-59-(60..62)-69-71:
  + Caso de teste: existem vendas realizadas no período de dataIni até dataFim, mas quando foi escrever o no arquivo, ouve algum erro.
  + Resultado:   System.err.println(de.getMessage());
* (3..6)-7-10-11-(12..33)-35-(38..46) – 35 – (38..46):
  + Caso de teste: existem vendas realizadas no período de dataIni até dataFim, onde a lista de venda é retorna mais de 1 elemento.
  + Resultado: escrita dos dados no arquivo
* (3..6)-7-10-11-(12..33)-35-(38..46) – 35-48-(50..58)- 69-71:
  + Caso de teste: existem vendas realizadas no período de dataIni até dataFim, onde a lista de venda é retorna mais de 1 elemento.
  + Resultado: “Documento gerado com sucesso”

# Padrão de Projeto:

Além do Singleton e DAO, tentaremos incorporar o padrão decorator ao projeto na parte de manipulação.

# Ajuda Online:

Para a realização do pedido.

# Funcionalidade Web:

Envio de e-mails aos clientes, alertando-os acerca de promoções.