**Universidade Estadual de Maringá**

**Trabalho 2 Bimestre – SIGMA**

Disciplina: Implementação de Sistemas de Software

Professor: Donizete Carlos Bruzarosco

Equipe:

Raul Ramires r.a.: 82293

Rafael Montrezol r.a.: 94980

Renato Alberto r.a.: 96565

Maringá, 27 de novembro de 2018

Sumário

[1 Product Backlog 3](#_Toc531022566)

[2 Planejamento das Sprints 4](#_Toc531022567)

[2.1 Sprint 1 (4 semanas) 4](#_Toc531022568)

[2.2 Sprint 2 (4 semanas) 4](#_Toc531022569)

[3 Diagrama de Classes MVC 5](#_Toc531022570)

[4 Planejamento dos Testes 6](#_Toc531022571)

[4.1 Recursos 6](#_Toc531022572)

[4.2 Testes Unitários 6](#_Toc531022573)

[4.2.1 TestRead – TerrenoDAO – Raul 6](#_Toc531022574)

[4.2.2 TestReadDefensivos – ProdutoDAO – Raul 6](#_Toc531022575)

[4.2.3 TestReadFiltro – PlantioDAO – Renato 6](#_Toc531022576)

[4.2.4 TestRead – ColheitaDAO – Renato 6](#_Toc531022577)

[4.2.5 testReadTerreno – MovimentoDAO – Rafael 6](#_Toc531022578)

[4.2.6 testRead – MovimentoDAO – Rafael 6](#_Toc531022579)

[4.3 Testes de Caixa Preta 6](#_Toc531022580)

[4.3.1 Realizar Aplicação de Defensivo em um Terreno – Raul 6](#_Toc531022581)

[4.3.2 Realizar um plantio em um terreno – Renato 8](#_Toc531022582)

[4.3.3 Compra de produtos para o estoque – Rafael 9](#_Toc531022583)

[4.4 Testes de Caixa Branca 14](#_Toc531022584)

[4.4.1 Realizar Aplicação de Defensivo em um Terreno – Raul 14](#_Toc531022585)

[4.4.2 Realizar um plantio em um terreno – Renato 15](#_Toc531022586)

[4.4.3 Confirmar a compra de produtos para o estoque – Rafael 17](#_Toc531022587)

[5 Padrão de Projeto 20](#_Toc531022588)

[6 Framework 20](#_Toc531022589)

# Product Backlog

|  |  |  |
| --- | --- | --- |
| **Descrição** | **Responsável** | **Observação** |
| **Prioridade Alta** | | |
| Cadastro de Máquinas | Raul | Já Implementado |
| Cadastro de Itens | Rafael | Já Implementado |
| Inserir Informações | Renato | Já Implementado |
| Inserir Horas | Rafael | Já Implementado |
| Informar Horas | Raul | Já Implementado |
| Emitir Aviso | Renato | Já Implementado |
| Informar Manutenção | Rafael | Já Implementado |
| Exportar Relatório de Manutenção | Renato | Já Implementado |
| Excluir Item | Raul | Já Implementado |
| Excluir Máquina | Rafael | Já Implementado |
| Cadastrar Usuário | Renato | Já Implementado |
| Fazer Login | Raul | Já Implementado |
| Fazer Logoff | Raul | Já Implementado |
| Cadastro de Estoque | Renato |  |
| Cadastro de Terrenos | Raul |  |
| Informar Colheita | Raul |  |
| Realizar venda de grãos | Rafael |  |
| Informar gastos com o terreno | Renato |  |
| Realizar plantio em um terreno | Renato |  |
| Realizar compra de produtos para o estoque | Rafael |  |
| Informar ao usuário a previsão do tempo para o dia atual | Raul |  |
| Realizar aplicação de defensivo em um terreno | Raul |  |
| Relatório Econômico | Rafael |  |
| Venda de grãos | Rafael |  |
| **Prioridade Média** | | |
| Alterar Máquina | Raul | Já Implementado |
| Alterar Item | Rafael | Já Implementado |
| Informar Variação de Preço | Raul | Já Implementado |
| Informar Variação de Tempo de Duração | Renato | Já Implementado |
| Informar Relação Preço/Tempo | Rafael | Já Implementado |
| Filtrar Colheitas por data | Raul |  |
| Filtrar Plantios por data | Raul |  |
| Filtrar Movimentações Financeiras por data | Raul |  |
| Filtrar Manutenções de Terreno por data | Raul |  |
| Filtrar produtos no estoque por Tipo | Raul |  |
| Filtrar terrenos por Estado | Raul |  |
| Ajuda na tela principal | Raul |  |
| Ajuda na tela compra de produtos | Rafael |  |
| Ajuda na tela venda de grãos | Rafael |  |
| Salvar os dados excluídos em um documento de texto | Renato |  |
| Ajuda na tela de Plantios | Renato |  |
| Ajuda na tela de dados excluídos | Renato |  |
| Recuperar dados excluídos | Renato |  |
| Mostrar dados excluídos | Renato |  |

# Planejamento das Sprints

## Sprint 1 (4 semanas)

|  |  |  |
| --- | --- | --- |
| **Descrição** | **Responsável** | **Tempo Estimado (horas)** |
| Inserção de Estoque | Renato | 4 |
| Alteração de Estoque | Raul | 4 |
| Remoção de Estoque | Rafael | 4 |
| Inserção de Terreno | Raul | 4 |
| Alteração de Terreno | Rafael | 4 |
| Remoção de Terreno | Renato | 4 |
| Informar Colheita | Raul | 4 |
| Filtrar Colheitas por data | Raul | 4 |
| Filtrar Plantios por data | Raul | 4 |
| Filtrar Movimentações Financeiras por data | Raul | 4 |
| Tela para mostrar os plantios | Renato | 4 |
| Tela para adicionar um plantio | Renato | 4 |

## Sprint 2 (4 semanas)

|  |  |  |
| --- | --- | --- |
| **Descrição** | **Responsável** | **Tempo Estimado (horas)** |
| Realizar Venda de grãos | Rafael | 5 |
| Informar gastos com o terreno | Renato | 8 |
| Realizar plantio em um terreno | Renato | 8 |
| Compra de produtos para o estoque | Rafael | 8 |
| Implementar uma opção para remover o filtro aplicado na tela de Colheitas. | Raul | 2 |
| Implementar uma opção para remover o filtro aplicado na tela de Plantios. | Raul | 2 |
| Implementar uma opção para remover o filtro aplicado na tela de Movimentações Financeiras. | Raul | 2 |
| Implementar uma opção para remover o filtro aplicado na tela de Terrenos. | Raul | 2 |
| Implementar um Filtro por tipo de produto no Estoque, incluindo a opção de remover o filtro. | Raul | 2 |
| Implementar um Filtro por tipo de transação na Movimentação Financeira, incluindo a opção de remover o filtro. | Raul | 2 |
| Implementar um método que mostre ao usuário a previsão do tempo no dia atual. | Raul | 8 |
| Implementar testes unitários para os métodos de inserção de Terreno e Colheita utilizando a biblioteca JUnit. | Raul | 4 |
| Implementar um método que realiza a aplicação de um defensivo escolhido do estoque em um Terreno selecionado pelo usuário. | Raul | 10 |
| Implementar Tela para mostrar as manutenções dos terrenos | Raul | 4 |
| Implementar Relatório Econômico | Rafael | 10 |
| Implementar ajuda para tela Compra de produtos | Rafael | 3 |
| Implementar ajuda para tela Venda de Grãos | Rafael | 3 |
| Implementar Venda de Grãos | Rafael | 5 |
| Implementar testes utilizando o JUnit. | Rafael | 3 |
| Implementar uma opção para salvar os dados excluídos em um documento. | Renato | 6 |
| Implementar uma opção para recuperar os dados excluídos de um documento. | Renato | 6 |
| Implementar uma opção para mostrar os dados excluídos de um documento. | Renato | 6 |
| Implementar um sistema de ajuda na tela plantio e na tela de dados excluídos para o usuário. | Renato | 8 |
| Implementar teste unitários. | Renato | 3 |
| Implementar método para ler sementes do usuário no estoque. | Renato | 5 |

# Diagrama de Classes MVC

O diagrama abaixo mostra as classes de modelo *bean,* as interfaces e as classes de modelo *DAO (Data Access Object)*. As classes de interface são por onde o usuário irá se comunicar com o sistema. As classes *DAO* são responsáveis por manipular o banco de dados, ou seja, todos métodos que manipulam o banco de dados estão nas classes *DAO* do seu respectivo *bean*.

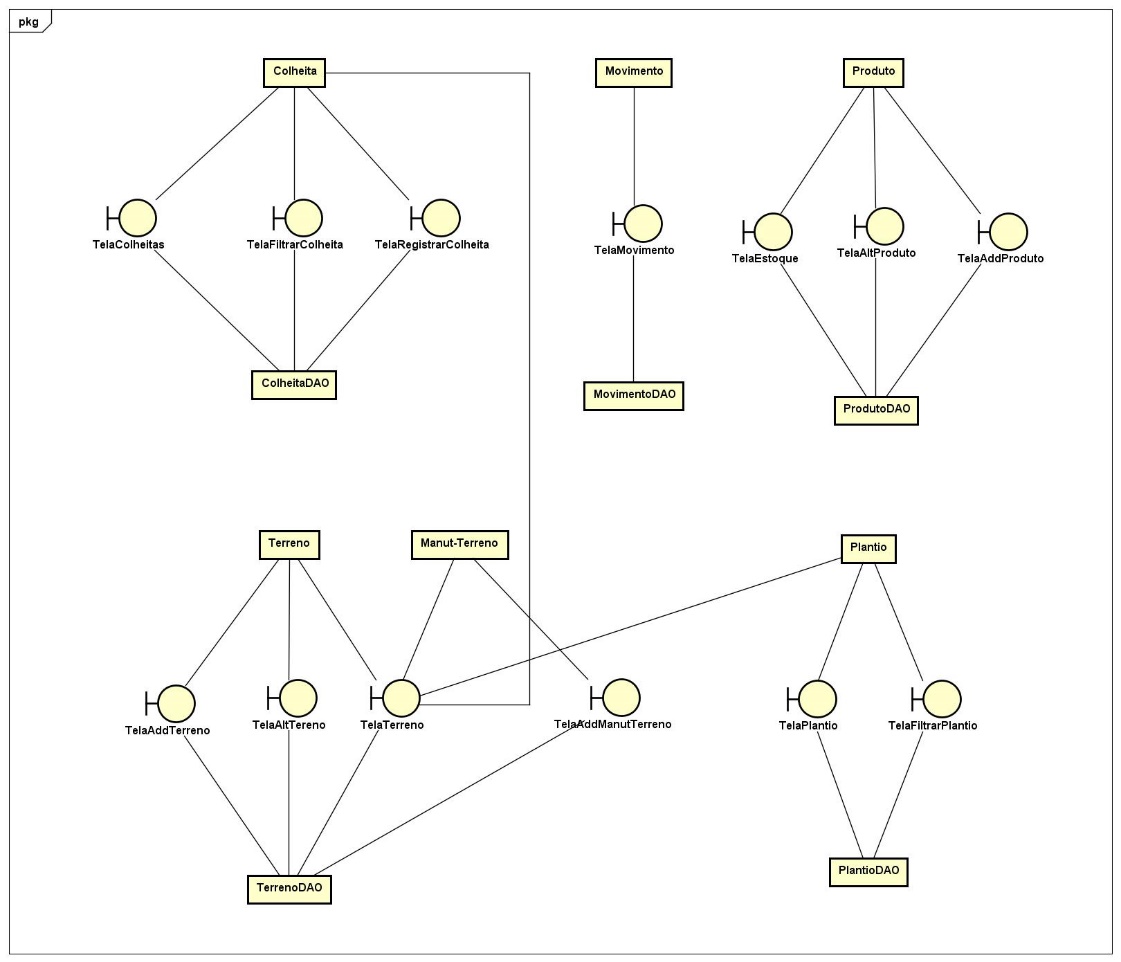


Figura 1: Diagrama de Classes MVC

# Planejamento dos Testes

## Recursos

Para a implementação de testes, será utilizada a biblioteca JUnit, que é responsável pela realização de testes unitários.

## Testes Unitários

Nessa seção serão apresentados os testes unitários implementados com a biblioteca JUnit.

### TestRead – TerrenoDAO – Raul

O teste “TestRead” é responsável por testar o método de leitura dos terrenos de um usuário. Foi implementado de modo a verificar se a lista de terrenos retornada pelo método não é vazia.

### TestReadDefensivos – ProdutoDAO – Raul

O teste “TestReadDefensivos” é responsável por retornar todos os produtos no estoque do usuário que são do tipo “Defensivo”. Foi implementado de modo a percorrer a lista de produtos resultante verificando, um a um, se o tipo do produto é “Defensivo”.

### TestReadFiltro – PlantioDAO – Renato

O teste “testReadFiltro” é responsável por testar se a tabela contida na tela plantios vai retornar o período de data especificada pelo usuário "testes". O parâmetro do teste foi dado como entre "01-12-2020" à "02-12-2020", como não há uma data com esses plantios temos que a lista de plantios retornada pelo método é vazia.

### TestRead – ColheitaDAO – Renato

O teste “TestRead” é responsável por retornar uma lista contendo todas as colheitas feitas pelo usuário "testes". Neste teste verificamos se a tabela colheita do usuário "testes" esta vazia, como não há nenhuma colheita feita ainda pelo usuário "testes" então temos que a lista de colheitas retornada pelo método é vazia.

### testReadTerreno – MovimentoDAO – Rafael

O teste “testReadTerreno” da classe MovimentoDAO é responsável por retornar todos os movimentos que foram feitos no terreno que tem o ID passado como parametro. Foi implementado de modo a percorrer a lista dos movimentos de modo que verifique em cada um deles se o ID do terreno é igual ao que foi passado.

### testRead – MovimentoDAO – Rafael

O teste “testRead” da classe MovimentoDAO é responsável por testar o método de leitura de todos os movimentos que aconteceram. Foi implementado de modo a verificar se a lista de movimentos retornada pelo método não é vazia.

## Testes de Caixa Preta

Nessa seção serão detalhados os testes de caixa preta que foram realizados.

### Realizar Aplicação de Defensivo em um Terreno – Raul

Considere *max* como sendo a quantidade do Defensivo selecionado disponível em estoque. Temos então que para esse caso, x deve pertencer ao intervalo [1, max].

Intervalo: Uma classe válida e duas inválidas

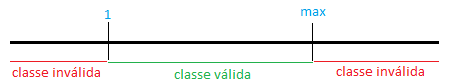


Figura 2: Classes de Equivalência

Conceitos utilizados:

* Conceito V: 1 ≤ qtde ≤ max;
* Conceito F: qtde ≤ 0 ou qtde > max.

Seja a variável *qtde* o valor de entrada para a quantidade de Defensivo que será aplicada, então temos as seguintes classes de equivalência para a variável de entrada:

Variável: qtde

* Classe 1: qtde ≤ 0 (Classe inválida);
* Classe 2: qtde > max (Classe inválida);
* Classe 3: 1 ≤ qtde ≤ max (Classe válida).

Variável de saída:

* Conceito: Conceito X (X=V ou X=F).

Definição dos casos de teste (considere max = 50):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Teste | Variável | Descrição | Resultado Esperado | Resultado Obtido |
| 1 | Qtde = -1 | Cobre a classe 1 | Conceito F | Conceito F |
| 2 | Qtde = 20 | Cobre a classe 3 | Conceito V | Conceito V |
| 3 | Qtde = 55 | Cobre a classe 2 | Conceito F | Conceito F |

Análise do valor limite (considere max = 50):

Variável: qtde

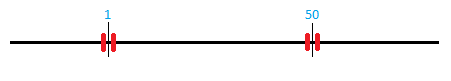


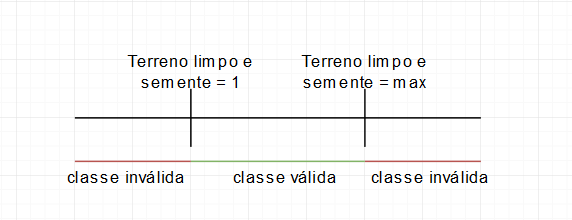
Figura 3: Valores limites para qtde

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de Teste | Variável | Resultado Esperado | Resultado Obtido |
| 1 | Qtde = 0 | Conceito F | Conceito F |
| 2 | Qtde = 1 | Conceito V | Conceito V |
| 3 | Qtde = 2 | Conceito V | Conceito V |
| 4 | Qtde = 49 | Conceito V | Conceito V |
| 5 | Qtde = 50 | Conceito V | Conceito V |
| 6 | Qtde = 51 | Conceito F | Conceito F |

### Realizar um plantio em um terreno – Renato

Considere T=0 para um terreno limpo e T=1 para um terreno que contém um plantio e também considere *max* como sendo a quantidade de sementes disponível em estoque. Temos então que para esse caso, x deve pertencer ao intervalo [1, max].

Intervalo: Uma classe válida e duas inválidas



*Figura 4: Classes de Equivalência*

Conceitos utilizados:

* Conceito V: 1 ≤ qde ≤ max e Terreno limpo;
* Conceito F: qde ≤ 0 ou qde > max e Terreno que já contém um plantio.

Seja a variável *qde* o valor de entrada para a quantidade de sementes que será plantada em um determinado terreno, então temos as seguintes classes de equivalência para a variável de entrada:

Variável: qtde

* Classe 1: T = 1 e qde ≤ 0 (Classe inválida);
* Classe 2: T = 1 e qde > max (Classe inválida);
* Classe 3: T = 0 e 1 ≤ qde ≤ max (Classe válida).

Variável de saída:

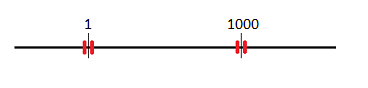
* Conceito: Conceito X (X=V ou X=F).

Definição dos casos de teste (considere max = 1000):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Teste | Variável | Descrição | Resultado Esperado | Resultado Obtido |
| 1 | T=1 e  qde = -1 | Cobre a classe 1 | Conceito F | Conceito F |
| 2 | T=0 e qde=200 | Cobre a classe 3 | Conceito V | Conceito V |
| 3 | T=1 e  qde=1500 | Cobre a classe 2 | Conceito F | Conceito F |

Análise do valor limite (considere max = 1000):

Variável: qde



*Figura 5: Valores limites para qtde*

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de Teste | Variável | Resultado Esperado | Resultado Obtido |
| 1 | T = 1 e qde = 0 | Conceito F | Conceito F |
| 2 | T = 1 e qde = 5 | Conceito F | Conceito F |
| 3 | T = 1 e  qde = 1001 | Conceito F | Conceito F |
| 4 | T = 0 e  qde = -1 | Conceito F | Conceito F |
| 5 | T = 0 e qde = 0 | Conceito F | Conceito F |
| 6 | T = 0 e  qde = 100 | Conceito V | Conceito V |
| 7 | T = 0 e  qde = 2000 | Conceito F | Conceito F |

### Compra de produtos para o estoque – Rafael

Variáveis internas:

* p1: objeto do tipo Produto;
* mov1: objeto do tipo Movimento;
* pDAO: objeto do tipo ProdutoDAO;
* movDAO: objeto do tipo MovimentoDAO;
* aux: objeto do tipo Produto para auxiliar nos cálculos;
* formato: controle do formato da data;
* dia: variável que recebe boxDia transformando em String;
* mes: variável que recebe boxMes transformando em String;
* ano: variável que recebe boxAno transformando em String;
* dataString: variável que recebe a junção das strings: dia+"/"+mes+"/"+ano;
* dataJava: variável que recebe dataString transformando para o tipo Date do Java;
* dataSql: variável que recebe dataJava transformando para o tipo Date do SQL;
* idProdutoAtual: variável que recebe o ID do produto que está sendo comprado.

Variáveis de entrada e suas respectivas classes de equivalência:

* textNome (nome do produto comprado):
  + Classe 1: produto cadastrado (classe válida);

.

* boxTipo: (tipo do produto comprado):
  + Classe 2: Semente (classe válida);
  + Classe 3: Defensivo (classe válida);

.

* textPreco: (preço do produto comprado):
  + Classe 4: números reais (classe válida);

.

* textQtde: (quantidade do produto comprado):
  + Classe 5: números reais (classe válida);

.

* textNF: (número da nota fiscal do produto comprado):
  + Classe 6: 0 <= textNF.length() <= 45 (classe válida);
  + Classe 7: textNF.length() > 45 (classe inválida);
  + Classe 8: textNF.length() < 0 (classe inválida).



* boxDia: (dia que o produto foi comprado):
  + Classe 9: 1 <= boxDia <= 31 (classe válida);
  + Classe 10: boxDia < 1 (classe inválida);
  + Classe 11: boxDia > 31 (classe inválida).



* boxMes: (mês que o produto foi comprado):
  + Classe 12: 1 <= boxMes <= 12 (classe válida);
  + Classe 13: boxMes < 1 (classe inválida);
  + Classe 14: boxMes > 12 (classe inválida).



* boxAno: (ano que o produto foi comprado):
  + Classe 15: 2018 <= boxAno <= 2025 (classe válida);
  + Classe 16: boxAno < 2018 (classe inválida);
  + Classe 17: boxAno > 2025 (classe inválida).



Variáveis de saída e suas respectivas classes de equivalência:

* Mensagem de sucesso X:
  + X = 1 (“Alterado com Sucesso!”):
    - Classe 18: “Alterado com Sucesso!” (classe válida).
  + X = 2 (“Salvo com sucesso”):
    - Classe 19: “Salvo com sucesso” (classe válida).
* Mensagem de erro X:
  + X = 1 (“Erro ao Alterar!”):
    - Classe 20: “Erro ao Alterar!” (classe válida).
  + X = 2 (“Erro ao salvar”):
    - Classe 21: “Erro ao salvar” (classe válida).
  + X = 3 (“Favor cadastre o produto”):
    - Classe 22: “Favor cadastre o produto” (classe válida).

Definição dos casos de testes:

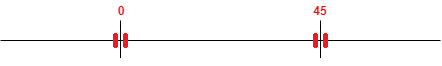
Para a análise, considere já efetuado o cadastro do seguinte produto:

* Nome: Semente.
* Tipo: Semente.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Teste | Variáveis | Resultado Esperado | Classes cobertas | Resultado Obtido |
| 1 | textNome = Semente  boxTipo = Semente  textPreco = 19.99  textQtde = 10  textNF.length() = 10  boxDia = 16  boxMes = 05  boxAno = 2018 | Mensagem de sucesso 1  Mensagem de sucesso 2 | 1, 2, 4, 5, 6, 9, 12, 15, 18, 19 | Mensagem de sucesso 1  Mensagem de sucesso 2 |
| 2 | textNome = Alimento  boxTipo = Defensivo  textPreco = 15.99  textQtde = 17  textNF.length() = 10  boxDia = 16  boxMes = 05  boxAno = 2018 | Mensagem de erro 1  Mensagem de erro 3 | 3, 20, 22 | Mensagem de erro 1  Mensagem de erro 3 |
| 3 | textNome = Semente  boxTipo = Semente  textPreco = 18.99  textQtde = 25  textNF.length() = 10  boxDia = 16  boxMes = 05  boxAno = 2018 | Mensagem de sucesso 1  Mensagem de erro 2 | 21 | Mensagem de sucesso 1  Mensagem de erro 2 |
| 4 | textNome = Alimento  boxTipo = Defensivo  textPreco = 15.99  textQtde = 17  textNF.length() = -10  boxDia = -1  boxMes = -1  boxAno = -1 | Não é possível selecionar | 8, 10, 13, 16 | Não é possível selecionar |
| 5 | textNome = Semente  boxTipo = Semente  textPreco = 18.99  textQtde = 25  textNF.length() = 50  boxDia = 40  boxMes = 20  boxAno = 2030 | Erro de execução  Não é possível selecionar | 7, 11, 14, 17 | Erro de execução  Não é possível selecionar |

Testes utilizando valor limite:

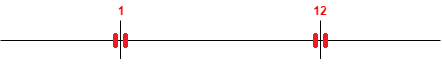
* textNF:



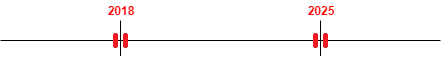
* boxDia:



* boxMes:



* boxAno:



Definição dos casos de testes:

Para a análise, considere já efetuado o cadastro do seguinte produto:

* Nome: Semente.
* Tipo: Semente.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Teste | Variáveis | Resultado Esperado | Classes cobertas | Resultado Obtido |
| 1 | textNome = Semente  boxTipo = Semente  textPreco = 15.99  textQtde = 17  textNF.length() = 1  boxDia = 2  boxMes = 2  boxAno = 2019 | Mensagem de sucesso 1  Mensagem de sucesso 2 | 6, 9, 12, 15 | Mensagem de sucesso 1  Mensagem de sucesso 2 |
| 2 | textNome = Semente  boxTipo = Semente  textPreco = 15.99  textQtde = 17  textNF.length() = 44  boxDia = 30  boxMes = 12  boxAno = 2024 | Mensagem de erro 1  Mensagem de erro 3 | 6, 9, 12, 15 | Mensagem de erro 1  Mensagem de erro 3 |
| 3 | textNome = Semente  boxTipo = Semente  textPreco = 18.99  textQtde = 25  textNF.length() = 0  boxDia = 1  boxMes = 1  boxAno = 2018 | Mensagem de sucesso 1  Mensagem de erro 2 | 6, 9, 12, 15 | Mensagem de sucesso 1  Mensagem de erro 2 |
| 4 | textNome = Semente  boxTipo = Semente  textPreco = 18.99  textQtde = 25  textNF.length() = 45  boxDia = 31  boxMes = 12  boxAno = 2025 | Mensagem de sucesso 1  Mensagem de erro 2 | 7, 11, 14, 17 | Mensagem de sucesso 1  Mensagem de erro 2 |
| 5 | textNome = Semente  boxTipo = Semente  textPreco = 18.99  textQtde = 25  textNF.length() = 46  boxDia = 32  boxMes = 13  boxAno = 2026 | Erro de execução  Não é possível selecionar | 8, 10, 13, 16 | Erro de execução  Não é possível selecionar |
| 6 | textNome = Semente  boxTipo = Semente  textPreco = 18.99  textQtde = 25  textNF.length() = -1  boxDia = 0  boxMes = 0  boxAno = 2017 | Erro de execução  Não é possível selecionar | 8, 10, 13, 16 | Erro de execução  Não é possível selecionar |

## Testes de Caixa Branca

Nessa seção serão detalhados os testes de caixa branca que foram realizados.

### Realizar Aplicação de Defensivo em um Terreno – Raul

Esse método tem como objetivo cadastrar uma aplicação de defensivos em um terreno escolhido pelo usuário. O defensivo utilizado será selecionado do estoque e a quantidade informada pelo usuário não pode ser negativa ou exceder a quantidade disponível em estoque.

A Figura abaixo mostra o método que é invocado ao pressionar o botão OK na tela responsável por executar esse método.

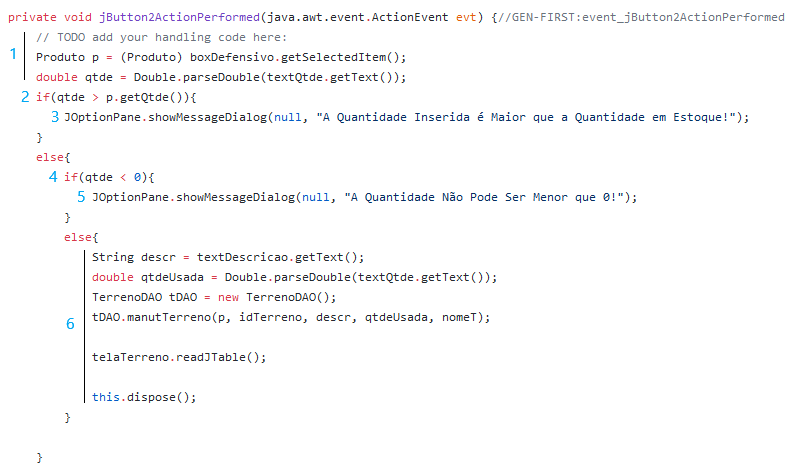


Figura 4: Código para cadastrar uma aplicação de defensivo.

A Figura abaixo mostra o grafo de fluxo do algoritmo descrito acima.

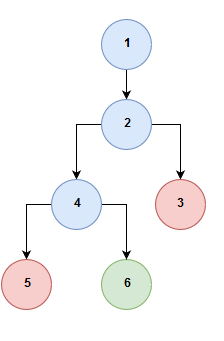


Figura 5: Grafo de fluxo do método para cadastrar uma aplicação de defensivo.

Os nós em azul são partes intermediárias do programa. Os nós em vermelho representam casos de erro, onde o programa não será executado devido a alguma entrada errada feita pelo usuário, o programa encerrará no nó 3 caso o usuário insira uma quantidade maior que a disponível em estoque. Será encerrado no nó 5 caso a quantidade seja menor que 0. Caso nenhum desses dois casos aconteçam, o método será executado no nó 6, destacado na cor verde.

O grafo acima possui complexidade ciclomática = 1. Utilizando a fórmula temos:

Onde E = 5 (número de ramos do grafo), N = 6 (número de nós do grafo), portanto:

Caminhos:

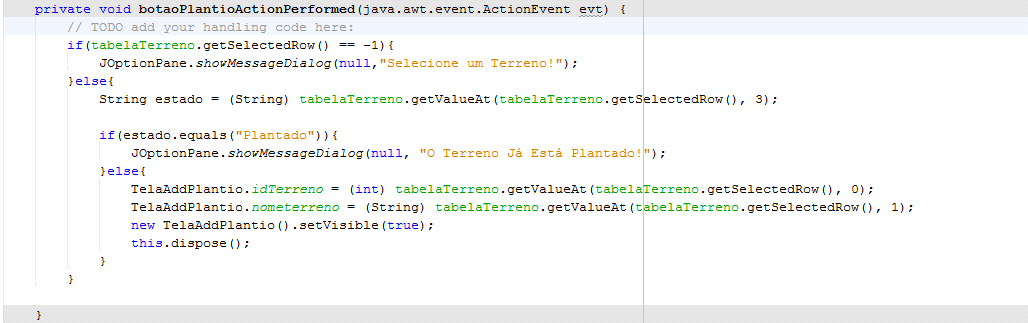
* 1-2-3;
* 1-2-4-5;
* 1-2-4-6.

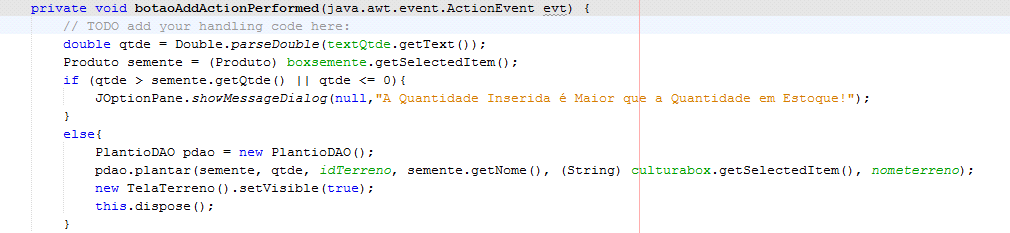
Casos de teste: CT = qtde (quantidade de defensivo inserida pelo usuário), considere max = 50.

* CT1: qtde = -1. Cobre os nós 1-2-4-5 → Quantidade negativa;
* CT2: qtde = 55. Cobre os nós 1-2-3 → Quantidade maior que a disponível;
* CT3: qtde = 30. Cobre os nós 1-2-4-6 → Quantidade aceitável.

### Realizar um plantio em um terreno – Renato

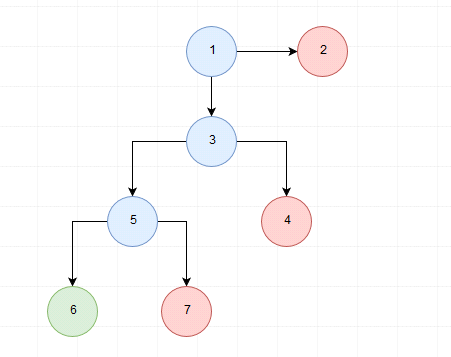
Este método tem como objetivo realizar um plantio no terreno que o usuário selecionou, teremos como agentes desta execução dois botões são eles o "Plantar" contido na tela "Terrenos" e o botão "Confirmar" da tela "TelaAddPlantio", temos que ao clicar em "Plantar", caso o terreno tenha algo plantado nele ocorrerá um erro, se o terreno estiver limpo abrirá uma nova janela para o usuário, para então ele inserir os dados, que são: semente, quantidade e cultura, caso o usuário não tenha nenhuma semente no estoque não será possível plantar e caso a quantidade de semente ultrapasse a quantidade no estoque também nao será possível plantar no terreno.





*Figura 8: Código para cadastrar realizar um plantio.*

A figura abaixo mostra o grafo de fluxo do algoritmo descrito acima.



*Figura 9: Grafo de fluxo do método para realizar um plantio.*

Os nós em azul são partes intermediárias do programa. Os nós em vermelho representam casos de erro, onde o programa não será executado devido ao terreno já existir um plantio ou alguma entrada errada feita pelo usuário, o programa encerrará no nó 2 caso o usuário tente fazer um plantio em um terreno já plantado, será encerrado no nó 4 caso o usuário insira uma quantidade maior que a disponível em estoque. Será encerrado no nó 7 caso a quantidade seja menor que 0. Caso nenhum desses dois casos aconteçam, o método será executado no nó 6, destacado na cor verde.

Complexidade ciclomática é dada pela fórmula: V(G)=E-N+2.

Temos então que o grafo acima tem complexidade ciclomática = 6-7+2 = 1

Caminhos:

* 1-2;
* 1-3-4;
* 1-3-5-7;
* 1-3-5-6.

Casos de teste: Considere T = 1 para terreno plantado e T = 0 para terreno limpo, qde para a quantidade de sementes que o usuário plantará no terreno, considere o maxímo de sementes = 1000.

* T = 1: Terreno já contém plantio. Cobre os nós 1-2→Já existe um plantio neste terreno;
* T = 0 e qde: quantidade = -1. Cobre os nós 1-3-5-7 → Quantidade negativa;
* T = 0 e qde: quantidade = 3000. Cobre os nós 1-3-4 → Quantidade maior que a disponível;
* T = 0 e qde: quantidade = 500. Cobre os nós 1-3-5-6 → Quantidade aceitável.

Onde E = 5 (número de ramos do grafo), N = 6 (número de nós do grafo), portanto:

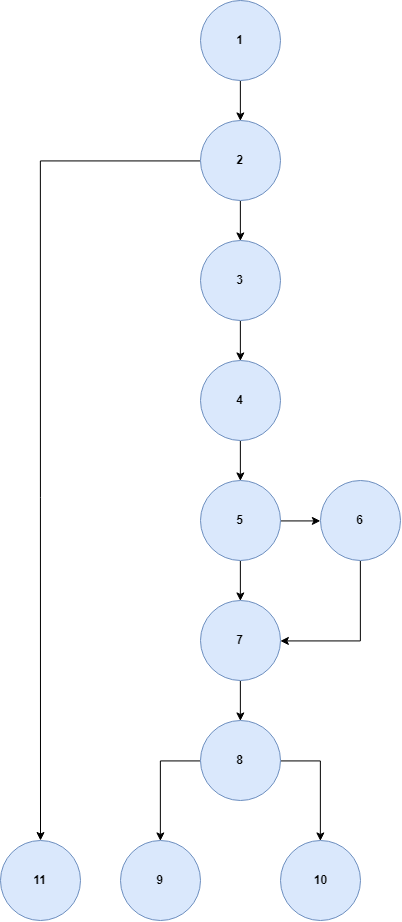
### Confirmar a compra de produtos para o estoque – Rafael

Esse método tem como objetivo confirmar uma compra de algum produto para o estoque. Esse produto já deve estar cadastrado no estoque para que a compra possa ser efetivamente cadastrada.

A figura abaixo apresenta o código do método invocado ao pressionar o botão para confirmar os dados preenchidos.



A Figura abaixo mostra o grafo de fluxo do algoritmo descrito acima.



O grafo acima possui 2 regiões, ou seja, possui complexidade ciclomática 2.

Os caminhos independentes são:

* 1, 2, 11;
* 1, 2, 3, 4, 5, 7, 8, 9;
* 1, 2, 3, 4, 5, 6, 7, 8, 9;
* 1, 2, 3, 4, 5, 7, 8, 10;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Teste | Caminho | Valor das variáveis | Resultado Esperado | Resultado Obtido |
| 1 | 1, 2, 11 | confirmacao = 0 | Voltar à tela para corrigir os dados. | Voltou à tela para corrigir os dados. |
| 2 | 1, 2, 3, 4, 5, 7, 8, 9 | confirmacao = 1  dataJava = 2018-11-25  idProdutoAtual = 12 | Dados gravados com sucesso e voltar à tela de estoque. | Dados gravados com sucesso e voltou à tela de estoque. |
| 3 | 1, 2, 3, 4, 5, 6, 7, 8, 9 | confirmacao = 1  dataJava = “conversão incorreta”  idProdutoAtual = 12 | Gravar no estoque, mas não na movimentação. | Gravou no estoque, mas não na movimentação. |
| 4 | 1, 2, 3, 4, 5, 7, 8, 10 | confirmacao = 1  dataJava = 2018-11-25  idProdutoAtual = -1 | Ir para a tela de cadastro de produto. | Foi para a tela de cadastro de produto. |

# Padrão de Projeto

O padrão de projeto utilizado no desenvolvimento do SIGMA foi o padrão *Factory,* que está implementado na classe *ConnectionFactory,* que é responsável por gerenciar as conexões com o banco de dados.

O padrão *Factory* tem como característica o encapsulamento da construção de objetos complicados. Para criar uma conexão com o banco de dados são necessários vários parâmetros, tais como o *Driver JDBC do MySQL, URL* do banco de dados, usuário e senha de acesso, além disso, é necessário tratar várias exceções que podem ocorrer na conexão. Todos esses dados estão presente no método *getConnection()* que está na classe *ConnectionFactory,* ou seja, quando for preciso abrir uma conexão com o banco de dados, basta chamar o método *ConnectionFactory.getConnection()* e este retornará uma conexão pronta para uso.

# Framework

O framework utilizado para o desenvolvimento do projeto foi o Java Swing, o qual é utilizado para a criação de interfaces gráficas de uma maneira fácil e intuitiva, utilizando o método de arrastar e soltar componentes na tela.