Universidade Federal de Sergipe
Centro de Ciências Exatas e Tecnologia
DEPARTAMENTO DE COMPUTAÇÃO - DCOMP
CIÊNCIA DA COMPUTAÇÃO

Documentos Desenvolvimento de Software

Prof. Dr. MICHEL DOS SANTOS SOARES

São Cristóvão, Sergipe
Maio – 2017

Componentes:

ALICE RODRIGUES SANTOS – 201110008664
ARTUR SANTOS NASCIMENTO – 201310004513
ELIAS RABELO MATOS – 201410011486
FELIPE DE ALMEIDA FLORENCIO – 201600016733
GABRIEL LEITE LIMA – 201410011510
GIVALDO MARQUES DOS SANTOS – 201420029045
HUGO VINICIUS DANTAS BARRETO – 201420006238
JOAO MANOEL PIMENTEL CORREIA – 201410011584
JUSLEY ARLEY DE OLIVEIRA TAVARES – 201410061030
MARINA VIVAS ANDRADE REIS – 201410011646
NATA DE ALEXANDRIA MENEZES – 201320007410
PATRICK JONES DE SOUZA CRUZ – 201410011673

Conteúdo

1 Levantamento de Requisitos

1.1 Propósito do Documento

Este documento contém a especificação de um sistema para controle e atendimento em uma fazenda que produz insumos de origem animal e produtos agrícolas. Nas próximas seções, serão apresentadas de forma mais detalhada as características do sistema.

1.2 Escopo do Produto

O sistema destina-se à gerência da fazenda, abrangendo os seguintes setores: controle de estoque, controle de funcionários, controle de venda, controle de clientes, controle de finanças, gestão de rebanhos e controle de produção agrícola.

1.3 Definições e Abreviações

1.3.1 Definições

Item(ns): Instância de um produto que pode ser manipulada.

Produto(s): Mercadorias comercializadas pela fazenda.

Insumo(s): Produto gerado por um animal. (Ex.: Leite, couro)

Fazenda: Estabelecimento de produção e venda de insumos e produtos agrícolas.

Funcionário: Pessoa física com vínculo empregatício com a fazenda.

Relatório: Apresentação de um conjunto de informações específicas da fazenda.

Cliente: Pessoa física ou jurídica que compra produtos na fazenda.

Rebanho: Coletivo de animais.

Saída de Itens: A retirada de um item do estoque.

1.3.2 Abreviações

RF: Requisito Funcional.

RNF: Requisito Não Funcional.

Pr.: Prioridade do requisito.

1.4 Referências

SOMMERVILLE, I. Engenharia de Software. Pearson/Prentice Hall.
PRESSMAN, R. S. Engenharia de Software. McGraw Hill.

1.5 Visão Geral do Restante do Documento

Nas próximas seções serão apresentadas as características gerais do sistema e suas funcionalidades. Segue ainda, a descrição de restrições e dependências que devem ser consideradas para o devido funcionamento. Atrelado às funcionalidades do sistema, é apresentada também uma lista de requisitos funcionais e não funcionais que servirão como guia para o desenvolvimento do sistema.

1.6 Descrição Geral

1.6.1 Perspectiva do Produto

Espera-se que o sistema seja desenvolvido, implementado e testado durante as disciplinas de Desenvolvimento de Software I, II e III da Universidade Federal de Sergipe.

1.6.2 Funções do Produto

O produto permite a gestão de uma fazenda, possibilitando o controle de estoque, controle do atendimento, controle de finanças, controle de funcionários, gestão de rebanhos e controle de produção agrícola.

1.6.3 Características do Usuário

No sistema só existe um tipo de usuário que é o Funcionário. Ele é responsável pela venda, controle de produtos, controle de outros funcionários, controle de finanças, controle de produção e controle de clientes.

> Funcionário: usuário com acesso a funcionalidade de gerência do sistema, responsável pelas vendas e por todo o controle realizado no sitema.

1.6.4 Restrições Gerais

Os funcionários que utilizarão o sistema deverão ser treinados para que se tornem aptos para o uso do sistema.

A fazenda deverá possuir computadores pessoais executando máquina java com configuração mínima de: processor de 1,5GHz, memória RAM 1GB, espaço de armazamento de 500 MB. Nesses computadores os funcionários acessarão o sistema.

A fazenda deverá possuir uma rede local WiFi (IEEE 802.11) disponível para que os computadores pessoais sejam conectados em rede a um computador central. Essa rede permitirá a existência de vários computadores para os funcionários atenderem clientes.

A fazenda deverá possuir um computador central onde ficará hospedado o banco de dados do sistema.

1.6.5 Suposições e Dependências

Faz-se necessário para o funcionamento do software a existência de computadores pessoais para que os funcionários utilizem no atendimento ao cliente.

Rede WiFi para que os computadores se comuniquem com o sistema da fazenda, e um computador que funcione como servidor do sistema.

1.7 Prioridades dos Requisitos

Para estabelecer a prioridade dos requisitos, foram adotadas as denominações "essencial", "importante" e "desejável".

- Essencial é o requisito sem o qual o sistema não entra em funcionamento.Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- Importante é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- Desejável é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

1.8 Requisitos específicos

1.8.1 Requisitos Funcionais

RF1 Cadastrar funcionário.. (Pr.: Essencial)

O sistema deve permitir a um funcionário cadastrar um funcionário.

RF2 Cadastrar cargo. (Pr.: Essencial)

O sistema deve permitir que um funcionário cadastre um novo cargo.

RF3 Excluir cargo. (Pr.: Desejável)

O sistema deve permitir que um funcionário exclua um cargo existente..

RF4 Definir contrato. (Pr.: Essencial)

O sistema deve permitir a um funcionário definir um contrato referente a um funcionário e um cargo já cadastrados.

RF5 Definir Endereço. (Pr.: Desejável)

O sistema deve permitir a um funcionário definir o endereço de um funcionário.

RF6 Alterar dados de um funcionário. (Pr.: Importante)

O sistema deve permitir um funcionário alterar os dados cadastrais de um funcionário.

RF7 Buscar funcionário. (Pr.: Importante)

O sistema deve permitir um funcionário buscar um funcionário de acordo com o CPF.

RF8 Listar Contratos de um funcionário. (Pr.: Desejável)

Ao buscar um funcionário, o sistema deve permitir listar os contratos que o funcionário possui.

RF9 Autenticar Funcionário. (Pr.: Importante)

Ao iniciar, o sistema deve fazer autenticação do usuário através do CPF.

RF10 Inativar contrato do funcionário. (Pr.: Importante)

O sistema deve permitir que um funcionário cancele o contrato de um funcionário.

RF11 Listar funcionários contratados. (Pr.: Desejável)

O sistema deve permitir a um funcionário listar todos os funcionários da fazenda com contrato ativo.

RF12 Listar funcionários a partir do cargo. (Pr.: Desejável)

O sistema deve permitir a um funcionário listar funcionários a partir do cargo.

RF13 Deslogar do sistema. (Pr.: Importante)

O funcionário pode deslogar do sistema.

RF14 Tela de funcionário. (Pr.: Essencial)

O funcionário deve ter acesso a todos os módulos através de uma tela com menu.

RF15 Cadastrar Cliente. (Pr.: Essencial)

O sistema deve permitir a um funcionário cadastrar um cliente.

RF16 Alterar dados de um Cliente. (Pr.: Importante)

O sistema deve permitir a um funcionário alterar dados cadastrais de um cliente.

RF17 Buscar Cliente. (Pr.: Importante)

O sistema deve permitir a um funcionário buscar um cliente pelo CPF.

RF18 Cadastrar espécie. (Pr.: Essencial)

O sistema deve permitir que um funcionário cadastre uma espécie de animal.

RF19 Excluir espécie. (Pr.: Desejável)

O sistema deve permitir que um funcionário exclua uma espécie de animal cadastrado.

RF20 Cadastrar Animal . (Pr.: Essencial)

O sistema deve permitir a um funcionário cadastrar um animal.

RF21 Excluir animal. (Pr.: Desejável)

O sistema deve permitir que um funcionário exclua um animal previamente cadastrado.

RF22 Cadastrar cultura agrícola. (Pr.: Essencial)

O sistema deve permitir que um funcionário cadastre uma nova cultura para o plantio.

RF23 Excluir cultura agrícola. (Pr.: Desejável)

O sistema deve permitir que um funcionário exclua culturas agrícolas.

RF24 Cadastrar plantio de cultura. (Pr.: Essencial)

O sistema deve permitir que um funcionário cadastre o plantio de uma cultura.

RF25 Excluir plantio de cultura. (Pr.: Importante)

O sistema deve permitir que um funcionário exclua o plantio de uma cultura.

RF26 Cadastrar um produto. (Pr.: Essencial)

O sistema deve permitir que um funcionário cadastre um produto.

RF27 Excluir um Produto. (Pr.: Importante)

O sistema deve permitir que um funcionário exclua um produto previamente cadastrado.

RF28 Buscar produto pelo nome. (Pr.: Desejável)

O sistema deve permitir que um funcionário busque através do nome um produto dentre os disponíveis para a venda.

RF29 Listar Produtos. (Pr.: Importante)

O sistema deve permitir que um funcionário possa listar os produtos disponíveis para venda.

RF30 Adicionar itens de um produto. (Pr.: Essencial)

O sistema deve permitir que um funcionário adicione itens de um produto.

RF31 Excluir itens de um produto. (Pr.: Essenciall)

O sistema deve permitir que um funcionário exclua itens de um produto.

RF32 Criar venda. (Pr.: Essencial)

O sistema deve permitir que um funcionário crie uma venda.

RF33 Gerar nota fiscal. (Pr.: Importante)

Ao concluir uma venda, o sistema deve gerar uma nota fiscal com a quantidade de itens comprados, o preço de cada produto, o valor pago, o troco e o CPF do cliente.

RF34 Adicionar item na venda. (Pr.: Essencial)

O sistema deve permitir que um funcionário adicione um item de um produto na venda.

RF35 Remover item na venda. (Pr.: Essencial)

O sistema deve permitir que um funcionário remova um item de um produto na venda.

RF36 Gerar relatório de vendas. (Pr.: Desejável)

O sistema deve gerar um relatório com todos os produtos vendidos e a respectiva quantidade no mês corrente.

RF37 Armazenar vendas. (Pr.: Importante)

O sistema deve permitir que toda venda realizada seja armazenada.

1.8.2 Requisitos Não Funcionais

RNF1 (Pr.: Importante): O sistema deve retornar as consultas, ou seja, prover a exibição dos dados, em, no máximo, 3 segundos, em 95% dos casos.

RNF2 (Pr.: Desejável): O sistema deve excluir dados, em no máximo, 10 segundos, em 90% dos casos.

RNF3 (Pr.: Importante): O sistema deve incluir dados em, no máximo, 3 segundos, em 90% dos casos.

RNF4 (Pr.: Essencial): O sistema deve gerar relatórios em, no máximo, 8 segundos, em 90% dos casos.

2 Plano de Projeto

Devido ao avanço tecnológico e científico, a humanidade trabalha com um número cada vez maior de informações. Para lidar com a grande quantidade de dados, é necessário automatizar informações e por isso muitas empresas buscam soluções computacionais para auxiliar a automatizar as informações com que elas trabalham incluindo controle e gestão de suas atividades.

Esse projeto tem como objetivo a criação de um sistema para gerenciamento de fazenda, permitindo o controle de diversas funções como: estoque, atendimento e gestão de funcionários. Esse sistema será integrado entre vários computadores de forma a permitir uma maior eficiência tanto no atendimento ao cliente como no controle das funções da fazenda. Devido a essa necessidade é necessário ter vários computadores conectados em rede.

2.1 Motivação

As atividades realizadas em uma fazenda produzem uma grande quantidade de informações como: vendas de produtos, contratação de funcionários e gerenciamento de estoque. Devido a isso, surgem diversos desafios como:

- Gerência de grande quantidade de informações de funcionários.
- Garantia da eficiência das vendas.
- Garantia de produtividade.
- Gerência de cálculo das vendas e dos custos da fazenda.
- Garantia de qualidade dos produtos.

3 Arquitetura de Software (Arquitetura 4+1)

Foi utilizado no presente projeto o padrão de arquitetura 4 + 1. Assumindo como definição de arquitetura o conceito proposto por Kruchten, no qual arquitetura de software lida com design e implementação de alto nível de software, sendo composta por elementos arquitetônicos em formas específicas que satisfaçam as funcionalidades, desempenho e requisitos não funcionais do sistema (KRUCHTEN, 1995).

Já a motivação para a escolha da arquitetura 4+1 se deu pelo fato de sua característica genérica e de independência em relação a notações e métodos, além do diferencial de oferecer um modelo bastante completo baseado em visões e com uma visão mais generalista do todo do projeto. Vale ressaltar que cada visão foi trabalhada a luz de uma perspectiva particular pertinente a visão adotada, as visões que compõem o 4+1 são: lógica, processo, implementação, implantação, cenário.

3.1 Visão Lógica

Corresponde aos requisitos funcionais do projeto, ou seja, o serviço que o sistema deverá oferecer para os seus usuários finais. Também diz respeito a identificação dos principais pacotes, subsistemas e classes. Para se ter uma visualização adequada desta visão, foram adotados no projeto os diagramas de classes e sequência.

3.2 Visão de Processo

Trata da visão dos processos que formam os mecanismos de concorrência e sincronização do sistema. Mostra como e onde serão usadas as classes observadas através da visão lógica. Para se ter uma visualização adequada desta visão, foram utilizados os diagramas de sequência, atividades e de classes com foco em classes ativas.

3.3 Visão de Implementação

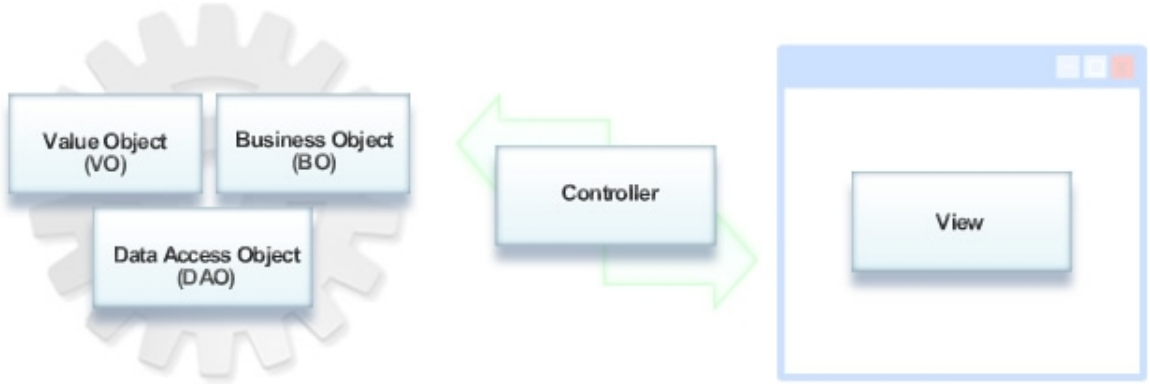
Trata da organização de módulos do software (código, arquivos de dados, componentes, executáveis, bibliotecas) para a montagem e fornecimento do sistema. Seus principais elementos são módulos, subsistemas e camadas. Considera requisitos como facilidade de desenvolvimento, restrições do ambiente e das linguagens de programação. Essa visão serve como base para alocação de requisitos e trabalhos entre a equipe; avaliação de custos; monitoração do projeto; reuso, portabilidade e segurança. No presente projeto foram utilizadas regras de separação de código de acordo com as funcionalidades, toda a aplicação foi distribuída em camadas tendo como objetivo a maior independência possível entre elas. As camadas definidas foram:

- Model - Trata-se da principal camada da aplicação, sua função se restringe a fornecer todas as funcionalidades do software independente de interação com o usuário ou interface gráfica. No projeto ela foi dividida em três tipos de classe
  - Value Objects (VO) - Denominada de objeto de valores, contém classes com variáveis e métodos de acesso, além de construtores. É responsável por modelar a entidade, uma abstração do mundo real.

  - Data Acess Object (DAO) - Essas classes são as responsáveis pelo acesso à dados como fluxo de rede, arquivo ou banco de dados.
  - Business Object (BO) - A especialidade desse tipo de classe é resolver operações complexas, trata-se dos processos principais da aplicação, pois nela são trabalhadas as de negócio e tomadas de decisão.

- View - Trata-se da camada onde há interação com o usuário, possui os objetos de fronteira e as telas do sistema.
- Controller - Trata-se da camada onde encontram-se as classes controladoras de eventos, responsáveis em fazer a interação entre a camada model e a view, ou seja, a ponte de comunicação entre os objetos de fronteira e entidades.

Figura 1: Diagrama de camadas utilizado no Sistema de Gestão de Fazenda



Fonte: Disponível em:
http://www.gqferreira.com.br/artigos/ver/mvc-com-java-desktop-parte1

3.4 Visão de Implantação

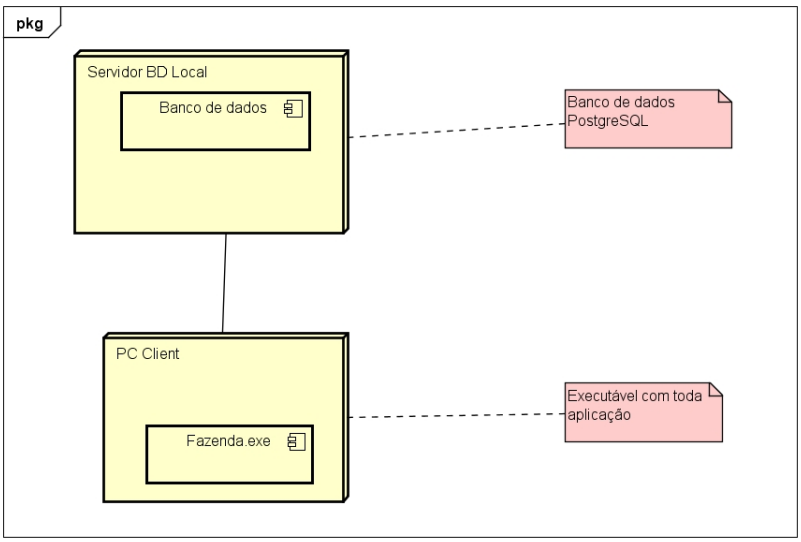
Trata-se da visão composta pelos nós que formam a topologia de hardware em que o sistema é executado. É uma visão direcionada principalmente a distribuição, o fornecimento e implantação das partes que formam o sistema físico.

O presente sistema consiste de uma aplicação desktop rodando localmente que funciona como cliente e servidor, ou seja, requisitando recursos próprios. O sistema possui um banco de dados PostgreSQL que serão executados com toda aplicação através do arquivo *Fazenda.exe*.

3.5 Visão de Cenários

Trata-se da exibição de como os elementos das 4 visões trabalham juntos, abrange os casos de uso que mostram como o sistema se comporta segundo a perspectiva dos usuários finais, analistas e pessoal de teste. É de responsabilidade dessa visão especificar as forças que definem a forma da arquitetura do sistema. Assim, seus principais objetivos são a

Figura 2: Diagrama de implantação do modelo cliente-servidor adaptado para uma aplicação local



direção em busca dos elementos de arquitetura; validação e ilustração da arquitetura de software; ponto de partida para desenvolvimento e testes.

O sistema possui 35 casos de uso:

- 1. Cadastrar funcionário: Um funcionário cadastra um funcionário.
- 2. Cadastrar cargo: Um funcionário cadastra um funcionário.
- 3. Excluir cargo: Um funcionário exclui um cargo.
- 4. Definir contrato: Um funcionário define contrato.
- 5. Definir endereço: Um funcionário define endereço.
- 6. Alterar dados de funcionário: Um funcionário altera dados do funcionário.
- 7. Buscar funcionário por CPF: Um funcionário busca um funcionário por CPF.
- 8. Listar contratos de funcionários: Um funcionário lista contratos de funcionários.
- 9. Autenticar funcionário: Um funcionário autentica um funcionário.
- 10. Inativar contrato do funcionário: Um funcionário altera dados do funcionário.
- 11. Listar funcionários contratados: Um funcionário lista funcionários contratados.
- 12. Listar funcionários a partir do cargo: Um funcionário lista funcionários a partir do cargo.
- 13. Deslogar do sistema: Um funcionário desloga do sistema.
- 14. Cadastrar um cliente: Um funcionário cadastra um cliente.

- 15. Alterar dados do cliente: Um funcionário altera dados do cliente.
- 16. Buscar cliente por CPF: Um funcionário busca um cliente por CPF.
- 17. Cadastrar espécie: Um funcionário cadastra espécie.
- 18. Excluir espécie: Um funcionário exclui espécie.
- 19. Cadastrar animal: Um funcionário cadastra animal.
- 20. Excluir animal: Um funcionário exclui animal.
- 21. Cadastrar cultura agrícola: Um funcionário cadastra cultura agrícola.
- 22. Excluir cultura agrícola: Um funcionário exclui cultura agrícola.
- 23. Cadastrar plantio de cultura: Um funcionário cadastra plantio de cultura.
- 24. Cadastrar produto: Um funcionário cadastra produto.
- 25. Excluir produto: Um funcionário exclui produto.
- 26. Buscar produto pelo nome: Um funcionário busca produto pelo nome.
- 27. Listar produtos: Um funcionário lista produtos.
- 28. Adicionar itens de um produto: Um funcionário adiciona itens de um produto.
- 29. Remover itens de um produto: Um funcionário remove itens de um produto.
- 30. Criar venda: Um funcionário cria venda.
- 31. Gerar nota fiscal: Um funcionário gera nota fiscal.
- 32. Adicionar item da venda: Um funcionário adiciona item da venda.
- 33. Remover item da venda: Um funcionário remove item da venda.
- 34. Gerar relatório de vendas: Um funcionário gera relatório de vendas.
- 35. Armazenar vendas: Um funcionário armazena vendas.

4 Casos de Uso

Nome: Cadastrar Funcionário.
Descrição: O funcionário cadastra um outro funcionário no sistema.
Identificador: CDU01.
Importância: Essencial.
Ator Primário: Funcionário1.
Fluxo Principal:

Sistema	Funcionário1	Funcionário2
	1 - Solicita dados do funcionário2	
		2 - Fornece dados ao funcionário1
	3 - Insere os dados do funcionário2 no sistema	
4 - Solicita ao funcionário1 confirmação dos dados		
	5 - Confirma dados	
6 - Regitra os dados		
7 - Finaliza a Operação		

Nome: Buscar Funcionário.
Descrição: O funcionário busca funcionários no sistema a partir do CPF.
Identificador: CDU02.
Importância: Desejável.
Ator Primário: Funcionário.
Fluxo Principal:

Sistema	Funcionário
	1 - Informa CPF do funcionário a ser buscado
2 - Reliza a busca a partir do CPF	
3 - Exibe o funcionário encontrado	
	4a - Solicita edição dos dados do Funcionário 4b - Solicita a contratação do Funcionário 4c - Solicita a inativação do Funcionário 4d - Solicita o retorno à tela inicial
5a - Encaminha para tela de edição de Funcionário 5b - Encaminha para tela de contratação de Funcionário 5c - Encaminha para tela de inativação de Funcionário 5d - Encaminha para tela inicial	
6 - Finaliza a Operação	

Nome: Cadastrar Cliente.
Descrição: O funcionário cadastra um Cliente no sistema.
Identificador: CDU03.
Importância: Essencial.
Ator Primário: Funcionário.
Fluxo Principal:

Sistema	Funcionário	Cliente
	1 - Solicita dados do cliente	
		2 - Fornece dados ao funcionário
	3 - Insere os dados do cliente no sistema	
4 - Solicita ao funcionário confirmação dos dados		
	5 - Confirma dados	
6 - Regitra os dados		
7 - Finaliza a operação		

Nome: Criar Venda.
Descrição: O funcionário realiza uma venda.
Identificador: CDU04.
Importância: Essencial.
Ator Primário: Funcionário.
Fluxo Principal:

Sistema	Funcionário	Cliente
	1 - Solicita CPF do cliente	
		2 - Fornece CPF ao funcionário
	3 - Insere o CPF do cliente no sistema	
4 - Exibe lista de itens disponíveis		
	5 - Consulta o cliente para saber quais itens ele deseja	
		6 - Informa ao funcionário os itens que deseja comprar
	7 - Adiciona itens à venda	
	8 - Realiza a venda	
9 - Solicita ao funcionário a confirmação dos dados		
	10 - Confirma Dados	
11 - Registra os dados		
	12a - Solicita relatório de vendas 12b - Solicita Nota Fiscal	
13a - Gera relatório de vendas 13b - Gera Nota Fiscal		
14 - Finaliza a operação		

Nome: Adicionar itens em estoque.
Descrição: O funcionário adiciona itens ao estoque.
Identificador: CDU05.
Importância: Essencial.
Ator Primário: Funcionário.
Fluxo Principal:

Sistema	Funcionário
	1 - Informa produto do qual será inserida uma quantidade de itens ao estoque
2 - Reliza a busca pelo produto cadastrado	
3 - Solicita a quantidade de itens que serão adicionados	
	4 - Informa a quantidade de itens a serem adicionados
5 - Solicita conformação do funcionário	
	6 - Confirma a quantidade a ser inserida em estoque
7 - Registra os dados	
8 - Finaliza a operação	

5 Codificação Java Modelos

5.1 Value Objects (VO)

5.1.1 Animal

```java
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class Animal {

    private int id;
    private Especie especie;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Especie getEspecie() {
        return especie;
    }

    public void setEspecie(Especie especie) {
        this.especie = especie;
    }


}
```

Código 1: Animal.java

5.1.2 Caixa

```java
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class Caixa {

    private float dinheiro;

    public float getDinheiro() {
        return dinheiro;
    }

    public void setDinheiro(float dinheiro) {
        this.dinheiro = dinheiro;
    }


}
```

Código 2: Caixa.java

5.1.3 Cargo

```java
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class Cargo {

    private int id;
    private String nome;

    public String getNome() {
        return nome;
```

```
14      }
15
16      public void setNome(String nome) {
17          this.nome = nome;
18      }
19
20      public int getId() {
21          return id;
22      }
23
24      public void setId(int id) {
25          this.id = id;
26      }
27  }
```

Código 3: Cargo.java

5.1.4 Cliente

```
1   package br.com.afazenda.model.vo;
2
3   /**
4    * @author givaldo
5    */
6   public class Cliente {
7
8       private String cpf;
9       private String telefone;
10      private String nome;
11      private String email;
12      private int endereco;
13      private boolean inadiplente;
14
15      public Cliente(String cpf, String nome,String telefone,String email, int endereco, boolean inadiplente) {
16          this.cpf =cpf;
17          this.nome = nome;
18          this.telefone = telefone;
19          this.email = email;
20          this.endereco = endereco;
21          this.inadiplente = inadiplente;
22      }
23
24      public Cliente() {
25
26      }
27
28      public String getCpf() {
29          return cpf;
30      }
31
32      public void setCPF(String cpf) {
33          this.cpf = cpf;
34      }
35
36      public String getNome() {
37          return nome;
38      }
39
40      public void setNome(String nome) {
41          this.nome = nome;
42      }
43
44      public String getEmail() {
45          return email;
46      }
47
48      public void setEmail(String email) {
49          this.email = email;
50      }
51
52      public int getEndereco() {
53          return endereco;
54      }
55
56      public void setEndereco(int endereco) {
57
58          this.endereco = endereco;
59      }
60
61      public boolean isInadiplente() {
62          return inadiplente;
63      }
64
65      public void setInadiplente(boolean inadiplente) {
66          this.inadiplente = inadiplente;
67      }
68
69      public String getTelefone() {
```

```
70          return telefone;
71      }
72
73      public void setTelefone(String telefone) {
74          this.telefone = telefone;
75      }
76
77
78  }
```

Código 4: Cliente.java

5.1.5 Colheita

```
1   package br.com.afazenda.model.vo;
2
3   /**
4    *
5    * @author givaldo
6    */
7   public class Colheita {
8
9       private CulturaAgricola cultura;
10      private String dataColheita;
11
12      public CulturaAgricola getCultura() {
13          return cultura;
14      }
15
16      public void setCultura(CulturaAgricola cultura) {
17          this.cultura = cultura;
18      }
19
20      public String getDataColheita() {
21          return dataColheita;
22      }
23
24      public void setDataColheita(String dataColheita) {
25          this.dataColheita = dataColheita;
26      }
27
28
29  }
```

Código 5: Colheita.java

5.1.6 Contrato

```
1   package br.com.afazenda.model.vo;
2
3   /**
4    *
5    * @author givaldo
6    */
7   public class Contrato {
8
9       private int id;
10      private String dataContratacao;
11      private String dataTermino;
12      private float salario;
13      private boolean ativo;
14      private Cargo cargo;
15      private Funcionario funcionario;
16
17      public String getDataContratacao() {
18          return dataContratacao;
19      }
20
21      public void setDataContratacao(String dataContratacao) {
22          this.dataContratacao = dataContratacao;
23      }
24
25      public String getDataTermino() {
26          return dataTermino;
27      }
28
29      public void setDataTermino(String dataTermino) {
30          this.dataTermino = dataTermino;
31      }
32
33      public boolean isAtivo() {
34          return ativo;
```

```
    }

    public void setAtivo(boolean isAtivo) {
        this.ativo = isAtivo;
    }

    public float getSalario() {
        return salario;
    }

    public void setSalario(float salario) {
        this.salario = salario;
    }

    public Funcionario getFuncionario() {
        return funcionario;
    }

    public void setFuncionario(Funcionario funcionario) {
        this.funcionario = funcionario;
    }

    public Cargo getCargo() {
        return cargo;
    }

    public void setCargo(Cargo cargo) {
        this.cargo = cargo;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
```

Código 6: Contrato.java

5.1.7 Cultura Agrícola

```
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class CulturaAgricola {

    private int id;
    private String nome;
    private String periodoPlantio;
    private String periodoColheita;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getPeriodoPlantio() {
        return periodoPlantio;
    }

    public void setPeriodoPlantio(String periodoPlantio) {
        this.periodoPlantio = periodoPlantio;
    }

    public String getPeriodoColheita() {
        return periodoColheita;
    }

    public void setPeriodoColheita(String periodoColheita) {
        this.periodoColheita = periodoColheita;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
```

Código 7: CulturaAgricola.java

5.1.8 Endereço

```java
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class Endereco {
    private String logradouro;
    private int numero;
    private String bairro;
    private String cidade;
    private String estado;
    private String cep;

    public Endereco(String logradouro, int numero, String bairro, String cidade, String estado, String cep) {
        this.logradouro = logradouro;
        this.numero = numero;
        this.bairro = bairro;
        this.cidade = cidade;
        this.estado = estado;
        this.cep = cep;
    }
    public Endereco(){

    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    public String getBairro() {
        return bairro;
    }

    public void setBairro(String bairro) {
        this.bairro = bairro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public void setCep(String cep){ this.cep=cep;}

    public String getCep() {return  this.cep;}

    public String getEndereco() {
        //rua cicero soares santos, n   238. Cidade Nova, Aracaju - SE. CEP: 49070-820
        return this.logradouro + ", " + "n  " + this.numero + ". " + this.bairro + ", " + this
                .cidade + " - " + this.estado + ". CEP: " + this.cep;
    }


}
```

Código 8: Endereco.java

5.1.9 Espécie

```java
package br.com.afazenda.model.vo;

import java.util.ArrayList;

/**
 *
 * @author givaldo
 */
public class Especie {
    private int id;
    private String name;
    private ArrayList<String> produtosGerados;
    private int quantidade;

    public Especie(int id_especie, String nome, int quantidade) {
    }

    public Especie() {

    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public ArrayList<String> getProdutosGerados() {
        return produtosGerados;
    }

    public void setProdutosGerados(ArrayList<String> produtosGerados) {
        this.produtosGerados = produtosGerados;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }



}
```

Código 9: Especie.java

5.1.10 Funcionário

```java
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class Funcionario {

    private String cpf;
    private String name;
    private String telefone;
    private String email;
    private int endereco;
    private boolean ativo;
    private String nascimento;

    public Funcionario(String cpf, String name, String telefone, String email, int endereco,
                       String nascimento, Boolean ativo) {
        this.cpf = cpf;
        this.name = name;
        this.telefone = telefone;
        this.email = email;
        this.endereco = endereco;
        this.nascimento = nascimento;
```

```
25      }
26
27      public Funcionario() {
28
29      }
30
31      public String getNascimento() {
32          return nascimento;
33      }
34
35      public void setNascimento(String nascimento) {
36          this.nascimento = nascimento;
37      }
38
39      public String getCpf() {
40          return cpf;
41      }
42
43      public void setCpf(String cpf) {
44          this.cpf = cpf;
45      }
46
47      public String getName() {
48          return name;
49      }
50
51      public void setName(String name) {
52          this.name = name;
53      }
54
55      public String getTelefone() {
56          return telefone;
57      }
58
59      public void setTelefone(String telefone) {
60          this.telefone = telefone;
61      }
62
63      public String getEmail() {
64          return email;
65      }
66
67      public void setEmail(String email) {
68          this.email = email;
69      }
70
71      public int getEndereco() {
72          return endereco;
73      }
74
75      public void setEndereco(int endereco) {
76          this.endereco = endereco;
77      }
78
79      public  boolean isAtivo(){ return  this.ativo;}
80
81      public  void setAtivo(Boolean ativo){ this.ativo = ativo;}
82
83  }
```

Código 10: Funcionario.java

5.1.11 Insumo Animal

```
1   package br.com.afazenda.model.vo;
2
3   /**
4    *
5    * @author givaldo
6    */
7   public class InsumoAnimal {
8
9   }
```

Código 11: InsumoAnimal.java

5.1.12 Item Produto

```
1   package br.com.afazenda.model.vo;
2
3   /**
4    *
```

```
 * @author givaldo
 */
public class ItemProduto {

    private int quantidade;
    private float preco;
    private Produto produto;

    public float getPreco() {
        return preco;
    }

    public void setPreco(float preco) {
        this.preco = preco;
    }

    public Produto getProduto() {
        return produto;
    }

    public void setProduto(Produto produto) {
        this.produto = produto;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }
}
```

Código 12: ItemProduto.java

5.1.13 Nota Fiscal

```
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo
 */
public class NotaFiscal {

    private Venda venda;
    private float imposto;

    public Venda getVenda() {
        return venda;
    }

    public void setVenda(Venda venda) {
        this.venda = venda;
    }

    public float getImposto() {
        return imposto;
    }

    public void setImposto(float imposto) {
        this.imposto = imposto;
    }



}
```

Código 13: NotaFiscal.java

5.1.14 Plantação

```
package br.com.afazenda.model.vo;

/**
 *
 * @author givaldo && Patrick Jones
 */
public class Plantacao {
    private int idPlantacao;
    private String dataPlantacao;
    private CulturaAgricola cultura;
```

```
11
12      public String getDataPlantacao() {
13          return dataPlantacao;
14      }
15
16      public void setDataPlantacao(String dataPlantacao) {
17          this.dataPlantacao = dataPlantacao;
18      }
19
20      public CulturaAgricola getCultura() {
21          return cultura;
22      }
23
24      public void setCultura(CulturaAgricola cultura) {
25          this.cultura = cultura;
26      }
27
28      public int getIdPlantacao() {
29          return idPlantacao;
30      }
31
32      public void setIdPlantacao(int idPlantacao) {
33          this.idPlantacao = idPlantacao;
34      }
35
36  }
```

Código 14: Plantacao.java

5.1.15 Produto

```
1   package br.com.afazenda.model.vo;
2
3   /**
4    * Created by Givaldo Marques
5    * on 01/05/17.
6    */
7   public class Produto {
8
9       private int id;
10      private int quantidade;
11      private float preco;
12      private String medida;
13      private int quantidaEstoque;
14      private String nome;
15      private char origem;
16      private String data;
17
18      public int getId() {
19          return id;
20      }
21
22      public void setId(int id) {
23          this.id = id;
24      }
25
26      public int getQuantidade() {
27          return quantidade;
28      }
29
30      public void setQuantidade(int quantidade) {
31          this.quantidade = quantidade;
32      }
33
34      public float getPreco() {
35          return preco;
36      }
37
38      public void setPreco(float preco) {
39          this.preco = preco;
40      }
41
42      public String getMedida() {
43          return medida;
44      }
45
46      public void setMedida(String medida) {
47          this.medida = medida;
48      }
49
50      public int getQuantidaEstoque() {
51          return quantidaEstoque;
52      }
53
54      public void setQuantidaEstoque(int quantidaEstoque) {
55          this.quantidaEstoque = quantidaEstoque;
56      }
57
```

```
58      public String getNome() {
59          return nome;
60      }
61
62      public void setNome(String nome) {
63          this.nome = nome;
64      }
65
66      public char getOrigem() {
67          return origem;
68      }
69
70      public void setOrigem(char origem) {
71          this.origem = origem;
72      }
73
74      public String getData() {
75          return data;
76      }
77
78      public void setData(String data) {
79          this.data = data;
80      }
81  }
```

Código 15: Produto.java

5.1.16 Venda

```
1   package br.com.afazenda.model.vo;
2
3   import java.util.ArrayList;
4
5   /**
6    *
7    * @author givaldo
8    */
9   public class Venda {
10
11      private String date;
12      private ArrayList<ItemProduto> itens;
13      private Funcionario funcionario;
14      private Cliente cliente;
15
16      private float valorVenda;
17      private float valorPago;
18
19      public String getDate() {
20          return date;
21      }
22
23      public void setDate(String date) {
24          this.date = date;
25      }
26
27      public ArrayList<ItemProduto> getItens() {
28          return itens;
29      }
30
31      public void setItens(ArrayList<ItemProduto> itens) {
32          this.itens = itens;
33      }
34
35
36      public Cliente getCliente() {
37          return cliente;
38      }
39
40      public void setCliente(Cliente cliente) {
41          this.cliente = cliente;
42      }
43
44      public Funcionario getFuncionario() {
45          return funcionario;
46      }
47
48      public void setFuncionario(Funcionario funcionario) {
49          this.funcionario = funcionario;
50      }
51
52      public float getValorPago() {
53          return this.valorPago;
54      }
55
56      public void setValorPago(float valorPago) {
57          this.valorPago = valorPago;
58      }
59
```

```
60      public float getValorVenda() {
61          return valorVenda;
62      }
63
64      public void setValorVenda(float valorVenda) {
65          this.valorVenda = valorVenda;
66      }
67
68      public float getTroco() {
69          return this.valorPago - this.valorVenda;
70      }
71
72  }
```

Código 16: Venda.java

5.2 Data Acess Object (DAO)

5.2.1 Agricultura Imp

```
1   package br.com.afazenda.model.dao;
2
3   import br.com.afazenda.model.interfacedao.AgriculturaDao;
4   import br.com.afazenda.model.vo.CulturaAgricola;
5   import br.com.afazenda.model.vo.Plantacao;
6
7   import java.sql.*;
8   import java.util.ArrayList;
9
10  /**
11   * Created by Givaldo Marques && Patrick Jones
12   * on 01/05/17.
13   */
14  public class AgriculturaImp implements AgriculturaDao {
15      private Connection conexao;
16
17      public AgriculturaImp(Connection conexao) {
18          this.conexao = conexao;
19      }
20
21      @Override
22      public void cadastrarCulturaAgricola(CulturaAgricola culturaAgricola) throws SQLException {
23          PreparedStatement preparedStatement = conexao.prepareStatement("INSERT␣INTO␣fazenda_bd" +
24                  ".cultura_agricola(nome,␣periodoplantio,␣periodocolheita)␣VALUES(␣?,␣?,␣?)");
25          preparedStatement.setString(1, culturaAgricola.getNome());
26          preparedStatement.setString(2, culturaAgricola.getPeriodoPlantio());
27          preparedStatement.setString(3, culturaAgricola.getPeriodoColheita());
28
29          preparedStatement.execute();
30      }
31
32      @Override
33      public void excluirCulturaAgricola(CulturaAgricola culturaAgricola) throws SQLException {
34          PreparedStatement preparedStatement = conexao.prepareStatement("DELETE␣FROM␣fazenda_bd" +
35                  ".cultura_agricola␣WHERE␣idcultura␣=␣(?)");
36          preparedStatement.setInt(1,culturaAgricola.getId());
37          preparedStatement.execute();
38      }
39      @Override
40      public void excluirPlantioDeCultura(Plantacao plantacao) throws SQLException {
41          PreparedStatement preparedStatement = conexao.prepareStatement("DELETE␣FROM␣fazenda_bd" +
42                  ".plantio␣WHERE␣idplantio␣=␣(?)");
43          preparedStatement.setInt(1,plantacao.getIdPlantacao());
44          preparedStatement.execute();
45      }
46
47      @Override
48      public void cadastarPlantio(Plantacao plantacao) throws SQLException {
49          PreparedStatement preparedStatement = conexao.prepareStatement("INSERT␣INTO␣fazenda_bd" +
50                  ".plantio(cultura,␣dataplantacao)␣VALUES␣(?,␣?)");
51          preparedStatement.setInt(1, plantacao.getCultura().getId());
52          preparedStatement.setString(2, String.valueOf(plantacao.getDataPlantacao()));
53
54          preparedStatement.execute();
55      }
56
57      @Override
58      public ArrayList<CulturaAgricola> listarCulturaAgricola() throws SQLException {
59          ArrayList<CulturaAgricola> culturaAgricola = new ArrayList<>();
60
61          String query = "SELECT␣*␣FROM␣fazenda_bd.cultura_agricola;";
62
63          Statement statement = conexao.createStatement();
64          ResultSet resultSet = statement.executeQuery(query);
65
66          while(resultSet.next()){
```

```
67              CulturaAgricola c = new CulturaAgricola();
68              c.setId(resultSet.getInt(1));
69              c.setNome(resultSet.getString(2));
70              c.setPeriodoPlantio(resultSet.getString(3));
71              c.setPeriodoColheita(resultSet.getString(4));
72
73              culturaAgricola.add(c);
74          }
75          return  culturaAgricola;
76      }
77
78      @Override
79      public ArrayList<Plantacao> listarPlantioDeCultura() throws SQLException {
80          ArrayList<Plantacao> plantacao = new ArrayList<>();
81
82          String query = "SELECT * FROM fazenda_bd.plantio NATURAL JOIN fazenda_bd.cultura_agricola WHERE cultura =
        cultura_agricola.idcultura;";
83
84          Statement statement = conexao.createStatement();
85          ResultSet resultSet = statement.executeQuery(query);
86
87          while(resultSet.next()){
88              Plantacao p = new Plantacao();
89              p.setIdPlantacao(resultSet.getInt(1));
90              CulturaAgricola c = new CulturaAgricola();
91              c.setId(resultSet.getInt(4));
92              c.setNome(resultSet.getString(5));
93              c.setPeriodoPlantio(resultSet.getString(6));
94              c.setPeriodoColheita(resultSet.getString(7));
95              p.setCultura(c);
96              p.setDataPlantacao(resultSet.getString(3));
97
98              plantacao.add(p);
99          }
100         return  plantacao;
101     }
102 }
```

Código 17: AgriculturaImp.java

5.2.2 Animal Imp

```
1   /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package br.com.afazenda.model.dao;
7
8   import br.com.afazenda.model.interfacedao.AnimalDao;
9   import br.com.afazenda.model.vo.Animal;
10  import br.com.afazenda.model.vo.Especie;
11
12  import java.sql.*;
13  import java.util.ArrayList;
14
15  /**
16   * @author givaldo
17   */
18  public class AnimalImp implements AnimalDao {
19
20      private Connection conexao;
21
22      public AnimalImp(Connection conexao) {
23          this.conexao = conexao;
24
25      }
26
27      public void insereAnimal(Animal animal) throws SQLException {
28          PreparedStatement preparedStatement = conexao.prepareStatement("INSERT INTO fazenda_bd" +
29                  ".animal(especie) VALUES (?)");
30          preparedStatement.setInt(1, animal.getEspecie().getId());
31
32          preparedStatement.execute();
33      }
34
35      public void removeAnimal(Animal animal) throws SQLException {
36          PreparedStatement preparedStatement = conexao.prepareStatement("DELETE FROM fazenda_bd" +
37                  ".animal WHERE idanimal =(?)");
38          preparedStatement.setInt(1, animal.getId());
39          preparedStatement.execute();
40      }
41
42      public ArrayList<Animal> getAnimais() throws SQLException {
43          ArrayList<Animal> animais = new ArrayList<>();
44
45          String query = "SELECT * FROM fazenda_bd.animal NATURAL JOIN fazenda_bd.especie WHERE especie = idespecie;";
46
```

```
47          Statement comando = conexao.createStatement();
48          ResultSet resultado = comando.executeQuery(query);
49
50          while (resultado.next()) {
51
52              Animal a = new Animal();
53
54              a.setId(resultado.getInt(1));
55              Especie e = new Especie();
56              e.setId(resultado.getInt(2));
57              e.setName(resultado.getString(4));
58              e.setQuantidade(resultado.getInt(5));
59              a.setEspecie(e);
60
61              animais.add(a);
62          }
63
64          return animais;
65      }
66
67  }
```

Código 18: AnimalImp.java

5.2.3 Cargo Imp

```
1   package br.com.afazenda.model.dao;
2
3   import br.com.afazenda.model.interfacedao.CargoDao;
4   import br.com.afazenda.model.vo.Cargo;
5
6   import java.sql.*;
7   import java.util.ArrayList;
8
9
10  /**
11   * Created by Givaldo Marques
12   * on 01/05/17.
13   */
14  public class CargoImp implements CargoDao {
15      private Connection conexao = null;
16
17      public CargoImp(Connection conexao) {
18          this.conexao = conexao;
19      }
20
21      @Override
22      public void cadastrarCargo(Cargo cargo) throws SQLException {
23          PreparedStatement preparedStatement = conexao.prepareStatement("INSERT␣INTO␣fazenda_bd" +
24                  ".cargo(nome)␣VALUES␣(?)");
25          preparedStatement.setString(1, cargo.getNome());
26
27          preparedStatement.execute();
28      }
29
30      @Override
31      public void excluirCargo(Cargo cargo) throws SQLException {
32          PreparedStatement preparedStatement = conexao.prepareStatement("DELETE␣FROM␣fazenda_bd" +
33                  ".cargo␣WHERE␣nome␣=␣?");
34          preparedStatement.setString(1, cargo.getNome());
35
36          preparedStatement.execute();
37      }
38
39      @Override
40      public ArrayList<Cargo> listarCargos() throws SQLException {
41          ArrayList<Cargo> cargosList = new ArrayList<>();
42
43          String query = "SELECT␣*␣FROM␣fazenda_bd.cargo;";
44
45          Statement statement = conexao.createStatement();
46          ResultSet resultSet = statement.executeQuery(query);
47
48          while (resultSet.next()) {
49              Cargo c = new Cargo();
50              c.setNome(resultSet.getString(2));
51              c.setId(resultSet.getInt(1));
52              cargosList.add(c);
53          }
54
55          return cargosList;
56      }
57
58  }
```

Código 19: CargoImp.java

5.2.4 Cliente Imp

```
1  package br.com.afazenda.model.dao;
2
3  import br.com.afazenda.model.interfacedao.ClienteDao;
4  import br.com.afazenda.model.vo.Cliente;
5
6  import java.sql.*;
7  import java.util.ArrayList;
8
9  /**
10  * Created by Givaldo Marques
11  * on 25/04/17.
12  */
13 public class ClienteImp implements ClienteDao {
14
15     private Connection conexao;
16
17     public ClienteImp(Connection conexao) throws SQLException {
18         this.conexao = conexao;
19     }
20
21
22     @Override
23     public void cadastrarCliente(Cliente cliente) throws SQLException {
24         PreparedStatement preparedStatement = conexao.prepareStatement("INSERT␣INTO␣fazenda_bd.cliente(cpf,␣nome,␣
       telefone,␣email,␣idendereco)␣VALUES␣(?,?,?,?,?)"
25         );
26
27         preparedStatement.setString(1, cliente.getCpf());
28         preparedStatement.setString(2,cliente.getNome());
29         preparedStatement.setString(3,cliente.getTelefone());
30         preparedStatement.setString(4,cliente.getEmail());
31         preparedStatement.setInt(5,cliente.getEndereco());
32         preparedStatement.execute();
33     }
34
35     @Override
36     public ArrayList<Cliente> buscarClientes() throws SQLException {
37         ArrayList<Cliente> clientesList = new ArrayList<>();
38
39         String query = "SELECT␣*␣FROM␣fazenda_bd.cliente;";
40
41         Statement statement = conexao.createStatement();
42         ResultSet resultSet = statement.executeQuery(query);
43
44         while (resultSet.next()) {
45             Cliente c = new Cliente();
46             c.setCPF(resultSet.getString(1));
47             c.setNome(resultSet.getString(2));
48             c.setTelefone(resultSet.getString(3));
49             c.setEndereco(resultSet.getInt(4));
50             c.setEmail(resultSet.getString(5));
51
52             clientesList.add(c);
53         }
54
55         return clientesList;
56     }
57
58
59     @Override
60     public Cliente buscarCliente(String cpfClinte) throws SQLException {
61         PreparedStatement preparedStatement = conexao.prepareStatement("SELECT␣*␣FROM␣fazenda_bd.cliente␣" +
62                 "WHERE␣cliente.cpf␣=␣?");
63
64         preparedStatement.setString(1, cpfClinte);
65         ResultSet resultSet = preparedStatement.executeQuery();
66
67         //nenhum funcion rio encontrado
68         if (!resultSet.next()) {
69             return null;
70         }
71
72         Cliente c = new Cliente();
73         c.setCPF(resultSet.getString(1));
74         c.setNome(resultSet.getString(2));
75         c.setTelefone(resultSet.getString(3));
76         c.setEndereco(resultSet.getInt(4));
77         c.setEmail(resultSet.getString(5));
78         return c;
79     }
80
81     @Override
82     public void editarCliente(Cliente cliente, String cpf) throws SQLException {
83             if(cliente.getEndereco()!=0) {
84                 PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE␣fazenda_bd.cliente␣SET␣cpf=?,
       nome␣=?,telefone=?,idendereco=?,email=?␣WHERE␣cpf=?;");
85                 preparedStatement.setString(1, cliente.getCpf());
86                 preparedStatement.setString(2, cliente.getNome());
87                 preparedStatement.setString(3, cliente.getTelefone());
88                 preparedStatement.setInt(4, cliente.getEndereco());
89                 preparedStatement.setString(5, cliente.getEmail());
```

```
90                  preparedStatement.setString(6, cpf);
91                  preparedStatement.execute();
92              }else{
93                  PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE fazenda_bd.cliente SET cpf=?,
     nome =?,telefone=?,email=? WHERE cpf=?;");
94                  preparedStatement.setString(1, cliente.getCpf());
95                  preparedStatement.setString(2, cliente.getNome());
96                  preparedStatement.setString(3, cliente.getTelefone());
97                  preparedStatement.setString(4, cliente.getEmail());
98                  preparedStatement.setString(5, cpf);
99                  preparedStatement.execute();
100             }
101
102     }
103
104     /*public static  void main(String Args[]) throws SQLException{
105         //Cliente n = new Cliente("1010",'f',"elias","9999","elias@elias",true,"rua",false);
106         ClienteImp a = new ClienteImp(ConnectionFactory.getConnection());
107         /*Cliente n = new Cliente();
108         n.setCPF("13");
109         n.setEndereco(1);
110         n.setInadiplente(false);
111         n.setAtivo(true);
112         n.setTelefone("jkjn,dn");
113         n.setEmail("elias@elias");
114         n.setNome("elias");
115         a.cadastrarCliente(n);
116         Cliente n = a.buscarCliente("13");
117
118
119     }*/
120 }
```

Código 20: ClienteImp.java

5.2.5 Contrato Imp

```
1  package br.com.afazenda.model.dao;
2
3  import br.com.afazenda.model.interfacedao.ContratoDao;
4  import br.com.afazenda.model.vo.Cargo;
5  import br.com.afazenda.model.vo.Contrato;
6  import br.com.afazenda.model.vo.Funcionario;
7
8  import java.sql.*;
9  import java.util.ArrayList;
10
11 /**
12  * Created by Givaldo Marques
13  * on 04/05/17.
14  */
15 public class ContratoImp implements ContratoDao {
16
17     private Connection conexao;
18
19     public ContratoImp(Connection conexao) {
20         this.conexao = conexao;
21     }
22
23     @Override
24     public void definirContrato(Contrato contrato) throws SQLException {
25
26         String query = "INSERT INTO fazenda_bd.contrato(datainicio, datafim, ativo, " +
27                 "idfuncionario, idcargo) VALUES (?, ?, ?, ?, ?)";
28
29         PreparedStatement preparedStatement = conexao.prepareStatement(query);
30
31         preparedStatement.setString(1, contrato.getDataContratacao());
32         preparedStatement.setString(2, contrato.getDataTermino());
33         preparedStatement.setBoolean(3, contrato.isAtivo());
34         preparedStatement.setString(4, contrato.getFuncionario().getCpf());
35         preparedStatement.setInt(5, contrato.getCargo().getId());
36
37         preparedStatement.execute();
38
39
40     }
41
42     @Override
43     public void inativarContrato(Contrato contrato) throws SQLException {
44         String query = "UPDATE fazenda_bd.contrato SET ativo = FALSE WHERE idfuncionario = ?";
45
46         PreparedStatement preparedStatement = conexao.prepareStatement(query);
47
48         preparedStatement.setString(1, contrato.getFuncionario().getCpf());
49         preparedStatement.execute();
50
51     }
```

```
52
53      @Override
54      public ArrayList<Contrato> listarContratos() throws SQLException {
55          ArrayList<Contrato> contratos = new ArrayList<>();
56
57          String query = "SELECT␣*␣FROM␣fazenda_bd.contrato␣NATURAL␣JOIN␣fazenda_bd.cargo";
58
59          Statement statement = conexao.createStatement();
60          ResultSet resultSet = statement.executeQuery(query);
61
62          while (resultSet.next()) {
63              Cargo cargo = new Cargo();
64              cargo.setId(resultSet.getInt(1));
65              cargo.setNome(resultSet.getString(7));
66
67              Funcionario funcionario = new Funcionario();
68              funcionario.setCpf(resultSet.getString(6));
69
70              Contrato contrato = new Contrato();
71              contrato.setId(resultSet.getInt(2));
72              contrato.setDataContratacao(resultSet.getString(3));
73              contrato.setDataTermino(resultSet.getString(4));
74              contrato.setCargo(cargo);
75              contrato.setFuncionario(funcionario);
76              contrato.setAtivo(resultSet.getBoolean(5));
77
78              contratos.add(contrato);
79
80          }
81
82          return contratos;
83      }
84
85      @SuppressWarnings("Duplicates")
86      @Override
87      public ArrayList<Funcionario> listarContratados() throws SQLException {
88
89          ArrayList<Funcionario> funcionarios = new ArrayList<>();
90
91          String query = "SELECT␣cpf,␣nome,␣telefone,␣email,␣idendereco,␣ativo,␣dtnascimento␣" +
92                  "FROM␣fazenda_bd.contrato␣" +
93                  "NATURAL␣JOIN␣fazenda_bd.funcionario␣WHERE␣ativo;";
94
95          Statement statement = conexao.createStatement();
96          ResultSet resultSet = statement.executeQuery(query);
97
98          while (resultSet.next()) {
99              //preenchendo um objeto funcion rio
100             Funcionario f = new Funcionario();
101             f.setCpf(resultSet.getString(1));
102             f.setName(resultSet.getString(2));
103             f.setTelefone(resultSet.getString(3));
104             f.setEmail(resultSet.getString(4));
105             f.setEndereco(resultSet.getInt(5));
106             f.setAtivo(resultSet.getBoolean(6));
107             f.setNascimento(resultSet.getString(7));
108             funcionarios.add(f);
109         }
110
111         return funcionarios;
112     }
113
114     @Override
115     public ArrayList<Contrato> listarContratosPorCPF(String cpf) throws SQLException {
116
117         ArrayList<Contrato> contratos = new ArrayList<>();
118
119         String query = "SELECT␣*␣FROM␣fazenda_bd.contrato␣NATURAL␣JOIN␣fazenda_bd.cargo␣WHERE␣contrato.idfuncionario␣=␣
     ?";
120
121         //String query = "SELECT * FROM fazenda_bd.contrato NATURAL JOIN fazenda_bd.cargo WHERE idfuncionario = " + cpf
     ;
122
123         PreparedStatement statement = conexao.prepareStatement(query);
124
125         statement.setString(1, cpf);
126         ResultSet resultSet = statement.executeQuery();
127
128         while (resultSet.next()) {
129
130             Cargo cargo = new Cargo();
131             cargo.setId(resultSet.getInt(1));
132             cargo.setNome(resultSet.getString(7));
133
134             Funcionario funcionario = new Funcionario();
135             funcionario.setCpf(resultSet.getString(6));
136
137             Contrato contrato = new Contrato();
138             contrato.setFuncionario(funcionario);
139             contrato.setId(resultSet.getInt(2));
140             contrato.setDataContratacao(resultSet.getString(3));
141             contrato.setDataTermino(resultSet.getString(4));
142             contrato.setAtivo(resultSet.getBoolean(5));
143             contrato.setCargo(cargo);
```

```
144
145             contratos.add(contrato);
146         }
147
148         return contratos;
149     }
150 }
```

Código 21: ContratoImp.java

5.2.6 Endereço Imp

```
1  package br.com.afazenda.model.dao;
2
3  import br.com.afazenda.model.interfacedao.EnderecoDao;
4  import br.com.afazenda.model.vo.Endereco;
5  import utils.ConnectionFactory;
6
7  import java.sql.*;
8
9  /**
10  * Created by gallotropo on 5/2/17.
11  */
12 public class EnderecoImp implements EnderecoDao {
13     private Connection conexao;
14
15     public EnderecoImp (Connection conexao){
16         this.conexao = conexao;
17     }
18     @Override
19     public void setEndereco(Endereco endereco) throws SQLException {
20         PreparedStatement preparedStatement = conexao.prepareStatement("INSERT␣INTO␣fazenda_bd.endereco(logradouro,␣
       numero,␣bairro,␣cidade,␣estado,cep)␣VALUES␣(?,?,?,?,?,?);");
21         preparedStatement.setString(1,endereco.getLogradouro());
22         preparedStatement.setInt(2,endereco.getNumero());
23         preparedStatement.setString(3,endereco.getBairro());
24         preparedStatement.setString(4,endereco.getCidade());
25         preparedStatement.setString(5,endereco.getEstado());
26         preparedStatement.setString(6,endereco.getCep());
27         preparedStatement.execute();
28
29     }
30
31     @Override
32     public Endereco getEndereco(int id_endereco) throws SQLException {
33
34         PreparedStatement preparedStatement = conexao.prepareStatement("SELECT␣*␣FROM␣fazenda_bd.endereco␣WHERE␣
       endereco.idendereco␣=␣␣?;");
35
36         preparedStatement.setInt(1, id_endereco);
37         ResultSet resultSet = preparedStatement.executeQuery();
38         if (!resultSet.next()){
39             return null;
40         }
41         Endereco endereco = new Endereco();
42         endereco.setLogradouro(resultSet.getString(2));
43         endereco.setNumero(resultSet.getInt(3));
44         endereco.setBairro(resultSet.getString(4));
45         endereco.setCidade(resultSet.getString(5));
46         endereco.setEstado(resultSet.getString(6));
47         endereco.setCep(resultSet.getString(7));;
48         return  endereco;
49     }
50
51     @Override
52     //Verifica se o endere o j  existe, se n o existir cadastra novo endere o e retorna o referente idEndere o
53     public int insereEndereco(Endereco endereco) throws SQLException {
54         PreparedStatement preparedStatement = conexao.prepareStatement("select␣fazenda_bd.insere_endereco(plogradouro␣
       =>?,␣pnumero=>?,␣pbairro=>␣?,␣pcidade␣=>␣?,␣pestado␣=>?,␣pcep=>?)");
55         preparedStatement.setString(1,endereco.getLogradouro());
56         preparedStatement.setInt(2,endereco.getNumero());
57         preparedStatement.setString(3,endereco.getBairro());
58         preparedStatement.setString(4,endereco.getCidade());
59         preparedStatement.setString(5,endereco.getEstado());
60         preparedStatement.setString(6,endereco.getCep());
61
62
63
64         ResultSet resultSet = preparedStatement.executeQuery();
65
66         //nenhum funcion rio  encontrado
67         if (!resultSet.next()) {
68             return 0;
69         }
70
71         int idEndereco = resultSet.getInt(1);
72
73         return idEndereco;
```

```
74
75      }
76
77
78  }
```

Código 22: EnderecoImp.java

5.2.7 Especie Imp

```
1   /*
2    * To change this license header, choose License Headers in Project Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package br.com.afazenda.model.dao;
7
8   import br.com.afazenda.model.interfacedao.EspecieDao;
9   import br.com.afazenda.model.vo.Especie;
10
11  import java.sql.*;
12  import java.util.ArrayList;
13
14  /**
15   * @author Elias Rabelo && Patrick Jones
16   */
17  public class EspecieImp implements EspecieDao {
18
19      Connection conexao;
20
21      public EspecieImp(Connection conexao) {
22          this.conexao = conexao;
23      }
24
25      public void insereEspecie(String nome, int quantidade) throws SQLException {
26          String query = "insert␣into␣fazenda_bd.especie(nome,␣quantidade)␣VALUES␣(?,?)";
27
28          PreparedStatement comando = conexao.prepareStatement(query);
29          comando.setString(1, nome);
30          comando.setInt(2, quantidade);
31
32          comando.execute();
33      }
34
35      public void insereProdutoGerado(String nomeProduto, int idEspecie) throws SQLException {
36          Statement comando = conexao.createStatement();
37          comando.executeQuery("INSERT␣INTO␣produtosgerados(\"id_especie\",\"nome\")␣VALUES␣(" + idEspecie + ",'" +
        nomeProduto + "');");
38      }
39
40      public int getIdEspecie(String especieNome) throws SQLException {
41          Statement comando = conexao.createStatement();
42          ResultSet resultado = comando.executeQuery("SELECT␣id_especie␣FROM␣fazenda_bd.especie␣WHERE␣nome='" +
        especieNome + "';");
43          try {
44              resultado.next();
45              return resultado.getInt(1);
46          } catch (Exception ex) {
47              return 0;
48          }
49      }
50
51      public ArrayList<Especie> getEspecies() throws SQLException {
52          ArrayList<Especie> especies = new ArrayList<>();
53
54          String query = "SELECT␣*␣FROM␣fazenda_bd.especie;";
55
56          Statement comando = conexao.createStatement();
57          ResultSet resultado = comando.executeQuery(query);
58
59          while (resultado.next()) {
60
61              Especie especie = new Especie();
62
63              System.out.println(resultado.getString(2));
64
65              especie.setId(resultado.getInt(1));
66              especie.setName(resultado.getString(2));
67              especie.setQuantidade(resultado.getInt(3));
68
69              especies.add(especie);
70          }
71
72          return especies;
73      }
74
75      public void excluirEspecie(Especie especie) throws SQLException {
76
```

```
        String query = "DELETE FROM fazenda_bd.especie WHERE idespecie = ?;";

        PreparedStatement preparedStatement = conexao.prepareStatement(query);
        preparedStatement.setInt(1, especie.getId());
        preparedStatement.execute();

    }


}
```

Código 23: EspecieImp.java

5.2.8 Funcionario Imp

```
package br.com.afazenda.model.dao;

import br.com.afazenda.model.interfacedao.FuncionarioDao;
import br.com.afazenda.model.vo.Funcionario;

import java.sql.*;
import java.util.ArrayList;

/**
 * Created by Givaldo Marques
 * on 25/04/17.
 */
public class FuncionarioImp implements FuncionarioDao {

    private Connection conexao;

    public FuncionarioImp(Connection conexao) throws SQLException {
        this.conexao = conexao;
    }

    @Override
    public void cadastrarFuncionario(Funcionario funcionario) throws SQLException {
        PreparedStatement preparedStatement = conexao.prepareStatement("INSERT INTO fazenda_bd." +
                "funcionario(cpf, nome, telefone, email, idendereco, ativo) VALUES (?, ?, ?, ?, ?, ?)");

        preparedStatement.setString(1, funcionario.getCpf());
        preparedStatement.setString(2, funcionario.getName());
        preparedStatement.setString(3, funcionario.getTelefone());
        preparedStatement.setString(4, funcionario.getEmail());
        preparedStatement.setInt(5, funcionario.getEndereco());
        preparedStatement.setBoolean(6,true);

        preparedStatement.execute();

    }

    @Override
    public Funcionario BuscarFuncionario(String cpfFuncionario) throws SQLException {
        PreparedStatement preparedStatement = conexao.prepareStatement("SELECT *FROM fazenda_bd.funcionario " +
                "WHERE funcionario.cpf = ?");

        preparedStatement.setString(1, cpfFuncionario);
        ResultSet resultSet = preparedStatement.executeQuery();

        //nenhum funcion rio encontrado
        if (!resultSet.next()) {
            return null;
        }

        Funcionario f = new Funcionario();
        f.setCpf(resultSet.getString(1));
        f.setName(resultSet.getString(2));
        f.setTelefone(resultSet.getString(3));
        f.setEmail(resultSet.getString(4));
        f.setEndereco(resultSet.getInt(5));
        f.setAtivo(resultSet.getBoolean(6));

        return f;

    }

    @SuppressWarnings("Duplicates")
    public ArrayList<Funcionario> listarFuncionarios() throws SQLException {

        ArrayList<Funcionario> funcionarios = new ArrayList<>();

        String query = "SELECT *FROM fazenda_bd.funcionario;";

        Statement statement = conexao.createStatement();
        ResultSet resultSet = statement.executeQuery(query);

        while(resultSet.next()){
            Funcionario f = new Funcionario();
```

```
74              f.setCpf(resultSet.getString(1));
75              f.setName(resultSet.getString(2));
76              f.setTelefone(resultSet.getString(3));
77              f.setEmail(resultSet.getString(4));
78              f.setEndereco(resultSet.getInt(5));
79              f.setAtivo(resultSet.getBoolean(6));
80              f.setNascimento(resultSet.getString(7));
81              funcionarios.add(f);
82          }
83          return  funcionarios;
84      }
85
86      @Override
87      public boolean alterarTelefone(Funcionario funcionario, String novoTelefone) throws SQLException {
88
89          PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE␣fazenda_bd.funcionario␣SET␣telefone␣=" +
90                  "␣?␣WHERE␣cpf␣=␣?;");
91
92          preparedStatement.setString(1, novoTelefone);
93          preparedStatement.setString(2, funcionario.getCpf());
94
95          return preparedStatement.execute();
96
97      }
98
99      @Override
100     public boolean alterarEmail(Funcionario funcionario, String novoEmail) throws SQLException {
101         PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE␣fazenda_bd.funcionario␣SET␣email␣=" +
102                 "␣?␣WHERE␣cpf␣=␣?;");
103
104         preparedStatement.setString(1, novoEmail);
105         preparedStatement.setString(2, funcionario.getCpf());
106
107         return preparedStatement.execute();
108
109     }
110
111     @Override
112     public boolean alterarEndereco(Funcionario funcionario, int novoEndereco) throws SQLException {
113
114         PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE␣fazenda_bd.funcionario␣SET␣endereco␣=" +
115                 "␣?␣WHERE␣cpf␣=␣?;");
116
117         preparedStatement.setInt(1, novoEndereco);
118         preparedStatement.setString(2, funcionario.getCpf());
119
120         return preparedStatement.execute();
121
122     }
123
124     @Override
125     public void setAtivo(String cpf) throws SQLException {
126         PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE␣fazenda_bd.funcionario␣SET␣ativo=TRUE␣
      WHERE␣cpf␣=␣?");
127         preparedStatement.setString(1,cpf);
128         preparedStatement.execute();
129     }
130
131     @Override
132     public void setInativo(String cpf) throws SQLException {
133         PreparedStatement preparedStatement = conexao.prepareStatement("UPDATE␣fazenda_bd.funcionario␣SET␣ativo=FALSE␣
      WHERE␣cpf␣=␣?");
134         preparedStatement.setString(1,cpf);
135         preparedStatement.execute();
136     }
137
138
139 }
```

Código 24: FuncionarioImp.java

5.2.9 Produto Imp

```
1  package br.com.afazenda.model.dao;
2
3  import br.com.afazenda.model.interfacedao.ProdutoDao;
4  import br.com.afazenda.model.vo.Produto;
5
6  import java.sql.*;
7  import java.util.ArrayList;
8
9  /**
10  * Created by Givaldo Marques
11  * on 04/05/17.
12  */
13 public class ProdutoImp implements ProdutoDao {
14
15     Connection conexao;
```

```
16
17      public ProdutoImp(Connection conexao) {
18          this.conexao = conexao;
19      }
20
21      @Override
22      public void cadastrarProduto(Produto produto) throws SQLException {
23          String query = "INSERT␣INTO␣fazenda_bd.produto(nome,␣preco,␣origem,␣unidadedemedida,␣" +
24                  "datarecebimento,quantidade)␣VALUES␣(?,␣?,␣?,␣?,␣?,?);";
25
26          PreparedStatement preparedStatement = conexao.prepareStatement(query);
27
28          preparedStatement.setString(1, produto.getNome());
29          preparedStatement.setFloat(2, produto.getPreco());
30          preparedStatement.setString(3, String.valueOf(produto.getOrigem()));
31          preparedStatement.setString(4, produto.getMedida());
32          preparedStatement.setString(5, produto.getData());
33          preparedStatement.setInt(6, produto.getQuantidade());
34
35          preparedStatement.execute();
36
37      }
38
39      @Override
40      public void excluirProduto(Produto produto) throws SQLException {
41
42          String query = "DELETE␣FROM␣fazenda_bd.produto␣WHERE␣idproduto␣=␣?;";
43
44          PreparedStatement preparedStatement = conexao.prepareStatement(query);
45          preparedStatement.setInt(1, produto.getId());
46          preparedStatement.execute();
47
48      }
49
50      @Override
51      public ArrayList<Produto> listarProdutos() throws SQLException {
52          ArrayList<Produto> produtos = new ArrayList<>();
53
54          String query = "SELECT␣*␣FROM␣fazenda_bd.produto;";
55
56          Statement statement = conexao.createStatement();
57          ResultSet resultSet = statement.executeQuery(query);
58
59          while (resultSet.next()) {
60              Produto produto = new Produto();
61              produto.setId(resultSet.getInt(1));
62              produto.setNome(resultSet.getString(2));
63              produto.setPreco(resultSet.getFloat(3));
64              produto.setOrigem(resultSet.getString(4).charAt(0));
65              produto.setMedida(resultSet.getString(5));
66              produto.setData(resultSet.getString(6));
67              produto.setQuantidade(resultSet.getInt(7));
68
69              produtos.add(produto);
70          }
71
72          return produtos;
73      }
74
75      @Override
76      public ArrayList<Produto> buscarProdutoPorNome(String nome) throws SQLException {
77          ArrayList<Produto> produtos = new ArrayList<>();
78
79          String query = "SELECT␣*␣FROM␣fazenda_bd.produto␣WHERE␣lower(nome)␣LIKE␣'%" + nome + "%'";
80
81          Statement statement = conexao.createStatement();
82          ResultSet resultSet = statement.executeQuery(query);
83
84          while (resultSet.next()) {
85              Produto produto = new Produto();
86              produto.setId(resultSet.getInt(1));
87              produto.setNome(resultSet.getString(2));
88              produto.setPreco(resultSet.getFloat(3));
89              produto.setOrigem(resultSet.getString(4).charAt(0));
90              produto.setMedida(resultSet.getString(5));
91              produto.setData(resultSet.getString(6));
92              produto.setQuantidade(resultSet.getInt(7));
93
94              produtos.add(produto);
95          }
96
97          return produtos;
98      }
99
100     @Override
101     public void adionarItem(Produto produto, int quantidade) throws SQLException {
102         String query = "UPDATE␣fazenda_bd.produto␣SET␣quantidade␣=␣quantidade␣+␣?␣WHERE␣idproduto" +
103                 "␣=␣?;";
104
105         PreparedStatement preparedStatement = conexao.prepareStatement(query);
106         preparedStatement.setInt(1, quantidade);
107         preparedStatement.setInt(2, produto.getId());
108         preparedStatement.execute();
109     }
```

```
110
111     @Override
112     public void removerItem(Produto produto, int quantidade) throws SQLException {
113         String query = "UPDATE␣fazenda_bd.produto␣SET␣quantidade␣=␣quantidade␣-␣?␣WHERE␣idproduto" +
114                 "␣=␣?;";
115
116         PreparedStatement preparedStatement = conexao.prepareStatement(query);
117         preparedStatement.setInt(1, quantidade);
118         preparedStatement.setInt(2, produto.getId());
119         preparedStatement.execute();
120
121     }
122 }
```

Código 25: ProdutoImp.java

5.2.10 Venda Imp

```
1   package br.com.afazenda.model.dao;
2
3   import br.com.afazenda.model.interfacedao.VendaDao;
4   import br.com.afazenda.model.vo.ItemProduto;
5   import br.com.afazenda.model.vo.Produto;
6   import br.com.afazenda.model.vo.Venda;
7
8   import java.sql.*;
9   import java.util.ArrayList;
10
11  /**
12   * Created by Givaldo Marques && Alice
13   * on 01/05/17.
14   */
15  public class VendaImp implements VendaDao {
16      Connection conexao;
17
18      public VendaImp(Connection conexao) {
19          this.conexao = conexao;
20      }
21
22      @Override
23      public void insereVenda(Venda venda) throws SQLException {
24          PreparedStatement preparedStatement = conexao.prepareStatement("select␣fazenda_bd.insere_venda(pdatavenda␣=>?,␣
        pvalor=>?,␣pcliente=>␣?,␣pfuncionario␣=>␣?)␣");
25          preparedStatement.setString(1,venda.getDate());
26          preparedStatement.setDouble(2,venda.getValorVenda());
27          preparedStatement.setString(3,venda.getCliente().getCpf());
28          preparedStatement.setString(4,venda.getFuncionario().getCpf());
29
30          ResultSet resultSet = preparedStatement.executeQuery();
31
32          //nenhuma venda inserida
33          if (!resultSet.next()) {
34              return;
35          }
36
37          int idVenda = resultSet.getInt(1);
38
39          for (ItemProduto i : venda.getItens()) {
40              PreparedStatement itensVenda = conexao.prepareStatement(
41                      "INSERT␣INTO␣fazenda_bd.item_venda(idproduto,␣idvenda,␣quantidade)␣VALUES␣?," +
42                              "␣?,␣?");
43
44              itensVenda.setInt(1,idVenda);
45              itensVenda.setInt(2, i.getProduto().getId());
46              itensVenda.setInt(3, i.getQuantidade());
47              preparedStatement.executeQuery();
48          }
49
50
51      }
52
53      @Override
54      public ArrayList<Produto> listarProdutosVendidos() throws SQLException {
55          ArrayList<Produto> produtos = new ArrayList<>();
56
57          String query = "SELECT␣produto.nome␣FROM␣fazenda_bd.venda␣NATURAL␣JOIN␣fazenda_bd" +
58                  ".item_venda␣NATURAL␣JOIN␣fazenda_bd.produto";
59
60          Statement statement = conexao.createStatement();
61          ResultSet resultSet = statement.executeQuery(query);
62
63          while (resultSet.next()) {
64              Produto produto = new Produto();
65              produto.setNome(resultSet.getString(1));
66              produtos.add(produto);
67          }
68          return produtos;
69      }
```

```
70 }
```

Código 26: VendaImp.java

5.3 Interface DAO

5.3.1 Agricultura Dao

```
1  package br.com.afazenda.model.interfacedao;
2
3  import br.com.afazenda.model.vo.CulturaAgricola;
4  import br.com.afazenda.model.vo.Plantacao;
5
6  import java.sql.SQLException;
7  import java.util.ArrayList;
8
9  /**
10  * Created by Givaldo Marques && Patrick Jones
11  * on 01/05/17.
12  */
13 public interface AgriculturaDao {
14
15     void cadastrarCulturaAgricola(CulturaAgricola culturaAgricola) throws SQLException;
16
17     void excluirCulturaAgricola(CulturaAgricola culturaAgricola) throws SQLException;
18
19     void cadastarPlantio(Plantacao plantacao) throws SQLException;
20
21     ArrayList<CulturaAgricola> listarCulturaAgricola() throws SQLException ;
22
23     ArrayList<Plantacao> listarPlantioDeCultura() throws SQLException ;
24
25     void excluirPlantioDeCultura(Plantacao plantacao) throws SQLException;
26
27 }
```

Código 27: AgriculturaDao.java

5.3.2 Animal Dao

```
1  package br.com.afazenda.model.interfacedao;
2
3  import br.com.afazenda.model.vo.Animal;
4
5  import java.sql.SQLException;
6  import java.util.ArrayList;
7
8  /**
9   * Created by Givaldo Marques
10  * on 25/04/17.
11  */
12 public interface AnimalDao {
13
14     void insereAnimal(Animal animal) throws SQLException;
15
16     void removeAnimal(Animal animal) throws SQLException;
17
18     ArrayList<Animal> getAnimais() throws SQLException;
19 }
```

Código 28: AnimalDao.java

5.3.3 Cargo Dao

```
1  package br.com.afazenda.model.interfacedao;
2
3  import br.com.afazenda.model.vo.Cargo;
4
5  import java.sql.SQLException;
6  import java.util.ArrayList;
7
8  /**
9   * Created by Givaldo Marques
10  * on 01/05/17.
11  */
```

```java
public interface CargoDao {

    void cadastrarCargo(Cargo cargo) throws SQLException;

    void excluirCargo(Cargo cargo) throws SQLException;

    ArrayList<Cargo> listarCargos() throws SQLException;
}
```

Código 29: CargoDao.java

5.3.4 Cliente Dao

```java
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Cliente;

import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by givaldo on 25/04/17.
 */
public interface ClienteDao {

    void cadastrarCliente(Cliente cliente) throws SQLException;

    ArrayList<Cliente> buscarClientes() throws SQLException;

    Cliente buscarCliente(String cpfClinte) throws SQLException;

    void editarCliente(Cliente cliente,String cpf) throws SQLException;
}
```

Código 30: ClienteDao.java

5.3.5 Contrato Dao

```java
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Contrato;
import br.com.afazenda.model.vo.Funcionario;

import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by Givaldo Marques
 * on 04/05/17.
 */
public interface ContratoDao {

    void definirContrato(Contrato contrato) throws SQLException;

    void inativarContrato(Contrato contrato) throws SQLException;

    ArrayList<Contrato> listarContratos() throws SQLException;

    ArrayList<Funcionario> listarContratados() throws SQLException;

    ArrayList<Contrato> listarContratosPorCPF(String cpf) throws SQLException;



}
```

Código 31: ContratoDao.java

5.3.6 Endereço Dao

```java
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Endereco;

import java.sql.SQLException;

/**
```

```
 * Created by gallotropo on 4/27/17
 */
public interface EnderecoDao {
    void setEndereco(Endereco endereco) throws SQLException;

    Endereco getEndereco(int id_endereco) throws  SQLException;

    int insereEndereco(Endereco endereco) throws SQLException;

}
```

Código 32: EnderecoDao.java

5.3.7 Especie Dao

```
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Especie;

import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by Givaldo Marques && Patrick Jones
 * on 25/04/17.
 */
public interface EspecieDao {

    void insereEspecie(String nome, int quantidade) throws SQLException;

    void excluirEspecie(Especie especie) throws SQLException;

    void insereProdutoGerado(String nomeProduto, int idEspecie) throws SQLException;

    int getIdEspecie(String especieNome) throws SQLException;

    public ArrayList<Especie> getEspecies() throws SQLException;

    //public ArrayList<String> getProdutosGerados(int id) throws SQLException;


}
```

Código 33: EspecieImp.java

5.3.8 Funcionario Dao

```
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Funcionario;

import java.sql.SQLException;

/**
 * Created by Givaldo Marques
 * on 25/04/17.
 */
public interface FuncionarioDao {

    void cadastrarFuncionario(Funcionario funcionario) throws SQLException;

    /**
     * @param cpfFuncionario cpf do funcion rio a ser autenticado
     * @return retorna null caso n  o seja encontrado e retorna Funcion rio
     * caso seja encontrado
     */
    Funcionario BuscarFuncionario(String cpfFuncionario) throws SQLException;

    boolean alterarTelefone(Funcionario funcionario, String novoTelefone) throws SQLException;

    boolean alterarEmail(Funcionario funcionario, String novoEmail) throws SQLException;

    boolean alterarEndereco(Funcionario funcionario, int novoEndereco) throws SQLException;

    void setAtivo (String cpf) throws SQLException;

    void setInativo (String cpf) throws SQLException;


}
```

Código 34: FuncionarioDao.java

5.3.9 Produto Dao

```java
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Produto;

import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by Givaldo Marques
 * on 04/05/17.
 */
public interface ProdutoDao {

    void cadastrarProduto(Produto produto) throws SQLException;

    void excluirProduto(Produto produto) throws SQLException;

    ArrayList<Produto> listarProdutos() throws SQLException;

    ArrayList<Produto> buscarProdutoPorNome(String nome) throws SQLException;

    void adionarItem(Produto produto, int quantidade) throws SQLException;

    void removerItem(Produto produto, int quantidade) throws SQLException;

}
```

Código 35: ProdutoDao.java

5.3.10 Venda Dao

```java
package br.com.afazenda.model.interfacedao;

import br.com.afazenda.model.vo.Produto;
import br.com.afazenda.model.vo.Venda;

import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by Givaldo Marques
 * on 01/05/17.
 */
public interface VendaDao {

    void insereVenda(Venda venda) throws SQLException;

    ArrayList<Produto> listarProdutosVendidos() throws SQLException;

}
```

Código 36: VendaDao.java

6 Codificação Java/SQL Controles, Fronteiras e Banco de Dados

6.1 Controles

6.1.1 Cadastro Cliente

```
1  package br.com.afazenda.controller;
2
3  import br.com.afazenda.model.dao.ClienteImp;
4  import br.com.afazenda.model.dao.EnderecoImp;
5  import br.com.afazenda.model.vo.Cliente;
6  import br.com.afazenda.model.vo.Endereco;
7  import br.com.afazenda.view.MenuCliente;
8  import singletons.UsuarioAtual;
9  import utils.ConnectionFactory;
10 import utils.RegexMatches;
11
12 import javax.swing.*;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.sql.SQLException;
16
17 /**
18  * Created by Nat  e Elias Rabelo on 27/04/17.
19  */
20 public class CadastroClienteController implements ActionListener {
21
22     private static JFrame frame;
23     private JLabel nomeUsuarioAtualLabel;
24     private JButton configura  esButton;
25     private JButton sairButton;
26     private JTextField NomeTextField;
27     private JComboBox sexoComboBox;
28     private JTextField emailTextField;
29     private JTextField cpfTextField;
30     private JTextField cepTextField;
31     private JTextField logradouroTextField;
32     private JTextField numeroTextField;
33     private JTextField cidadeTextField;
34     private JTextField bairroTextField;
35     private JComboBox estadoComboBox;
36     private JButton confirmarButton;
37     private JButton cancelarButton;
38     private JTextField telefoneTextField;
39     private JPanel MainPainel;
40     private JLabel outputLabel;
41
42
43     public CadastroClienteController() {
44         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
45
46         sairButton.addActionListener(this);
47         confirmarButton.addActionListener(this);
48
49         NomeTextField.setText("maria");
50         emailTextField.setText("maria@gmail.com");
51         cpfTextField.setText("233.322.331-32");
52         cepTextField.setText("38829-200");
53         logradouroTextField.setText("B");
54         numeroTextField.setText("200");
55         cidadeTextField.setText("aracaju");
56         bairroTextField.setText("centro");
57         telefoneTextField.setText("(79)23323322");
58         estadoComboBox.setSelectedIndex(1);
59         sexoComboBox.setSelectedIndex(1);
60
61         sairButton.addActionListener(new ActionListener() {
62             @Override
63             public void actionPerformed(ActionEvent e) {
64                 new UsuarioAtual().logoffFuncionario();
65                 new LoginController().iniciarJanela();
66                 frame.dispose();
67             }
68         });
69         cancelarButton.addActionListener(new ActionListener() {
70             @Override
71             public void actionPerformed(ActionEvent e) {
72                 new MenuCliente().iniciarJanela();
73                 frame.dispose();
74             }
75         });
76     }
77
78
79     public void iniciarJanela() {
80         JFrame frame = new JFrame("Cadastrar␣Cliente");
```

```
81          frame.setContentPane(new CadastroClienteController().MainPainel);
82          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
83          frame.pack();
84          frame.setVisible(true);
85
86      }
87
88      @SuppressWarnings("Duplicates")
89      private boolean validarCampos() {
90
91          outputLabel.setText("<html>");
92
93          if (NomeTextField.getText().trim().isEmpty() ||
94                  emailTextField.getText().trim().isEmpty() ||
95                  cpfTextField.getText().trim().isEmpty() ||
96                  logradouroTextField.getText().trim().isEmpty() ||
97                  cepTextField.getText().trim().isEmpty() ||
98                  cidadeTextField.getText().trim().isEmpty() ||
99                  numeroTextField.getText().trim().isEmpty() ||
100                 telefoneTextField.getText().trim().isEmpty() ||
101                 bairroTextField.getText().trim().isEmpty() ||
102                 sexoComboBox.getSelectedIndex() == 0 ||
103                 estadoComboBox.getSelectedIndex() == 0
104
105                 ) {
106             outputLabel.setText("Todos␣os␣campos␣s  o␣obrigat rios");
107             return false;
108         }
109
110         String nome = NomeTextField.getText();
111         if (!RegexMatches.isNome(nome)) {
112             outputLabel.setText(outputLabel.getText() + "<br>Nome␣inv lido");
113             return false;
114         }
115
116         String email = emailTextField.getText();
117         if (!RegexMatches.isEmail(email)) {
118             outputLabel.setText(outputLabel.getText() + "<br>Email␣inv lido.␣Ex:␣" +
119                     "user@dominio.com");
120             return false;
121         }
122
123         String cpf = cpfTextField.getText();
124         if (!RegexMatches.isCPF(cpf)) {
125             outputLabel.setText(outputLabel.getText() + "<br>CPF␣inv lido.␣Ex:␣" +
126                     "012.331.442-21");
127             return false;
128         }
129
130         String numeroCasa = numeroTextField.getText();
131         if (!RegexMatches.isNumber(numeroCasa)) {
132             outputLabel.setText(outputLabel.getText() + "<br>N mero␣da␣casa␣inv lido");
133             return false;
134         }
135         outputLabel.setText(outputLabel.getText() + "</html>");
136
137
138         return true;
139     }
140
141     private void cadastrarCliente() {
142
143         if (!validarCampos()) return;
144
145         try {
146             Endereco endereco = new Endereco(
147                     logradouroTextField.getText(),
148                     Integer.parseInt(numeroTextField.getText()),
149                     bairroTextField.getText(),
150                     cidadeTextField.getText(),
151                     estadoComboBox.getSelectedItem().toString(),
152                     cepTextField.getText());
153             EnderecoImp NovoEnderecoImp = new EnderecoImp(ConnectionFactory.getConnection());
154             ClienteImp novoClienteIMP = new ClienteImp(ConnectionFactory.getConnection());
155             Cliente novoCliente = new Cliente();
156             novoCliente.setEndereco(NovoEnderecoImp.insereEndereco(endereco));
157             novoCliente.setNome(NomeTextField.getText());
158             novoCliente.setEmail(emailTextField.getText());
159             novoCliente.setTelefone(telefoneTextField.getText());
160             novoCliente.setInadiplente(false);
161             novoCliente.setCPF(cpfTextField.getText());
162
163             novoClienteIMP.cadastrarCliente(novoCliente);
164
165             outputLabel.setText("Cliente␣cadastrado␣com␣sucesso");
166
167         } catch (SQLException e) {
168             if (e.getErrorCode() == 0) {
169                 outputLabel.setText("Este␣CPF␣j  ␣est  ␣cadastrado");
170             } else {
171                 outputLabel.setText("Houve␣um␣erro␣ao␣realizar␣conex o␣com␣o␣banco␣de␣dados");
172             }
173
174         }
```

```
175     }
176
177     @Override
178     public void actionPerformed(ActionEvent e) {
179
180         if (e.getSource() == confirmarButton) {
181             cadastrarCliente();
182
183         } else if (e.getSource() == cancelarButton) {
184
185         } else if (e.getSource() == sairButton) {
186
187         }
188     }
189
190 }
```

Código 37: CadastroClienteController.java

6.1.2 Cadastro Contrato

```
1   package br.com.afazenda.controller;
2
3   import br.com.afazenda.model.dao.CargoImp;
4   import br.com.afazenda.model.dao.ContratoImp;
5   import br.com.afazenda.model.dao.FuncionarioImp;
6   import br.com.afazenda.model.vo.Cargo;
7   import br.com.afazenda.model.vo.Contrato;
8   import br.com.afazenda.model.vo.Funcionario;
9   import br.com.afazenda.view.MenuContrato;
10  import singletons.UsuarioAtual;
11  import utils.ConnectionFactory;
12  import utils.RegexMatches;
13
14  import javax.swing.*;
15  import java.awt.event.ActionEvent;
16  import java.awt.event.ActionListener;
17  import java.sql.SQLException;
18  import java.util.ArrayList;
19
20  /**
21   * Created by patrick && Givaldo Marques
22   * on 03/05/17.
23   */
24  public class CadastroContratoController implements ActionListener {
25      private static JFrame frame;
26      private JPanel MainPanel;
27      private JLabel nomeUsuarioAtualLabel;
28      private JComboBox cargoComboBox;
29      private JTextField dataInicioTextField;
30      private JTextField DataTerminoTextField;
31      private JTextField salarioTextField;
32      private JTextField cpfTextField;
33      private JButton confirmarButton;
34      private JButton voltarButton;
35      private JButton configuracoesButton;
36      private JButton sairButton;
37      private JLabel outputLabel;
38      private ArrayList<Cargo> cargosArrayList;
39
40      //funcionario a ser contratado
41      private Funcionario funcionario;
42
43      public CadastroContratoController() {
44          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
45          confirmarButton.addActionListener(this);
46          voltarButton.addActionListener(this);
47          sairButton.addActionListener(this);
48
49          CargoImp cargoImp = new CargoImp(ConnectionFactory.getConnection());
50          try {
51              cargosArrayList = cargoImp.listarCargos();
52              for (Cargo cargo : cargosArrayList) {
53                  cargoComboBox.addItem(cargo.getNome());
54              }
55          } catch (SQLException e) {
56              e.printStackTrace();
57          }
58
59          sairButton.addActionListener(new ActionListener() {
60              @Override
61              public void actionPerformed(ActionEvent e) {
62                  new UsuarioAtual().logoffFuncionario();
63                  new LoginController().iniciarJanela();
64                  frame.dispose();
65              }
66          });
67      }
```

```
68
69      public void iniciarJanela() {
70          frame = new JFrame("Funcion rios");
71          frame.setContentPane(new CadastroContratoController().MainPanel);
72          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
73          frame.pack();
74          frame.setVisible(true);
75      }
76
77      private boolean buscarFuncionarioContratado() {
78          try {
79              String cpf = cpfTextField.getText();
80              FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
81              funcionario = funcionarioImp.BuscarFuncionario(cpf);
82              if (funcionario == null) {
83                  outputLabel.setText("Nenhum␣funcion rio␣localizado,␣verifique␣se␣o␣CPF␣digitado␣" +
84                          "pertence␣a␣algum␣funcionario");
85                  return false;
86              }
87
88              return true;
89
90          } catch (SQLException e) {
91              e.printStackTrace();
92              return false;
93          }
94
95      }
96
97      private boolean validarCampos() {
98
99          boolean r = true;
100
101         outputLabel.setText("<html>");
102
103         if (dataInicioTextField.getText().trim().isEmpty() ||
104                 DataTerminoTextField.getText().trim().isEmpty() ||
105                 salarioTextField.getText().trim().isEmpty() ||
106                 cpfTextField.getText().trim().isEmpty()) {
107
108             outputLabel.setText("Todos␣os␣campos␣s  o␣obrigat rios");
109             return false;
110         }
111
112         String dataInico = dataInicioTextField.getText();
113         String dataFim = DataTerminoTextField.getText();
114         if (!RegexMatches.isData(dataInico) || !RegexMatches.isData(dataFim)) {
115             outputLabel.setText(outputLabel.getText() + "<br>Data␣inv lida.␣Ex:␣12/02/2010");
116             r = false;
117         } else if (dataInico.compareTo(dataFim) != -1) {
118             outputLabel.setText(outputLabel.getText() + "<br>Data␣de␣in cio␣e␣t rmino␣inv lidas");
119             r = false;
120
121         }
122
123         String cpf = cpfTextField.getText();
124         if (!RegexMatches.isCPF(cpf)) {
125             outputLabel.setText(outputLabel.getText() + "<br>CPF␣inv lido.␣Ex:␣011.303.212-33");
126             r = false;
127         }
128
129         String salario = salarioTextField.getText();
130         if (!RegexMatches.isDecimalNumber(salario)) {
131             outputLabel.setText(outputLabel.getText() + "<br>Sal rio␣inv lido.␣Ex:␣998.23␣");
132             r = false;
133         }
134
135         outputLabel.setText(outputLabel.getText() + "</html>");
136
137         return r;
138     }
139
140
141     private void criarContrato() {
142
143         if (!validarCampos()) return;
144         if (!buscarFuncionarioContratado()) return;
145
146         float salario = Float.parseFloat(salarioTextField.getText());
147         String dataInico = dataInicioTextField.getText();
148         String dataFim = DataTerminoTextField.getText();
149
150         Contrato contrato = new Contrato();
151         contrato.setAtivo(true);
152         contrato.setDataContratacao(dataInico);
153         contrato.setDataTermino(dataFim);
154         contrato.setSalario(salario);
155         contrato.setFuncionario(funcionario);
156         contrato.setCargo(cargosArrayList.get(cargoComboBox.getSelectedIndex()));
157
158         try {
159             ContratoImp contratoImp = new ContratoImp(ConnectionFactory.getConnection());
160             contratoImp.definirContrato(contrato);
161             outputLabel.setText("Contrato␣realizado␣com␣sucesso");
```

```
162         } catch (SQLException e) {
163             outputLabel.setText("Erro ao realizar contrato");
164             e.printStackTrace();
165             System.out.println("CadastrarContrato: " + e.getMessage());
166         }
167 
168 
169     }
170 
171     @Override
172     public void actionPerformed(ActionEvent e) {
173 
174         if (e.getSource() == voltarButton) {
175             new MenuContrato().iniciarJanela();
176             frame.dispose();
177         } else if (e.getSource() == confirmarButton) {
178             criarContrato();
179 
180         }
181     }
182 }
```

Código 38: CadastroContratoController.java

6.1.3 Cadastro Funcionario

```
1  package br.com.afazenda.controller;
2  
3  import br.com.afazenda.model.dao.EnderecoImp;
4  import br.com.afazenda.model.dao.FuncionarioImp;
5  import br.com.afazenda.model.vo.Endereco;
6  import br.com.afazenda.model.vo.Funcionario;
7  import singletons.UsuarioAtual;
8  import utils.ConnectionFactory;
9  import utils.RegexMatches;
10 
11 import javax.swing.*;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import java.sql.SQLException;
15 import java.util.Calendar;
16 
17 /**
18  * Created by Nat  & Givaldo Marques
19  * on 27/04/17.
20  */
21 public class CadastroFuncionarioController implements ActionListener {
22 
23     private static JFrame frame;
24     private JButton configuracoesButton;
25     private JButton sairButton;
26     private JButton confirmarButton;
27     private JButton cancelarButton;
28     private JTextField nomeTextField;
29     private JTextField nascTextField;
30     private JTextField emailTextField;
31     private JTextField cpfTextField;
32     private JTextField cepTextField;
33     private JTextField lagradouroTextField;
34     private JTextField numeroTextField;
35     private JTextField cidadeTextField;
36     private JTextField bairroTextField;
37     private JTextField telefoneTextField;
38     private JComboBox estadoComboBox;
39     private JComboBox sexoComboBox;
40     private JComboBox cargoComboBox;
41     private JPanel MainPainel;
42     private JLabel nomeUsuarioAtualLabel;
43     private JLabel outputLabel;
44     private JPanel MainPanel;
45 
46     public CadastroFuncionarioController() {
47         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
48         sairButton.addActionListener(this);
49         confirmarButton.addActionListener(this);
50         cancelarButton.addActionListener(this);
51 
52     }
53 
54     public void iniciarJanela() {
55         JFrame frame = new JFrame("Cadastrar Funcionario");
56         frame.setContentPane(new CadastroFuncionarioController().MainPainel);
57         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
58         frame.pack();
59         frame.setVisible(true);
60 
61     }
62 
```

```
63      private boolean validarCampos() {
64
65          //necess rio para a quebra de linha
66          outputLabel.setText("<html>");
67
68          boolean r = true;
69
70          //nenhum campo pode estar vazaio
71          if (nomeTextField.getText().trim().isEmpty() ||
72                  nascTextField.getText().trim().isEmpty() ||
73                  emailTextField.getText().trim().isEmpty() ||
74                  cpfTextField.getText().trim().isEmpty() ||
75                  cepTextField.getText().trim().isEmpty() ||
76                  lagradouroTextField.getText().trim().isEmpty() ||
77                  numeroTextField.getText().trim().isEmpty() ||
78                  cidadeTextField.getText().trim().isEmpty() ||
79                  telefoneTextField.getText().trim().isEmpty() ||
80                  bairroTextField.getText().trim().isEmpty() ||
81                  estadoComboBox.getSelectedIndex() == 0 ||
82                  //cargoComboBox.getSelectedIndex() == 0 || nao cadastrado ainda
83                  sexoComboBox.getSelectedIndex() == 0) {
84
85              outputLabel.setText("Todos␣os␣campos␣s  o␣obrigat rios");
86              return false;
87
88          }
89
90          String nome = nomeTextField.getText();
91          if (!RegexMatches.isNome(nome)) {
92              outputLabel.setText(outputLabel.getText() + "<br>Nome␣inv lido");
93          }
94
95          String email = emailTextField.getText();
96          if (!RegexMatches.isEmail(email)) {
97              outputLabel.setText(outputLabel.getText() + "<br>Email␣inv lido.␣Ex:␣" +
98                      "user@dominio.com");
99              r = false;
100         }
101
102         String cpf = cpfTextField.getText();
103         if (!RegexMatches.isCPF(cpf)) {
104             outputLabel.setText(outputLabel.getText() + "<br>CPF␣inv lido.␣Ex:␣" +
105                     "012.331.442-21");
106             r = false;
107         }
108
109         String cep = cepTextField.getText();
110         if (!RegexMatches.isCEP(cep)) {
111             outputLabel.setText(outputLabel.getText() + "<br>CEP␣inv lido.␣Ex:␣" +
112                     "276732-820");
113             r = false;
114         }
115
116         String nasc = nascTextField.getText();
117         if (!RegexMatches.isDataNascimento(nasc)) {
118             outputLabel.setText(outputLabel.getText() + "<br>Data␣inv lida.␣Ex:␣" +
119                     "01/12/1999");
120             r = false;
121         } else {
122             int anoAtual = Calendar.getInstance().get(Calendar.YEAR);
123             int anoEscolhido = Integer.parseInt(nasc.split("/")[2]);
124             int idade = anoAtual - anoEscolhido;
125
126             if (idade < 11) {
127                 outputLabel.setText(outputLabel.getText() + "<br>Ano␣escolhido␣inv lido");
128             }
129         }
130
131         String numeroCasa = numeroTextField.getText();
132         if (!RegexMatches.isNumber(numeroCasa)) {
133             outputLabel.setText(outputLabel.getText() + "<br>N mero␣da␣casa␣inv lido");
134
135         }
136
137         //necess rio para a quebra de linha
138         outputLabel.setText(outputLabel.getText() + "</html>");
139
140         return r;
141     }
142
143     private void cadastrarFuncionario() {
144         if (!validarCampos()) return;
145
146         try {
147
148             Endereco endereco = new Endereco(
149                     lagradouroTextField.getText(),
150                     Integer.parseInt(numeroTextField.getText()),
151                     bairroTextField.getText(),
152                     cidadeTextField.getText(),
153                     estadoComboBox.getSelectedItem().toString(),
154                     cepTextField.getText());
155             EnderecoImp NovoEnderecoImp =  new EnderecoImp(ConnectionFactory.getConnection());
156             FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
```

```
157             Funcionario funcionario = new Funcionario();
158             funcionario.setEndereco(NovoEnderecoImp.insereEndereco(endereco));
159             funcionario.setName(nomeTextField.getText());
160             funcionario.setEmail(emailTextField.getText());
161             funcionario.setTelefone(telefoneTextField.getText());
162             funcionario.setCpf(cpfTextField.getText());
163             funcionario.setNascimento(nascTextField.getText());
164             funcionario.setEndereco(1);
165             funcionario.setAtivo(true);
166
167             funcionarioImp.cadastrarFuncionario(funcionario);
168
169             outputLabel.setText("Funcion rio␣cadastrado␣com␣sucesso");
170
171         } catch (SQLException e) {
172             if (e.getErrorCode() == 0) {
173                 outputLabel.setText("Este␣CPF␣j ␣est ␣cadastrado");
174             } else {
175                 outputLabel.setText("Houve␣um␣erro␣ao␣realizar␣conex o␣com␣o␣banco␣de␣dados");
176             }
177         }
178
179
180     }
181
182     @Override
183     public void actionPerformed(ActionEvent e) {
184
185         if (e.getSource() == confirmarButton) {
186             cadastrarFuncionario();
187
188         } else if (e.getSource() == cancelarButton) {
189             new PesquisarFuncionariosController().iniciarJanela();
190             frame.dispose();
191
192         } else if (e.getSource() == sairButton) {
193             new UsuarioAtual().logoffFuncionario();
194             new LoginController().iniciarJanela();
195             frame.dispose();
196         }
197
198     }
199
200 }
```

Código 39: CadastroFuncionarioController.java

6.1.4 Criar Venda

```
1  package br.com.afazenda.controller;
2
3  import br.com.afazenda.model.dao.ClienteImp;
4  import br.com.afazenda.model.vo.Cliente;
5  import br.com.afazenda.model.vo.ItemProduto;
6  import br.com.afazenda.view.MenuPrincipal;
7  import br.com.afazenda.view.MenuVendas;
8  import singletons.CarrinhoSingleton;
9  import singletons.UsuarioAtual;
10 import utils.ConnectionFactory;
11 import utils.RegexMatches;
12
13 import javax.swing.*;
14 import java.sql.SQLException;
15
16 /**
17  * Created by patrick && Givaldo Marques
18  * on 03/05/17.
19  */
20 public class CriarVendaController {
21     private static JFrame frame;
22     ClienteImp clienteImp = new ClienteImp(ConnectionFactory.getConnection());
23     private JPanel MainPanel;
24     private JLabel nomeUsuarioAtualLabel;
25     private JButton configuracoesButton;
26     private JButton sairButton;
27     private JTextField cpfTextField;
28     private JTextField valorPagoTextField;
29     private JButton cancelarButton;
30     private JButton gerarNotaFiscalButton;
31     private JButton removerItemButton;
32     private JButton adicionarItemButton;
33     private JList<String> carrinhoList;
34     private JLabel precoTotalField;
35     private JLabel outputLabel;
36     private Cliente cliente;
37
38     public CriarVendaController() throws SQLException {
39
```

```
40          preencheLista();
41
42          cancelarButton.addActionListener(actionEvent -> {
43              new MenuVendas().iniciarJanela();
44              frame.dispose();
45          });
46
47          adicionarItemButton.addActionListener(e -> {
48              new ListarProdutosController().iniciarJanela();
49              salvarVenda();
50              nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
51              frame.dispose();
52          });
53
54          cancelarButton.addActionListener(actionEvent -> {
55              new MenuPrincipal().iniciarJanela();
56              frame.dispose();
57          });
58
59          sairButton.addActionListener(e -> {
60              new UsuarioAtual().logoffFuncionario();
61              new LoginController().iniciarJanela();
62              frame.dispose();
63          });
64          removerItemButton.addActionListener(e -> {
65              int selectedPositon = carrinhoList.getSelectedIndex();
66              CarrinhoSingleton.ourInstance.getVenda().getItens().remove(selectedPositon);
67              preencheLista();
68          });
69
70          gerarNotaFiscalButton.addActionListener(e -> {
71              if (validarCampos()) {
72
73                  salvarVenda();
74
75                  new GerarNotaFiscalController().iniciarJanela();
76              }
77          });
78      }
79
80
81      private void salvarVenda() {
82
83          try {
84              float valorPago = Float.parseFloat(valorPagoTextField.getText());
85              float valorVenda = Float.parseFloat(precoTotalField.getText());
86              CarrinhoSingleton.ourInstance.getVenda().setValorPago(valorPago);
87              CarrinhoSingleton.ourInstance.getVenda().setValorVenda(valorVenda);
88              cliente = clienteImp.buscarCliente(cpfTextField.getText());
89              CarrinhoSingleton.ourInstance.getVenda().setCliente(cliente);
90
91          } catch (SQLException | NumberFormatException ignore) {
92
93          }
94      }
95
96      private boolean validarCampos() {
97
98          if (CarrinhoSingleton.ourInstance.getVenda().getItens().size() == 0) {
99              outputLabel.setText("Lista␣vazia");
100             return false;
101         }
102         String valorPago = valorPagoTextField.getText();
103         if (!RegexMatches.isDecimalNumber(valorPago)) {
104             outputLabel.setText("Valor␣pago␣inv lido");
105             return false;
106         }
107
108         float precoTotal = Float.parseFloat(precoTotalField.getText());
109         if (Float.parseFloat(valorPago) < precoTotal) {
110             outputLabel.setText("Valor␣pago␣insuficiente");
111
112         }
113
114         String cpf = cpfTextField.getText();
115         if (!RegexMatches.isCPF(cpf)) {
116             outputLabel.setText("CPF␣inv lido.␣Ex:123.433.443-12");
117             return false;
118         }
119
120         try {
121
122             cliente = clienteImp.buscarCliente(cpf);
123             if (cliente == null) {
124                 outputLabel.setText("CPF␣informado␣n  o␣pertence␣a␣nenhum␣cliente");
125                 return false;
126             }
127
128         } catch (SQLException e1) {
129             e1.printStackTrace();
130         }
131
132
133         return true;
```

```
134     }
135
136     private void preencheLista() {
137         float precoTotal = 0;
138
139         DefaultListModel<String> modelo = new DefaultListModel<>();
140         for (ItemProduto p : CarrinhoSingleton.ourInstance.getVenda().getItens()) {
141             float precoItem = p.getProduto().getPreco() * p.getQuantidade();
142
143             //Arroz 10 x 3 = 30
144             String s = p.getProduto().getNome() + "␣-␣" + p.getProduto().getPreco() + "␣x␣" + p
145                     .getQuantidade() + "␣=␣" + precoItem;
146             modelo.addElement(s);
147
148             precoTotal += precoItem;
149         }
150         carrinhoList.setModel(modelo);
151         precoTotalField.setText(String.valueOf(precoTotal));
152
153         Cliente cliente = CarrinhoSingleton.ourInstance.getVenda().getCliente();
154
155         cpfTextField.setText(cliente.getCpf());
156
157     }
158
159     public void iniciarJanela() {
160         frame = new JFrame("Venda");
161         try {
162             frame.setContentPane(new CriarVendaController().MainPanel);
163         } catch (SQLException e) {
164             e.printStackTrace();
165         }
166         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
167         frame.pack();
168         frame.setVisible(true);
169     }
170
171
172 }
```

Código 40: CriarVendaController.java

6.1.5 Editar Cliente

```
1  package br.com.afazenda.controller;
2
3  import br.com.afazenda.model.dao.ClienteImp;
4  import br.com.afazenda.model.dao.EnderecoImp;
5  import br.com.afazenda.model.vo.Cliente;
6  import br.com.afazenda.model.vo.Endereco;
7  import br.com.afazenda.view.Configura  esDeUsuario;
8  import singletons.UsuarioAtual;
9  import utils.ConnectionFactory;
10 import utils.RegexMatches;
11
12 import javax.swing.*;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.sql.SQLException;
16
17 /**
18  * Created by gallotropo on 5/3/17.
19  */
20 @SuppressWarnings("Duplicates")
21 public class EditClienteController implements ActionListener {
22
23     private JButton configura  esButton;
24     private JButton sairButton;
25     private JTextField NomeTextField;
26     private JComboBox sexoComboBox;
27     private JTextField emailTextField;
28     private JTextField cpfTextField;
29     private JTextField cepTextField;
30     private JTextField logradouroTextField;
31     private JTextField numeroTextField;
32     private JTextField cidadeTextField;
33     private JTextField bairroTextField;
34     private JComboBox estadoComboBox;
35     private JButton salvarButton;
36     private JButton cancelarButton;
37     private JTextField telefoneTextField;
38     private JPanel MainPainel;
39     private JLabel outputLabel;
40     private String cpf;
41     private Endereco endereco;
42     private static JFrame frame;
43     private JLabel nomeUsuarioAtualLabel;
44
```

```
45      public EditClienteController(){
46          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
47          configura  esButton.addActionListener(this);
48          sairButton.addActionListener(this);
49          salvarButton.addActionListener(this);
50          cancelarButton.addActionListener(this);
51
52          configura  esButton.addActionListener(new ActionListener() {
53              @Override
54              public void actionPerformed(ActionEvent e) {
55                  new Configura  esDeUsuario().iniciarJanela();
56                  frame.dispose();
57              }
58          });
59
60          sairButton.addActionListener(new ActionListener() {
61              @Override
62              public void actionPerformed(ActionEvent e) {
63                  new UsuarioAtual().logoffFuncionario();
64                  new LoginController().iniciarJanela();
65                  frame.dispose();
66              }
67          });
68      }
69
70      public EditClienteController(String cpf) throws SQLException{
71          configura  esButton.addActionListener(this);
72          sairButton.addActionListener(this);
73          salvarButton.addActionListener(this);
74          cancelarButton.addActionListener(this);
75          ClienteImp clienteImp = new ClienteImp(ConnectionFactory.getConnection());
76          Cliente cliente = clienteImp.buscarCliente(cpf);
77          EnderecoImp enderecoImp = new EnderecoImp(ConnectionFactory.getConnection());
78          endereco = enderecoImp.getEndereco(cliente.getEndereco());
79          cpfTextField.setText(cpf);
80          System.out.println(cliente.getNome());
81
82          NomeTextField.setText(cliente.getNome());
83          emailTextField.setText(cliente.getEmail());
84          telefoneTextField.setText(cliente.getTelefone());
85          cepTextField.setText(endereco.getCep());
86          logradouroTextField.setText(endereco.getLogradouro());
87          numeroTextField.setText(Integer.toString(endereco.getNumero()));
88          cidadeTextField.setText(endereco.getCidade());
89          bairroTextField.setText(endereco.getBairro());
90
91      }
92      public void iniciarJanela() {
93          JFrame frame = new JFrame("Editar␣Cliente");
94          frame.setContentPane(new EditClienteController().MainPainel);
95          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
96          frame.pack();
97          frame.setVisible(true);
98
99      }
100     public void iniciarJanela(String cpf) throws SQLException {
101         JFrame frame = new JFrame("Editar␣Cliente");
102         frame.setContentPane(new EditClienteController(cpf).MainPainel);
103         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
104         frame.pack();
105         frame.setVisible(true);
106
107     }
108
109     private boolean validarCampos(){
110
111         outputLabel.setText("<html>");
112
113         if(NomeTextField.getText().trim().isEmpty() ||
114                 emailTextField.getText().trim().isEmpty() ||
115                 cpfTextField.getText().trim().isEmpty() ||
116                 logradouroTextField.getText().trim().isEmpty() ||
117                 cepTextField.getText().trim().isEmpty() ||
118                 cidadeTextField.getText().trim().isEmpty() ||
119                 numeroTextField.getText().trim().isEmpty() ||
120                 telefoneTextField.getText().trim().isEmpty() ||
121                 bairroTextField.getText().trim().isEmpty() ||
122                 sexoComboBox.getSelectedIndex()== 0||
123                 estadoComboBox.getSelectedIndex()==0
124
125                 ){
126             outputLabel.setText("Todos␣os␣campos␣s  o␣obrigat rios");
127             return  false;
128         }
129
130         String nome = NomeTextField.getText();
131         if (!RegexMatches.isNome(nome)) {
132             outputLabel.setText(outputLabel.getText() + "<br>Nome␣inv lido");
133             return false;
134         }
135
136         String email = emailTextField.getText();
137         if (!RegexMatches.isEmail(email)) {
138             outputLabel.setText(outputLabel.getText() + "<br>Email␣inv lido.␣Ex:␣" +
```

```
                    "user@dominio.com");
            return  false;
        }

        String cpf = cpfTextField.getText();
        if (!RegexMatches.isCPF(cpf)) {
            outputLabel.setText(outputLabel.getText() + "<br>CPF inv lido. Ex: " +
                    "012.331.442-21");
            return false;
        }

        String numeroCasa = numeroTextField.getText();
        if (!RegexMatches.isNumber(numeroCasa)) {
            outputLabel.setText(outputLabel.getText() + "<br>N mero da casa inv lido");
            return  false;
        }
        outputLabel.setText(outputLabel.getText() + "</html>");


        return true;
    }

    private void editarCliente(){

        if(!validarCampos()) return;

        try{
            Endereco endereco = new Endereco(
                    logradouroTextField.getText(),
                    Integer.parseInt(numeroTextField.getText()),
                    bairroTextField.getText(),
                    cidadeTextField.getText(),
                    estadoComboBox.getSelectedItem().toString(),
                    cepTextField.getText());
            EnderecoImp NovoEnderecoImp =  new EnderecoImp(ConnectionFactory.getConnection());

            ClienteImp novoClienteIMP = new ClienteImp(ConnectionFactory.getConnection());
            Cliente novoCliente = new Cliente();
            if(!NovoEnderecoImp.equals(this.endereco))  novoCliente.setEndereco(NovoEnderecoImp.insereEndereco(endereco
     ));
            else novoCliente.setEndereco(0);
            novoCliente.setNome(NomeTextField.getText());
            novoCliente.setEmail(emailTextField.getText());
            novoCliente.setTelefone(telefoneTextField.getText());
            novoCliente.setInadiplente(false);
            novoCliente.setCPF(cpfTextField.getText());


            novoClienteIMP.editarCliente(novoCliente,this.cpf);

            outputLabel.setText("Cliente editado com sucesso");

        } catch (SQLException e) {
            if (e.getErrorCode() == 0) {
                outputLabel.setText("Este CPF j  est cadastrado");
            } else {
                outputLabel.setText("Houve um erro ao realizar conex o com o banco de dados");
            }
            e.printStackTrace();

        }
    }
    @Override
    public void actionPerformed(ActionEvent e) {
            if(e.getSource()==salvarButton){
                editarCliente();
            }
    }

}
```

Código 41: EditClienteController.java

6.1.6 Gerar Nota Fiscal

```
package br.com.afazenda.controller;

import br.com.afazenda.model.dao.VendaImp;
import br.com.afazenda.model.vo.ItemProduto;
import singletons.CarrinhoSingleton;
import singletons.UsuarioAtual;
import utils.ConnectionFactory;

import javax.swing.*;
import java.sql.SQLException;

/**
 * Created by patrick on 03/05/17.
```

```
 */
public class GerarNotaFiscalController {
    private static JFrame frame;
    private JPanel MainPanel;
    private JLabel nomeUsuarioAtualLabel;
    private JButton sairButton;
    private JList<String> listaProdutosComprados;
    private JButton voltarButton;
    private JButton confirmarButton;
    private JLabel valorTotalLabel;
    private JLabel valorPagoLabel;
    private JLabel trocoLabel;
    private JLabel vendedorLabel;
    private JLabel cpfClienteLabel;

    public GerarNotaFiscalController() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());

        vendedorLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        CarrinhoSingleton.ourInstance.getVenda().setFuncionario(UsuarioAtual.getInstance().getFuncionarioAtual());
        cpfClienteLabel.setText(CarrinhoSingleton.ourInstance.getVenda().getCliente().getCpf());

        valorPagoLabel.setText(CarrinhoSingleton.ourInstance.getVenda().getValorPago() + "");
        valorTotalLabel.setText(CarrinhoSingleton.ourInstance.getVenda().getValorVenda() + "");
        trocoLabel.setText(CarrinhoSingleton.ourInstance.getVenda().getTroco() + "");

        preencheLista();

        voltarButton.addActionListener(e -> {
            try {
                new CriarVendaController().iniciarJanela();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
            frame.dispose();
        });

        confirmarButton.addActionListener(e -> {
            VendaImp vendaImp = new VendaImp(ConnectionFactory.getConnection());
            try {
                vendaImp.insereVenda(CarrinhoSingleton.ourInstance.getVenda());
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        });
    }

    private void preencheLista() {
        float precoTotal = 0;

        DefaultListModel<String> modelo = new DefaultListModel<>();
        for (ItemProduto p : CarrinhoSingleton.ourInstance.getVenda().getItens()) {
            float precoItem = p.getProduto().getPreco() * p.getQuantidade();

            //Arroz 10 x 3 = 30
            String s = p.getProduto().getNome() + " - " + p.getProduto().getPreco() + " x " + p
                    .getQuantidade() + " = " + precoItem;
            modelo.addElement(s);

            precoTotal += precoItem;
        }

        listaProdutosComprados.setModel(modelo);

    }

    public void iniciarJanela() {
        frame = new JFrame("Venda");
        frame.setContentPane(new GerarNotaFiscalController().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }
}
```

Código 42: GerarNotaFiscalController.java

6.1.7 Gerar Relatório de Vendas

```
package br.com.afazenda.controller;

import br.com.afazenda.model.dao.VendaImp;
import br.com.afazenda.model.vo.Produto;
import br.com.afazenda.view.MenuVendas;
import singletons.UsuarioAtual;
import utils.ConnectionFactory;

```

```java
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by Marina && Givaldo Marques
 * on 04/05/2017.
 */
public class GerarRelatorioVendasController {
    private static JFrame frame;
    private JPanel MainPanel;
    private JLabel nomeUsuarioAtualLabel;
    private JButton configura esButton;
    private JButton sairButton;
    private JList<String> list1;
    private JButton voltarButton;
    private JLabel outputLabel;

    public GerarRelatorioVendasController() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        outputLabel.setText("");
        voltarButton.addActionListener(actionEvent -> {
            new MenuVendas().iniciarJanela();
            frame.dispose();
        });

        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });

        list1.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                super.componentResized(e);
                preencheLista();
            }
        });

    }

    public void iniciarJanela() {
        frame = new JFrame("Vendas");
        frame.setContentPane(new GerarRelatorioVendasController().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }

    private void preencheLista() {
        try {
            VendaImp vendaImp = new VendaImp(ConnectionFactory.getConnection());
            ArrayList<Produto> produtos = vendaImp.listarProdutosVendidos();

            DefaultListModel<String> modelo = new DefaultListModel<>();
            int d = produtos.size();
            Produto produto;
            for (Produto p : produtos) {
                produto = p;
                modelo.addElement(produto.getNome());
            }
            list1.setModel(modelo);
        } catch (SQLException x) {
            x.printStackTrace();
        }
    }

}
```

Código 43: GerarRelatorioVendasController.java

6.1.8 Gerenciar Animais

```java
package br.com.afazenda.controller;

import br.com.afazenda.model.dao.AnimalImp;
import br.com.afazenda.model.dao.EspecieImp;
import br.com.afazenda.model.vo.Animal;
import br.com.afazenda.model.vo.Especie;
```

```
7  import br.com.afazenda.view.MenuAnimais;
8  import singletons.UsuarioAtual;
9  import utils.ConnectionFactory;
10
11 import javax.swing.*;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import java.sql.SQLException;
15 import java.util.ArrayList;
16
17 /**
18  * Created by patrick on 02/05/17.
19  */
20 public class GerenciarAnimaisController {
21     private static JFrame frame;
22     private JPanel MainPanel;
23     private JButton configura  esButton;
24     private JButton sairButton;
25     private JButton adicionarButton;
26     private JList list1;
27     private JButton deletarButton1;
28     private JButton voltarButton;
29     private JComboBox especieComboBox;
30     private JLabel nomeUsuarioAtualLabel;
31     private JLabel outputLabel;
32     private ArrayList<Especie> especies;
33     private ArrayList<Animal> animais;
34     private EspecieImp especieImp = new EspecieImp(ConnectionFactory.getConnection());
35     private AnimalImp animalImp = new AnimalImp(ConnectionFactory.getConnection());
36
37     public GerenciarAnimaisController() {
38         preencherCombo();
39         initLista();
40
41         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
42
43         voltarButton.addActionListener(actionEvent -> {
44             new MenuAnimais().iniciarJanela();
45             frame.dispose();
46         });
47         sairButton.addActionListener(actionEvent -> {
48             new UsuarioAtual().logoffFuncionario();
49             new LoginController().iniciarJanela();
50             frame.dispose();
51         });
52
53         adicionarButton.addActionListener(new ActionListener() {
54             @Override
55             public void actionPerformed(ActionEvent actionEvent) {
56                 cadastrarAnimal();
57                 initLista();
58             }
59         });
60         deletarButton1.addActionListener(new ActionListener() {
61             @Override
62             public void actionPerformed(ActionEvent actionEvent) {
63                 removerAnimal();
64                 initLista();
65             }
66         });
67     }
68
69
70     public void iniciarJanela() throws SQLException {
71         frame = new JFrame("Animal");
72         frame.setContentPane(new GerenciarAnimaisController().MainPanel);
73         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
74         frame.pack();
75         frame.setVisible(true);
76     }
77
78     private void cadastrarAnimal() {
79         Animal animalAuxiliar = new Animal();
80         Especie especieAuxiliar = new Especie();
81         especieAuxiliar = especies.get(especieComboBox.getSelectedIndex());
82         animalAuxiliar.setEspecie(especieAuxiliar);
83
84         try {
85             animalImp.insereAnimal(animalAuxiliar);
86             outputLabel.setText("Animal␣cadastrado␣com␣sucesso!");
87         } catch (SQLException e) {
88             e.printStackTrace();
89         }
90
91     }
92
93     private void removerAnimal() {
94         try {
95             animalImp.removeAnimal(animais.get(list1.getSelectedIndex()));
96             outputLabel.setText("Animal␣removido␣com␣sucesso!");
97         } catch (SQLException e) {
98             e.printStackTrace();
99         }
100    }
```

```
101
102     private void preencherCombo() {
103         try {
104             especies = especieImp.getEspecies();
105             DefaultComboBoxModel listModel = new DefaultComboBoxModel();
106             // listModel.addElement(" Nome da Cultura | Periodo Plantio | Periodo Colheita "  );
107             for (Especie percorrer : especies) {
108                 listModel.addElement(percorrer.getName());
109             }
110             especieComboBox.setModel(listModel);
111         } catch (SQLException e) {
112             e.printStackTrace();
113         }
114     }
115
116     private void initLista() {
117         try {
118             animais = animalImp.getAnimais();
119             DefaultListModel listModel = new DefaultListModel();
120             // listModel.addElement(" Nome da Cultura | Periodo Plantio | Periodo Colheita "  );
121             for (Animal percorrer : animais) {
122                 listModel.addElement("␣ID␣do␣Animal:␣" + percorrer.getEspecie().getName().toUpperCase() + percorrer.
     getId());
123             }
124             list1.setModel(listModel);
125         } catch (SQLException e) {
126             e.printStackTrace();
127         }
128     }
129
130 }
```

Código 44: GerenciarAnimaisController.java

6.1.9 Listar Produtos

```
1  package br.com.afazenda.controller;
2
3  import br.com.afazenda.model.dao.ProdutoImp;
4  import br.com.afazenda.model.vo.ItemProduto;
5  import br.com.afazenda.model.vo.Produto;
6  import singletons.CarrinhoSingleton;
7  import singletons.UsuarioAtual;
8  import utils.ConnectionFactory;
9  import utils.RegexMatches;
10
11 import javax.swing.*;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14
15 /**
16  * Created by patrick on
17  * 03/05/17.
18  */
19 public class ListarProdutosController {
20     private static JFrame frame;
21     private JPanel MainPanel;
22     private JLabel nomeUsuarioAtualLabel;
23     private JButton configuracoesButton;
24     private JButton sairButton;
25     private JTextField produtoTextField;
26     private JButton buscarButton;
27     private JList<String> produtosJlist;
28     private JTextField quantidadeTextField;
29     private JButton voltarButton;
30     private JButton adicionarItemButton;
31     private JButton listarTodos;
32     private JLabel outputLabel;
33     private ProdutoImp produtoImp = new ProdutoImp(ConnectionFactory.getConnection());
34     private ArrayList<Produto> produtoArrayList;
35
36     public ListarProdutosController() {
37
38         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
39
40         sairButton.addActionListener(actionEvent -> {
41             new UsuarioAtual().logoffFuncionario();
42             new LoginController().iniciarJanela();
43             frame.dispose();
44         });
45
46         listarTodos.addActionListener(e -> {
47             try {
48                 produtoArrayList = produtoImp.listarProdutos();
49                 preencheLista();
50
51             } catch (SQLException e1) {
52                 e1.printStackTrace();
```

```
53              }
54          });
55
56          buscarButton.addActionListener(e -> {
57              outputLabel.setText("");
58
59              String nomeProduto = produtoTextField.getText();
60
61              try {
62                  produtoArrayList = produtoImp.listarProdutos();
63                  ArrayList<Produto> aux = new ArrayList<>();
64
65                  for (Produto p : produtoArrayList) {
66                      if (p.getNome().toLowerCase().contains(nomeProduto.toLowerCase())) {
67                          aux.add(p);
68                      }
69                  }
70
71                  produtoArrayList = aux;
72                  preencheLista();
73
74              } catch (SQLException e1) {
75                  e1.printStackTrace();
76              }
77
78          });
79
80          adicionarItemButton.addActionListener(e -> {
81              String quantidade = quantidadeTextField.getText();
82              if (RegexMatches.isNumber(quantidade)) {
83
84                  Produto p = produtoArrayList.get(produtosJlist.getSelectedIndex());
85
86                  if (Integer.parseInt(quantidade) > p.getQuantidade()) {
87                      outputLabel.setText("Quantidade de itens indispon veis");
88                      return;
89                  }
90
91                  ItemProduto itemProduto = new ItemProduto();
92                  itemProduto.setProduto(p);
93                  itemProduto.setQuantidade(Integer.parseInt(quantidade));
94
95                  CarrinhoSingleton.ourInstance.getVenda().getItens().add(itemProduto);
96                  outputLabel.setText("Produto adicionado");
97                  quantidadeTextField.setText("");
98              } else {
99                  outputLabel.setText("Quantidade inv lida");
100             }
101
102             preencheLista();
103         });
104
105         voltarButton.addActionListener(e -> {
106             try {
107                 new CriarVendaController().iniciarJanela();
108             } catch (SQLException e1) {
109                 e1.printStackTrace();
110             }
111             frame.dispose();
112         });
113
114         try {
115             produtoArrayList = produtoImp.listarProdutos();
116             preencheLista();
117         } catch (SQLException e) {
118             e.printStackTrace();
119         }
120
121     }
122
123     private void preencheLista() {
124         boolean possuiNoCarrinho = false;
125         DefaultListModel<String> modelo = new DefaultListModel<>();
126         for (Produto p : produtoArrayList) {
127
128             for (ItemProduto i : CarrinhoSingleton.ourInstance.getVenda().getItens()) {
129                 if (i.getProduto().getId() == p.getId()) {
130                     possuiNoCarrinho = true;
131                 }
132             }
133
134             if (!possuiNoCarrinho) {
135                 modelo.addElement(p.getNome() + ": " + p.getQuantidade() + " itens restantes");
136             }
137
138             possuiNoCarrinho = false;
139         }
140
141         produtosJlist.setModel(modelo);
142
143     }
144
145     public void iniciarJanela() {
146         frame = new JFrame("Venda");
```

```
147         frame.setContentPane(new ListarProdutosController().MainPanel);
148         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
149         frame.pack();
150         frame.setVisible(true);
151         nomeUsuarioAtualLabel.setText(new UsuarioAtual().getFuncionarioAtual().getName());
152     }
153 }
```

Código 45: ListarProdutosController.java

6.1.10 Login

```
1  package br.com.afazenda.controller;
2
3  import br.com.afazenda.model.dao.FuncionarioImp;
4  import br.com.afazenda.model.vo.Funcionario;
5  import br.com.afazenda.view.MenuPrincipal;
6  import singletons.UsuarioAtual;
7  import utils.ConnectionFactory;
8  import utils.RegexMatches;
9
10 import javax.swing.*;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import java.sql.SQLException;
14
15 /**
16  * Created by Nat
17  * on 27/04/17.
18  */
19 public class LoginController implements ActionListener {
20     private JTextField cpfField;
21     private JButton entrarButton;
22     private JPanel MainPanel;
23     private JLabel infoLabel;
24     private static JFrame frame;
25
26     public LoginController() {
27         entrarButton.addActionListener(this);
28
29     }
30
31     public void iniciarJanela() {
32         frame = new JFrame("Login de funcion rio");
33         frame.setContentPane(new LoginController().MainPanel);
34         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
35         frame.pack();
36         frame.setVisible(true);
37
38         cpfField.setText("");
39     }
40
41     private boolean validaCampos() {
42
43         String cpf = cpfField.getText();
44
45         if (cpf.equals("")) {
46             infoLabel.setText("Informe seu CPF");
47             return false;
48         }
49
50         if (!RegexMatches.isCPF(cpf)) {
51             infoLabel.setText("Formato de CPF inv lido. Ex: 123.456.789-10");
52             return false;
53         }
54
55         return true;
56     }
57
58     private void fazerLogin() {
59         FuncionarioImp funcionarioImp;
60
61         if (!validaCampos()) return;
62
63         try {
64             funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
65             Funcionario funcionario = funcionarioImp.BuscarFuncionario(cpfField.getText());
66
67
68             if (funcionario == null) {
69                 infoLabel.setText("CPF n o encontrado");
70             }else{
71
72             infoLabel.setText("Login realizado com sucesso");
73
74             //salva o funcionario logado em um singleton
75             UsuarioAtual.getInstance().setFuncionarioAtual(funcionario);
76
```

```
77              new MenuPrincipal().iniciarJanela();
78              frame.dispose();
79              }
80
81          } catch (SQLException e1) {
82              infoLabel.setText("Houve um erro ao realizar conex o com o banco de dados");
83              e1.printStackTrace();
84          }
85
86      }
87
88      @Override
89      public void actionPerformed(ActionEvent e) {
90          if (e.getSource() == entrarButton) {
91              fazerLogin();
92          }
93      }
94  }
```

Código 46: LoginController.java

6.1.11 Pesquisar Cargos

```
1   package br.com.afazenda.controller;
2
3   import br.com.afazenda.model.dao.CargoImp;
4   import br.com.afazenda.model.vo.Cargo;
5   import br.com.afazenda.view.MenuFuncionarios;
6   import singletons.UsuarioAtual;
7   import utils.ConnectionFactory;
8   import utils.RegexMatches;
9
10  import javax.swing.*;
11  import java.sql.SQLException;
12  import java.util.ArrayList;
13
14  /**
15   * Created by patrick && Givaldo Marques
16   * on 02/05/17.
17   */
18  public class PesquisarCargosController {
19      private static JFrame frame;
20      private JButton configura  esButton;
21      private JButton sairButton;
22      private JTextField cargoTextField;
23      private JButton adicionarButton;
24      private JList list1;
25      private JButton apagarButton;
26      private JButton voltarButton;
27      private JPanel MainPanel;
28      private JLabel nomeUsuarioAtualLabel;
29      private JButton listarTodosButton;
30      private JLabel outputLabel;
31      private ArrayList<Cargo> cargoArrayList;
32      private CargoImp cargoImp = new CargoImp(ConnectionFactory.getConnection());
33
34      public PesquisarCargosController() {
35
36          try {
37              cargoArrayList = cargoImp.listarCargos();
38          } catch (SQLException e) {
39              e.printStackTrace();
40          }
41
42          preencheLista();
43
44          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
45          voltarButton.addActionListener(actionEvent -> {
46              new MenuFuncionarios().iniciarJanela();
47              frame.dispose();
48          });
49
50          sairButton.addActionListener(e -> {
51              new UsuarioAtual().logoffFuncionario();
52              new LoginController().iniciarJanela();
53              frame.dispose();
54          });
55
56          adicionarButton.addActionListener(e -> {
57              try {
58                  String nomeCargo = cargoTextField.getText();
59                  if (RegexMatches.isNome(nomeCargo)) {
60                      Cargo cargo = new Cargo();
61                      cargo.setNome(nomeCargo);
62                      cargoImp.cadastrarCargo(cargo);
63                      cargoArrayList = cargoImp.listarCargos();
64                      preencheLista();
65                  } else {
```

```
66                      outputLabel.setText("Cargo inv lido");
67                  }
68              } catch (SQLException e1) {
69                  outputLabel.setText("Cargo j  cadastrado");
70              }
71
72          });
73          apagarButton.addActionListener(e -> {
74
75              try {
76                  Cargo cargo = cargoArrayList.get(list1.getSelectedIndex());
77                  cargoImp.excluirCargo(cargo);
78                  cargoArrayList = cargoImp.listarCargos();
79                  preencheLista();
80                  outputLabel.setText("");
81
82              } catch (SQLException e1) {
83                  outputLabel.setText("Erro ao excluir cargo");
84
85              }
86          });
87      }
88
89      public static void main(String[] args) {
90          new PesquisarCargosController().iniciarJanela();
91      }
92
93      private void preencheLista() {
94          DefaultListModel<String> modelo = new DefaultListModel<>();
95          for (Cargo c : cargoArrayList) {
96              modelo.addElement(c.getNome());
97          }
98
99          list1.setModel(modelo);
100
101     }
102
103     public void iniciarJanela() {
104         frame = new JFrame("Funcion rios");
105         frame.setContentPane(new PesquisarCargosController().MainPanel);
106         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
107         frame.pack();
108         frame.setVisible(true);
109     }
110 }
```

Código 47: PesquisarCargosController.java

6.1.12 Pesquisar Cliente

```
1  package br.com.afazenda.controller;
2
3  import br.com.afazenda.model.dao.ClienteImp;
4  import br.com.afazenda.model.vo.Cliente;
5  import br.com.afazenda.view.MenuCliente;
6  import singletons.UsuarioAtual;
7  import utils.ConnectionFactory;
8  import utils.RegexMatches;
9
10 import javax.swing.*;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import java.sql.SQLException;
14 import java.util.ArrayList;
15
16 /**
17  * Created by soulplan e Elias Rabelo on 27/04/17.
18  */
19 public class PesquisarClientesController implements ActionListener {
20
21     private static JFrame frame;
22     private JButton configura  esButton;
23     private JButton sairButton;
24     private JTextField cpfTextField;
25     private JButton buscarButton;
26     private JButton listarTodosClientesButton;
27     private JList list1;
28     private JPanel MainPanel;
29     private JButton editarButton;
30     private JButton voltarButton;
31     private JButton cadastrarButton;
32     private JLabel nomeUsuarioAtualLabel;
33
34     public PesquisarClientesController() {
35         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
36         buscarButton.addActionListener(this);
37         listarTodosClientesButton.addActionListener(this);
38         editarButton.addActionListener(this);
```

```
39          cadastrarButton.addActionListener(this);
40
41
42          sairButton.addActionListener(new ActionListener() {
43              @Override
44              public void actionPerformed(ActionEvent e) {
45                  new UsuarioAtual().logoffFuncionario();
46                  new LoginController().iniciarJanela();
47                  frame.dispose();
48              }
49          });
50          voltarButton.addActionListener(new ActionListener() {
51              @Override
52              public void actionPerformed(ActionEvent e) {
53                  new MenuCliente().iniciarJanela();
54                  frame.dispose();
55              }
56          });
57      }
58
59      public void iniciarJanela() {
60          frame = new JFrame("PesquisarClientesController");
61          frame.setContentPane(new PesquisarClientesController().MainPanel);
62          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
63          frame.pack();
64          frame.setVisible(true);
65      }
66
67      private void buscarCliente(String cpf) throws SQLException{
68          ClienteImp clienteImp = new ClienteImp(ConnectionFactory.getConnection());
69          if (!RegexMatches.isCPF(cpf)) {
70              return;
71          }
72          Cliente cliente = clienteImp.buscarCliente(cpf);
73          DefaultListModel modelo = new DefaultListModel();
74          modelo.addElement(cliente.getNome()+"␣-␣"+ cliente.getCpf());
75          list1.setModel(modelo);
76      }
77
78      private  void listarTodosClientes() throws SQLException {
79          ClienteImp clienteImp = new ClienteImp(ConnectionFactory.getConnection());
80          ArrayList<Cliente> clientes = clienteImp.buscarClientes();
81          DefaultListModel modelo = new DefaultListModel();
82          int d =clientes.size();
83          Cliente cliente;
84          for (int i=0; i<d;i++){
85              cliente = clientes.get(i);
86              modelo.addElement(cliente.getNome()+"␣-␣"+ cliente.getCpf());
87          }
88          list1.setModel(modelo);
89      }
90
91      private  void cadastrarCliente(){
92          CadastroClienteController c = new CadastroClienteController();
93          c.iniciarJanela();
94          frame.dispose();
95      }
96
97      @Override
98      public void actionPerformed(ActionEvent e) {
99              if(e.getSource() == buscarButton){
100                 try {
101                     buscarCliente(cpfTextField.getText());
102                 } catch (SQLException e1) {
103                     e1.printStackTrace();
104                 }
105             }
106             if (e.getSource() == listarTodosClientesButton){
107                 try {
108                     listarTodosClientes();
109                 } catch (SQLException e1) {
110                     e1.printStackTrace();
111                 }
112             }
113             if (e.getSource()==cadastrarButton){
114                 cadastrarCliente();
115             }
116     }
117
118
119 }
```

Código 48: PesquisarClientesController.java

6.1.13 Pesquisar Funcionários

```
1  package br.com.afazenda.controller;
2
```

```
3  import br.com.afazenda.model.dao.CargoImp;
4  import br.com.afazenda.model.dao.FuncionarioImp;
5  import br.com.afazenda.model.vo.Cargo;
6  import br.com.afazenda.model.vo.Funcionario;
7  import br.com.afazenda.view.MenuFuncionarios;
8  import singletons.UsuarioAtual;
9  import utils.ConnectionFactory;
10
11 import javax.swing.*;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import java.sql.SQLException;
15 import java.util.ArrayList;
16
17 /**
18  * Created by patrick e Elias Rabelo on 02/05/17.
19  */
20 public class PesquisarFuncionariosController implements ActionListener {
21
22     private static JFrame frame;
23     private JButton configura  esButton;
24     private JButton sairButton;
25     private JTextField cpfTextField;
26     private JButton buscarButton;
27     private JButton listarTodosOsFuncion riosButton;
28     private JList<String> list1;
29     private JButton voltarButton;
30     private JComboBox<String> cargoComboBox;
31     private JButton buscarButtonCargo;
32     private JButton inativarButton;
33     private JButton ativarButton;
34     private JButton cadastrarButton;
35     private JPanel MainPanel;
36     private JLabel nomeUsuarioAtualLabel;
37     private ArrayList<Cargo> cargoArrayList;
38
39     public PesquisarFuncionariosController() {
40         Funcionario funcionario = UsuarioAtual.getInstance().getFuncionarioAtual();
41
42         try {
43             cargoArrayList = new CargoImp(ConnectionFactory.getConnection()).listarCargos();
44             for (Cargo c : cargoArrayList) {
45                 cargoComboBox.addItem(c.getNome());
46             }
47         } catch (SQLException e) {
48             e.printStackTrace();
49         }
50
51         nomeUsuarioAtualLabel.setText(funcionario.getName());
52         buscarButton.addActionListener(this);
53         listarTodosOsFuncion riosButton.addActionListener(this);
54         voltarButton.addActionListener(this);
55         buscarButtonCargo.addActionListener(this);
56         inativarButton.addActionListener(this);
57         ativarButton.addActionListener(this);
58         cadastrarButton.addActionListener(this);
59
60         sairButton.addActionListener(new ActionListener() {
61             @Override
62             public void actionPerformed(ActionEvent e) {
63                 new UsuarioAtual().logoffFuncionario();
64                 new LoginController().iniciarJanela();
65                 frame.dispose();
66             }
67         });
68         voltarButton.addActionListener(new ActionListener() {
69             @Override
70             public void actionPerformed(ActionEvent e) {
71                 new MenuFuncionarios().iniciarJanela();
72                 frame.dispose();
73             }
74         });
75     }
76
77
78     public void iniciarJanela() {
79         frame = new JFrame("Pesquisar␣Funcionario");
80         frame.setContentPane(new PesquisarFuncionariosController().MainPanel);
81         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
82         frame.pack();
83         frame.setVisible(true);
84
85     }
86
87     private void buscarFuncionario() throws SQLException {
88
89         FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
90         Funcionario funcionario = funcionarioImp.BuscarFuncionario(cpfTextField.getText());
91         DefaultListModel<String> modelo = new DefaultListModel<String>();
92         modelo.addElement(funcionario.getName());
93         list1.setModel(modelo);
94     }
95
96     private void buscarFuncionarioCargo() throws SQLException {
```

```
97
98          String cargo = cargoArrayList.get(cargoComboBox.getSelectedIndex()).getNome();
99
100         FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
101         ArrayList<Funcionario> funcionarios = funcionarioImp.listarFuncionariosPorCargo(cargo);
102         DefaultListModel<String> modelo = new DefaultListModel<>();
103
104         for (Funcionario f : funcionarios) {
105             modelo.addElement(f.getName());
106         }
107
108         list1.setModel(modelo);
109     }
110
111     private void listarTodosFuncionarios() throws SQLException {
112         FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
113         ArrayList<Funcionario> funcionarios = funcionarioImp.listarFuncionarios();
114         DefaultListModel<String> modelo = new DefaultListModel<>();
115         int d = funcionarios.size();
116         Funcionario funcionario;
117         for (int i = 0; i < d; i++) {
118             funcionario = funcionarios.get(i);
119             modelo.addElement(funcionario.getName() + "␣-␣" + funcionario.getCpf());
120         }
121         list1.setModel(modelo);
122     }
123
124     private void ativarFuncionario() throws SQLException {
125         String cpf = list1.getSelectedValue();
126         cpf = cpf.substring(cpf.lastIndexOf('␣') + 1);
127         System.out.println(cpf);
128         FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
129         funcionarioImp.setAtivo(cpf);
130
131     }
132
133     private void inativarFuncionario() throws SQLException {
134         String cpf = list1.getSelectedValue();
135         cpf = cpf.substring(cpf.lastIndexOf('␣') + 1);
136         System.out.println(cpf);
137         FuncionarioImp funcionarioImp = new FuncionarioImp(ConnectionFactory.getConnection());
138         funcionarioImp.setInativo(cpf);
139     }
140
141     private void cadastrarFuncionario() {
142         CadastroFuncionarioController c = new CadastroFuncionarioController();
143         c.iniciarJanela();
144         frame.dispose();
145     }
146
147     @Override
148     public void actionPerformed(ActionEvent e) {
149         if (e.getSource() == buscarButton) {
150             try {
151                 buscarFuncionario();
152             } catch (SQLException e1) {
153                 e1.printStackTrace();
154             }
155
156         } else if (e.getSource() == listarTodosOsFuncion riosButton) {
157             try {
158                 listarTodosFuncionarios();
159             } catch (SQLException e1) {
160                 e1.printStackTrace();
161             }
162
163         } else if (e.getSource() == ativarButton) {
164             try {
165                 ativarFuncionario();
166             } catch (SQLException e1) {
167                 e1.printStackTrace();
168             }
169
170         } else if (e.getSource() == cadastrarButton) {
171             cadastrarFuncionario();
172
173         } else if (e.getSource() == inativarButton) {
174             try {
175                 inativarFuncionario();
176             } catch (SQLException e1) {
177                 e1.printStackTrace();
178             }
179         } else if (e.getSource() == buscarButtonCargo) {
180             try {
181                 buscarFuncionarioCargo();
182             } catch (SQLException e1) {
183                 e1.printStackTrace();
184             }
185         }
186     }
187 }
```

Código 49: PesquisarFuncionariosController.java

6.2 Fronteiras

6.2.1 Adicionar Item

```
1  package br.com.afazenda.view;
2
3  import br.com.afazenda.controller.LoginController;
4  import singletons.UsuarioAtual;
5
6  import javax.swing.*;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9
10 /**
11  * Created by patrick on 03/05/17.
12  */
13 public class AdicionarItem {
14     private JPanel MainPanel;
15     private JLabel nomeUsuarioAtualLabel;
16     private JButton configuracoesButton;
17     private JButton sairButton;
18     private JTextField textField1;
19     private JButton buscarButton;
20     private JList list1;
21     private JTextField textField2;
22     private JButton voltarButton;
23     private JButton adicionarItemButton;
24     private static JFrame frame;
25
26     public AdicionarItem() {
27         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
28         sairButton.addActionListener(new ActionListener() {
29             @Override
30             public void actionPerformed(ActionEvent actionEvent) {
31                 new UsuarioAtual().logoffFuncionario();
32                 new LoginController().iniciarJanela();
33                 frame.dispose();
34             }
35         });
36         configuracoesButton.addActionListener(new ActionListener() {
37             @Override
38             public void actionPerformed(ActionEvent e) {
39                 new Configura  esDeUsuario().iniciarJanela();
40                 frame.dispose();
41             }
42         });
43     }
44
45     public void iniciarJanela() {
46         frame = new JFrame("Venda");
47         frame.setContentPane(new AdicionarItem().MainPanel);
48         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
49         frame.pack();
50         frame.setVisible(true);
51         nomeUsuarioAtualLabel.setText(new UsuarioAtual().getFuncionarioAtual().getName());
52     }
53 }
```

Código 50: AdicionarItem.java

6.2.2 Cadastrar Produtos

```
1  package br.com.afazenda.view;
2
3  import br.com.afazenda.controller.LoginController;
4  import singletons.UsuarioAtual;
5
6  import javax.swing.*;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9
10 /**
11  * Created by patrick on 04/05/17.
12  */
13 public class CadastrarProdutos {
14     private JPanel MainPanel;
15     private JLabel nomeUsuarioAtualLabel;
16     private JButton configuracoesButton;
17     private JButton sairButton;
18     private JTextField textField1;
19     private JButton adicionarButton;
20     private JList list1;
21     private JButton voltarButton;
22     private JButton apagarButton;
23     private JComboBox comboBox1;
24     private static JFrame frame;
```

```
25
26      public void iniciarJanela() {
27          frame = new JFrame("Estoque");
28          frame.setContentPane(new CadastrarProdutos().MainPanel);
29          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
30          frame.pack();
31          frame.setVisible(true);
32      }
33
34      public CadastrarProdutos() {
35          voltarButton.addActionListener(new ActionListener() {
36              @Override
37              public void actionPerformed(ActionEvent actionEvent) {
38                  new MenuEstoque().iniciarJanela();
39                  frame.dispose();
40              }
41          });
42          configuracoesButton.addActionListener(new ActionListener() {
43              @Override
44              public void actionPerformed(ActionEvent e) {
45                  new Configura  esDeUsuario().iniciarJanela();
46                  frame.dispose();
47              }
48          });
49          sairButton.addActionListener(new ActionListener() {
50              @Override
51              public void actionPerformed(ActionEvent e) {
52                  new UsuarioAtual().logoffFuncionario();
53                  new LoginController().iniciarJanela();
54                  frame.dispose();
55              }
56          });
57      }
58  }
```

Código 51: CadastrarProdutos.java

6.2.3 Configurações de Usuario

```
1   package br.com.afazenda.view;
2
3   import singletons.UsuarioAtual;
4
5   import javax.swing.*;
6
7   /**
8    * Created by patrick on 04/05/17.
9    */
10  public class Configura  esDeUsuario {
11
12      private JLabel nomeUsuarioAtualLabel;
13      private JPanel MainPanel;
14      private static JFrame frame;
15
16      public Configura  esDeUsuario () {
17          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
18      }
19
20
21
22      public void iniciarJanela() {
23          frame = new JFrame("Configura  es");
24          frame.setContentPane(new Configura  esDeUsuario().MainPanel);
25          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
26          frame.pack();
27          frame.setVisible(true);
28
29      }
30  }
```

Código 52: ConfiguracoesDeUsuario.java

6.2.4 Confirma Nota Fiscal

```
1   package br.com.afazenda.view;
2
3   import javax.swing.*;
4
5   /**
6    * Created by patrick on 03/05/17.
7    */
8   public class ConfirmaNotaFiscal {
```

```
9       private JPanel MainPanel;
10      private JLabel nomeUsuarioAtualLabel;
11      private JButton configuracoesButton;
12      private JButton sairButton;
13      private JList list1;
14      private JButton voltarButton;
15      private JButton confirmarButton;
16
17      private static JFrame frame;
18
19      public void iniciarJanela() {
20          frame = new JFrame("Venda");
21          frame.setContentPane(new ConfirmaNotaFiscal().MainPanel);
22          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
23          frame.pack();
24          frame.setVisible(true);
25
26      }
27  }
```

Código 53: ConfirmaNotaFiscal.java

6.2.5 Criar Venda

```
1   package br.com.afazenda.view;
2
3   import br.com.afazenda.controller.LoginController;
4   import singletons.UsuarioAtual;
5
6   import javax.swing.*;
7
8   /**
9    * Created by patrick on 03/05/17.
10   */
11  public class CriarVenda {
12      private static JFrame frame;
13      private JPanel MainPanel;
14      private JLabel nomeUsuarioAtualLabel;
15      private JButton configuracoesButton;
16      private JButton sairButton;
17      private JTextField textField1;
18      private JTextField textField2;
19      private JButton cancelarButton;
20      private JButton gerarNotaFiscalButton;
21      private JButton removerItemButton;
22      private JButton adicionarItemButton;
23      private JList list1;
24
25      public CriarVenda() {
26          cancelarButton.addActionListener(actionEvent -> {
27              new MenuPrincipal().iniciarJanela();
28              frame.dispose();
29          });
30
31          adicionarItemButton.addActionListener(e -> {
32              new AdicionarItem().iniciarJanela();
33              nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
34          });
35
36          cancelarButton.addActionListener(actionEvent -> {
37              new MenuPrincipal().iniciarJanela();
38              frame.dispose();
39          });
40
41          configuracoesButton.addActionListener(e -> {
42              new Configura  esDeUsuario().iniciarJanela();
43              frame.dispose();
44          });
45
46          sairButton.addActionListener(e -> {
47              new UsuarioAtual().logoffFuncionario();
48              new LoginController().iniciarJanela();
49              frame.dispose();
50          });
51      }
52
53
54      public static void main(String[] args) {
55          new CriarVenda().iniciarJanela();
56      }
57
58      public void iniciarJanela() {
59          frame = new JFrame("Venda");
60          frame.setContentPane(new CriarVenda().MainPanel);
61          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
62          frame.pack();
63          frame.setVisible(true);
64      }
```

```
65
66
67  }
```

Código 54: CriarVenda.java

6.2.6 Gerenciar Animais

```
1   package br.com.afazenda.view;
2
3   import br.com.afazenda.controller.LoginController;
4   import br.com.afazenda.model.dao.AnimalImp;
5   import br.com.afazenda.model.dao.EspecieImp;
6   import br.com.afazenda.model.vo.Especie;
7   import singletons.UsuarioAtual;
8   import utils.ConnectionFactory;
9
10  import javax.swing.*;
11  import java.sql.SQLException;
12  import java.util.ArrayList;
13
14  /**
15   * Created by patrick on 02/05/17.
16   */
17  public class GerenciarAnimais {
18      private static JFrame frame;
19      private JPanel MainPanel;
20      private JButton configura  esButton;
21      private JButton sairButton;
22      private JButton adicionarButton;
23      private JList list1;
24      private JButton deletarButton1;
25      private JButton voltarButton;
26      private JComboBox especieComboBox;
27      private JLabel nomeUsuarioAtualLabel;
28      private ArrayList<Especie> especies;
29
30      private EspecieImp especieImp = new EspecieImp(ConnectionFactory.getConnection());
31      private AnimalImp animalImp = new AnimalImp(ConnectionFactory.getConnection());
32
33      public GerenciarAnimais() throws SQLException {
34
35          especies = especieImp.getEspecies();
36          especieComboBox.removeAllItems();
37          for (Especie e : especies) {
38              especieComboBox.addItem(e.getName());
39          }
40
41          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
42
43          EspecieImp especieImp;
44          try {
45              especieImp = new EspecieImp(ConnectionFactory.getConnection());
46              especies =  especieImp.getEspecies();
47
48          } catch (SQLException e) {
49              e.printStackTrace();
50          }
51
52          voltarButton.addActionListener(actionEvent -> {
53              new MenuAnimais().iniciarJanela();
54              frame.dispose();
55          });
56          sairButton.addActionListener(actionEvent -> {
57              new UsuarioAtual().logoffFuncionario();
58              new LoginController().iniciarJanela();
59              frame.dispose();
60          });
61          adicionarButton.addActionListener(e -> {
62
63          });
64          configura  esButton.addActionListener(actionEvent -> {
65              new Configura  esDeUsuario().iniciarJanela();
66              frame.dispose();
67          });
68
69          adicionarButton.addActionListener(e -> {
70
71          });
72      }
73
74      public static void main(String[] args) throws SQLException {
75          new GerenciarAnimais().iniciarJanela();
76      }
77
78      public void iniciarJanela() throws SQLException {
79          frame = new JFrame("Animal");
80          frame.setContentPane(new GerenciarAnimais().MainPanel);
```

```
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);


    }

}
```

Código 55: GerenciarAnimais.java

6.2.7 Gerenciar Cultura Agricola

```
package br.com.afazenda.view;

import br.com.afazenda.controller.LoginController;
import br.com.afazenda.model.dao.AgriculturaImp;
import br.com.afazenda.model.vo.CulturaAgricola;
import singletons.UsuarioAtual;
import utils.ConnectionFactory;
import utils.RegexMatches;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by patrick on 03/05/17.
 */
public class GerenciarCulturaAgricola {
    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JPanel MainPanel;
    private JTextField nomeTextField;
    private JButton cadastrarButton;
    private JButton voltarButton;
    private JButton removerButton;
    private JTextField DataDePlantioTextField;
    private JTextField DataDeColheitaTextField;
    private JList list1;
    private JLabel outputLabel;
    private static JFrame frame;
    private AgriculturaImp agriculturaImp = new AgriculturaImp(ConnectionFactory.getConnection());
    private ArrayList<CulturaAgricola> culturaAgricola  = new ArrayList<CulturaAgricola>();

    public GerenciarCulturaAgricola() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuAgricola().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
        removerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                removerCulturaAgricola();
                initLista();
                outputLabel.setText("");
            }
        });
        cadastrarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                cadastrarCulturaAgricola();
                initLista();

            }
        });
```

```
76
77          initLista();
78      }
79
80      private void removerCulturaAgricola(){
81          try {
82              CulturaAgricola culturaAgricolaAuxiliar = culturaAgricola.get(list1.getSelectedIndex());
83              agriculturaImp.excluirCulturaAgricola(culturaAgricolaAuxiliar);
84          } catch (SQLException e) {
85              e.printStackTrace();
86          }
87
88      }
89
90      private void initLista(){
91          try {
92              culturaAgricola = agriculturaImp.listarCulturaAgricola();
93              DefaultListModel listModel = new DefaultListModel();
94             // listModel.addElement(" Nome da Cultura | Periodo Plantio | Periodo Colheita "  );
95              for (CulturaAgricola percorrer : culturaAgricola) {
96                  listModel.addElement("␣Nome␣da␣Cultura:" + percorrer.getNome() + "␣|␣Periodo␣Plantio␣:␣" + percorrer.
       getPeriodoPlantio() + "␣|␣Periodo␣Colheita:␣" + percorrer.getPeriodoColheita() );
97              }
98              list1.setModel(listModel);
99          } catch (SQLException e) {
100             e.printStackTrace();
101         }
102     }
103
104     public void iniciarJanela() {
105         frame = new JFrame("Agr cola");
106         frame.setContentPane(new GerenciarCulturaAgricola().MainPanel);
107         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
108         frame.pack();
109         frame.setVisible(true);
110     }
111
112
113     private void cadastrarCulturaAgricola(){
114         if(!validaCampos())return;
115
116         CulturaAgricola culturaAgricola = new CulturaAgricola();
117         nomeTextField.getText();
118
119         culturaAgricola.setNome(nomeTextField.getText());
120         culturaAgricola.setPeriodoColheita(DataDeColheitaTextField.getText());
121         culturaAgricola.setPeriodoPlantio(DataDePlantioTextField.getText());
122         outputLabel.setText("Cultura␣Agr cola␣cadastrado␣com␣sucesso!");
123         try {
124             agriculturaImp.cadastrarCulturaAgricola(culturaAgricola);
125         } catch (SQLException e) {
126             e.printStackTrace();
127         }
128
129     }
130
131
132     private boolean validaCampos(){
133         boolean r =true;
134             if(DataDeColheitaTextField.getText().trim().isEmpty() ||
135                 DataDePlantioTextField.getText().trim().isEmpty()||
136                 nomeTextField.getText().trim().isEmpty() ){
137                 outputLabel.setText("Todos␣os␣campos␣s  o␣obrigat rios");
138                 r=false;
139         }else if (!RegexMatches.isNome(nomeTextField.getText())) {
140             outputLabel.setText("␣Nome␣inv lido");
141             r = false;
142         }
143
144         return r;
145     }
146
147
148 }
```

Código 56: GerenciarCulturaAgricola.java

6.2.8 Gerenciar Especies

```
1  package br.com.afazenda.view;
2
3  import br.com.afazenda.controller.LoginController;
4  import br.com.afazenda.model.dao.EspecieImp;
5  import br.com.afazenda.model.vo.Especie;
6  import singletons.UsuarioAtual;
7  import utils.ConnectionFactory;
8
9  import javax.swing.*;
```

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.sql.SQLException;
import java.util.ArrayList;

/**
 * Created by patrick on 02/05/17.
 */
public class GerenciarEspecies {

    private static JFrame frame;
    private JPanel MainPanel;
    private JButton configura  esButton;
    private JButton sairButton;
    private JList list1;
    private JButton deletarButton1;
    private JButton voltarButton;
    private JButton cadastrarButton;
    private JTextField textField1;
    private JLabel nomeUsuarioAtualLabel;
    private JLabel outputLabel;

    public GerenciarEspecies() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        outputLabel.setText("");
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuAnimais().iniciarJanela();
                frame.dispose();
            }
        });
        configura  esButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });

        list1.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                super.componentResized(e);
                preencheLista();
            }
        });
        cadastrarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                cadastrarEspecie();
                list1.setModel(new DefaultListModel());
                preencheLista();
            }
        });
    }

    public void iniciarJanela() {
        frame = new JFrame("Animal");
        frame.setContentPane(new GerenciarEspecies().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }

    private void preencheLista() {
        try {
            EspecieImp especieImp = new EspecieImp(ConnectionFactory.getConnection());
            ArrayList<Especie> especies = especieImp.getEspecies();
            DefaultListModel modelo = new DefaultListModel();
            int d = especies.size();
            Especie especie;
            for (int i = 0; i < d; i++) {
                especie = especies.get(i);
                modelo.addElement(especie.getName());
            }
            list1.setModel(modelo);
        } catch (SQLException x) {
            x.printStackTrace();
        }
    }


```

```
104     private void cadastrarEspecie() {
105
106         try {
107
108             EspecieImp novaEspecieIMP = new EspecieImp(ConnectionFactory.getConnection());
109            /* Especie novaEspecie = new Especie();
110             novaEspecie.setName(textField1.getText());
111             novaEspecie.setQuantidade(0);
112             */
113             novaEspecieIMP.insereEspecie(textField1.getText(), 0);
114
115             outputLabel.setText("Esp cie cadastrada com sucesso");
116
117
118         } catch (SQLException e) {
119             if (e.getErrorCode() == 0) {
120                 outputLabel.setText("Esta esp cie j  est cadastrada");
121             } else {
122                 outputLabel.setText("Houve um erro ao realizar conex o com o banco de dados");
123             }
124
125         }
126     }
127
128     /*public void actionPerformed(ActionEvent e) {
129         if (e.getSource()==cadastrarButton){
130             cadastrarEspecie();
131
132         }
133         if(e.getSource() == deletarButton1) {
134
135         }
136     }*/
137
138 }
```

Código 57: GerenciarEspecies.java

6.2.9 Gerenciar Plantio de Cultura

```
1  package br.com.afazenda.view;
2
3  import br.com.afazenda.controller.LoginController;
4  import br.com.afazenda.model.dao.AgriculturaImp;
5  import br.com.afazenda.model.vo.CulturaAgricola;
6  import br.com.afazenda.model.vo.Plantacao;
7  import singletons.UsuarioAtual;
8  import utils.ConnectionFactory;
9  import utils.RegexMatches;
10
11 import javax.swing.*;
12 import java.awt.event.ActionEvent;
13 import java.awt.event.ActionListener;
14 import java.sql.SQLException;
15 import java.util.ArrayList;
16
17 /**
18  * Created by patrick on 03/05/17.
19  */
20 public class GerenciarPlantioDeCultura {
21     private JPanel MainPanel;
22     private JLabel nomeUsuarioAtualLabel;
23     private JButton configuracoesButton;
24     private JButton sairButton;
25     private JComboBox culturaAgricolaComboBox;
26     private JButton cadastrarButton;
27     private JTextField dataDePlantioTextField;
28     private JButton voltarButton;
29     private JButton removerButton;
30     private JList list1;
31     private JLabel outputLabel;
32     private static JFrame frame;
33     private AgriculturaImp agriculturaImp = new AgriculturaImp(ConnectionFactory.getConnection());
34     private ArrayList<CulturaAgricola> culturaAgricola  = new ArrayList<CulturaAgricola>();
35     private ArrayList<Plantacao> plantioDeCultura = new ArrayList<Plantacao>();
36
37
38     public GerenciarPlantioDeCultura() {
39         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
40         voltarButton.addActionListener(new ActionListener() {
41             @Override
42             public void actionPerformed(ActionEvent actionEvent) {
43                 new MenuAgricola().iniciarJanela();
44                 frame.dispose();
45             }
46         });
47         configuracoesButton.addActionListener(new ActionListener() {
48             @Override
```

```
49              public void actionPerformed(ActionEvent e) {
50                  new Configura  esDeUsuario().iniciarJanela();
51                  frame.dispose();
52              }
53          });
54          sairButton.addActionListener(new ActionListener() {
55              @Override
56              public void actionPerformed(ActionEvent e) {
57                  new UsuarioAtual().logoffFuncionario();
58                  new LoginController().iniciarJanela();
59                  frame.dispose();
60              }
61          });
62
63          cadastrarButton.addActionListener(new ActionListener() {
64              @Override
65              public void actionPerformed(ActionEvent actionEvent) {
66                  cadastrarPlantioDeCultura();
67                  initLista();
68              }
69          });
70          removerButton.addActionListener(new ActionListener() {
71              @Override
72              public void actionPerformed(ActionEvent actionEvent) {
73
74              }
75          });
76
77          preencherCombo();
78          initLista();
79      }
80
81      private void initLista(){
82          try {
83              plantioDeCultura = agriculturaImp.listarPlantioDeCultura();
84              DefaultListModel listModel = new DefaultListModel();
85              // listModel.addElement(" Nome da Cultura | Periodo Plantio | Periodo Colheita "  );
86              for (Plantacao percorrer : plantioDeCultura) {
87                  listModel.addElement("␣Identifica  o␣do␣Plantio:" + percorrer.getIdPlantacao() + "␣|␣Cultura␣:␣" +
   percorrer.getCultura().getNome() + "␣|␣Data␣da␣Planta  o:␣" + percorrer.getDataPlantacao() );
88              }
89              list1.setModel(listModel);
90          } catch (SQLException e) {
91              e.printStackTrace();
92          }
93      }
94
95
96      private void preencherCombo(){
97          try {
98              culturaAgricola = agriculturaImp.listarCulturaAgricola();
99              DefaultComboBoxModel listModel = new DefaultComboBoxModel();
100             // listModel.addElement(" Nome da Cultura | Periodo Plantio | Periodo Colheita "  );
101             for (CulturaAgricola percorrer : culturaAgricola) {
102                 listModel.addElement(percorrer.getNome());
103             }
104             culturaAgricolaComboBox.setModel(listModel);
105         } catch (SQLException e) {
106             e.printStackTrace();
107         }
108     }
109
110     private void cadastrarPlantioDeCultura(){
111         if(!validaCampos())return;
112
113         Plantacao plantacao = new Plantacao();
114         dataDePlantioTextField.getText();
115
116         plantacao.setDataPlantacao(dataDePlantioTextField.getText());
117         CulturaAgricola c = new CulturaAgricola();
118         c = culturaAgricola.get(culturaAgricolaComboBox.getSelectedIndex());
119         plantacao.setCultura(c);
120         try {
121             agriculturaImp.cadastarPlantio(plantacao);
122             outputLabel.setText("Cultura␣Agr cola␣cadastrado␣com␣sucesso!");
123         } catch (SQLException e) {
124             e.printStackTrace();
125         }
126
127     }
128
129     private void removerPlantioDeCultura(){
130         try {
131             Plantacao plantacaoAuxiliar = plantioDeCultura.get(list1.getSelectedIndex());
132             agriculturaImp.excluirPlantioDeCultura(plantacaoAuxiliar);
133         } catch (SQLException e) {
134             e.printStackTrace();
135         }
136     }
137
138
139     private boolean validaCampos(){
140         boolean r =true;
141         if(dataDePlantioTextField.getText().trim().isEmpty()){
```

```
142             outputLabel.setText("Todos␣os␣campos␣s  o␣obrigat rios");
143             r=false;
144         }else if (!RegexMatches.isData(dataDePlantioTextField.getText())) {
145             outputLabel.setText("␣Data␣inv lida");
146             r = false;
147         }
148 
149         return r;
150     }
151     public void iniciarJanela() {
152         frame = new JFrame("Menu␣Principal");
153         frame.setContentPane(new GerenciarPlantioDeCultura().MainPanel);
154         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
155         frame.pack();
156         frame.setVisible(true);
157 
158     }
159 }
```

Código 58: GerenciarPlantioDeCultura.java

6.2.10 Gerenciar Produtos

```
1  package br.com.afazenda.view;
2
3  import br.com.afazenda.controller.LoginController;
4  import singletons.UsuarioAtual;
5
6  import javax.swing.*;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9
10 /**
11  * Created by patrick on 04/05/17.
12  */
13 public class GerenciarProdutos {
14     private JPanel MainPanel;
15     private JLabel nomeUsuarioAtualLabel;
16     private JButton configuracoesButton;
17     private JButton sairButton;
18     private JTextField textField1;
19     private JButton buscarButton;
20     private JList list1;
21     private JButton voltarButton;
22     private JButton removerButton;
23     private JButton adicionarButton;
24     private JTextField textField2;
25     private static JFrame frame;
26
27     public void iniciarJanela() {
28         frame = new JFrame("Estoque");
29         frame.setContentPane(new GerenciarProdutos().MainPanel);
30         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
31         frame.pack();
32         frame.setVisible(true);
33     }
34
35     public GerenciarProdutos() {
36         voltarButton.addActionListener(new ActionListener() {
37             @Override
38             public void actionPerformed(ActionEvent actionEvent) {
39                 new MenuEstoque().iniciarJanela();
40                 frame.dispose();
41             }
42         });
43         configuracoesButton.addActionListener(new ActionListener() {
44             @Override
45             public void actionPerformed(ActionEvent e) {
46                 new Configura  esDeUsuario().iniciarJanela();
47                 frame.dispose();
48             }
49         });
50         sairButton.addActionListener(new ActionListener() {
51             @Override
52             public void actionPerformed(ActionEvent e) {
53                 new UsuarioAtual().logoffFuncionario();
54                 new LoginController().iniciarJanela();
55                 frame.dispose();
56             }
57         });
58     }
59 }
```

Código 59: GerenciarProdutos.java

6.2.11 Login

```
package br.com.afazenda.view;

/**
 *
 * @author Artur Santos Nascimento
 */

import javax.swing.*;
import javax.swing.text.MaskFormatter;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;

public class Login extends JFrame{

    JButton entrar = new JButton("Entrar");
    JFormattedTextField cpf = new JFormattedTextField();
    JTextField psw = new JPasswordField();
    JLabel erro = new JLabel("Dados␣incorretos.␣Tente␣de␣novo");

    public Login(){

        // I N I C I A L I Z A O  INICIAL DA JANELA
        super("Fazenda");
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());

        Container c1 = new JPanel();
        Container c2 = new JPanel();
        Container c3 = new JPanel();
        c1.setLayout(new GridLayout(2, 3));
        c2.setLayout(new GridLayout(6, 4, 5, 5));
        c3.setLayout(new GridLayout(2, 3));

        erro.setForeground(Color.RED);
        erro.setVisible(false);
        setarMascara();
        entrar.addActionListener(new ActionBotao());

        c1.add(new JLabel(""));
        c1.add(new JLabel(""));
        c1.add(new JLabel(""));
        c1.add(new JLabel(""));
        c1.add(new JLabel("␣␣␣␣␣␣␣␣␣A␣FAZENDA"));
        c1.add(new JLabel(""));

        // CONTEINER 2, COM OS LABELS E ENTRADAS DE TEXTO
        // LINHA 1
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));

        // LINHA 2 E 3
        c2.add(new JLabel(""));
        c2.add(new JLabel("CPF:␣"));
        c2.add(cpf);
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel("Senha:␣"));
        c2.add(psw);
        c2.add(new JLabel(""));

        // LINHA 4
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));
        c2.add(new JLabel(""));

        c3.add(new JLabel(""));
        c3.add(entrar);
        c3.add(new JLabel(""));
        c3.add(new JLabel(""));
        c3.add(erro);
        c3.add(new JLabel(""));

        cp.add(BorderLayout.NORTH, c1);
        cp.add(BorderLayout.CENTER, c2);
        cp.add(BorderLayout.SOUTH, c3);

        // TORNANDO VISIVEL E ATRIBUINDO ACAO DE FECHAMENTO
        setSize(600, 300);
```

```
92          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
93          setResizable(false);
94          setLocationRelativeTo(null);
95          setVisible(true);
96      }
97
98      /*public static void main(String[] args) {
99          new Login();
100     }*/
101
102     // FUNCAO QUE SETA A MASCARA PRO CPF
103     private void setarMascara() {
104         try {
105             MaskFormatter mascara_cpf = new MaskFormatter("###.###.###-##");
106             mascara_cpf.setValidCharacters("0123456789");
107             cpf = new javax.swing.JFormattedTextField(mascara_cpf);
108         } catch (ParseException error_mask) {
109             JOptionPane.showMessageDialog(null, "Erro:␣" + error_mask);
110         }
111     }
112
113     private class ActionBotao implements ActionListener {
114         @Override
115         public void actionPerformed(ActionEvent ae) {
116
117             if(ae.getSource() == entrar) {
118                 String cpfusuario = cpf.getText();
119                 String senha = psw.getText();
120
121                 // SE QUISEREM RETIRAR OS PONTOS E  H FEN
122 //              cpfusuario = cpfusuario.replace(".", "");
123 //              cpfusuario = cpfusuario.replace("-", "");
124
125 //              System.out.println(cpfusuario +" " +senha);
126
127                 // ESTE IF DEVE RECEBER A  V A L I C A O  TRUE OU FALSE DA  A U T E N T I C A O  DO USUARIO
128                 if(false){
129                     // CHAMA PROXIMA JANELA
130                     erro.setVisible(false);
131                     setVisible(false);
132                     //new Menu();
133                 } else{
134                     // MOSTRA JLABEL DE ERRO
135                     erro.setVisible(true);
136
137                     //new Menu();
138                 }
139             }
140         }
141     }
142
143 }
```

Código 60: Login.java

6.2.12 Menu Agrícola

```
1  package br.com.afazenda.view;
2
3  import br.com.afazenda.controller.LoginController;
4  import singletons.UsuarioAtual;
5
6  import javax.swing.*;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9
10 /**
11  * Created by patrick on 03/05/17.
12  */
13 public class MenuAgricola {
14     private JLabel nomeUsuarioAtualLabel;
15     private JButton configuracoesButton;
16     private JButton sairButton;
17     private JButton gerenciarPlantioDeCulturaButton;
18     private JButton gerenciarCulturaAgricolaButton;
19     private JButton voltarButton;
20     private JPanel MainPanel;
21     private static JFrame frame;
22
23     public MenuAgricola() {
24         nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
25         voltarButton.addActionListener(new ActionListener() {
26             @Override
27             public void actionPerformed(ActionEvent actionEvent) {
28                 new MenuPrincipal().iniciarJanela();
29                 frame.dispose();
30             }
31         });
```

```
        gerenciarCulturaAgricolaButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new GerenciarCulturaAgricola().iniciarJanela();
                frame.dispose();
            }
        });
        gerenciarPlantioDeCulturaButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new GerenciarPlantioDeCultura().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
    }

    public void iniciarJanela() {
        frame = new JFrame("Agr cola");
        frame.setContentPane(new MenuAgricola().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }
}
```

Código 61: MenuAgricola.javaa

6.2.13 Menu Animais

```
package br.com.afazenda.view;

import br.com.afazenda.controller.LoginController;
import singletons.UsuarioAtual;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;

/**
 * Created by patrick on 02/05/17.
 */
public class MenuAnimais {
    private static JFrame frame;
    private JPanel MainPanel;
    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JButton gerenciarEspeciesButton;
    private JButton gerenciarAnimaisButton;
    private JButton voltarButton;

    public MenuAnimais() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        gerenciarAnimaisButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                try {
                    new GerenciarAnimais().iniciarJanela();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                frame.dispose();
            }
        });
        gerenciarEspeciesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new GerenciarEspecies().iniciarJanela();
                frame.dispose();
            }
        });
```

```
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuPrincipal().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
    }

    public void iniciarJanela() {
        frame = new JFrame("Animal");
        frame.setContentPane(new MenuAnimais().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }
}
```

Código 62: MenuAnimais.java

6.2.14 Menu Cliente

```
package br.com.afazenda.view;

import br.com.afazenda.controller.CadastroClienteController;
import br.com.afazenda.controller.LoginController;
import br.com.afazenda.controller.PesquisarClientes;
import singletons.UsuarioAtual;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Created by patrick on 02/05/17.
 */
public class MenuCliente {
    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JButton gerenciarClientesButton;
    private JButton cadastrarClienteButton;
    private JButton voltarButton;
    private JPanel MainPanel;
    private static JFrame frame;

    public MenuCliente() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuPrincipal().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
        cadastrarClienteButton.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                new CadastroClienteController().iniciarJanela();
                frame.dispose();
            }
        });
        gerenciarClientesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new PesquisarClientes().iniciarJanela();
                frame.dispose();
            }
        });
    }

    public void iniciarJanela() {
        frame = new JFrame("Clientes");
        frame.setContentPane(new MenuCliente().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);


    }
}
```

Código 63: MenuCliente.java

6.2.15 Menu Contrato

```
package br.com.afazenda.view;

import br.com.afazenda.controller.CadastroContratoController;
import br.com.afazenda.controller.LoginController;
import singletons.UsuarioAtual;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Created by patrick on 03/05/17.
 */
public class MenuContrato {
    private JPanel MainPanel;
    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JButton pesquisarContratoButton;
    private JButton adicionarContratoButton;
    private JButton voltarButton;
    private static JFrame frame;

    public void iniciarJanela() {
        frame = new JFrame("Funcion rios");
        frame.setContentPane(new MenuContrato().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }

    public MenuContrato() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuFuncionarios().iniciarJanela();
                frame.dispose();
            }
        });
        adicionarContratoButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new CadastroContratoController().iniciarJanela();
                frame.dispose();
            }
        });
        pesquisarContratoButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new PesquisarContrato().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
```

```
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
    }
}
```

Código 64: MenuContrato.java

6.2.16 Menu Estoque

```
package br.com.afazenda.view;

import br.com.afazenda.controller.LoginController;
import singletons.UsuarioAtual;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Created by patrick on 04/05/17.
 */
public class MenuEstoque {
    private JPanel MainPanel;
    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JButton gerenciarProdutosButton;
    private JButton cadastrarProdutoButton;
    private JButton voltarButton;
    private static JFrame frame;

    public MenuEstoque() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        cadastrarProdutoButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new CadastrarProdutos().iniciarJanela();
                frame.dispose();
            }
        });
        gerenciarProdutosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new GerenciarProdutos().iniciarJanela();
                frame.dispose();
            }
        });
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuPrincipal().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
    }

    public void iniciarJanela() {
        frame = new JFrame("Estoque");
        frame.setContentPane(new MenuEstoque().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }

```

```
}
```

Código 65: MenuEstoque.java

6.2.17 Menu Funcionários

```
package br.com.afazenda.view;

import br.com.afazenda.controller.LoginController;
import br.com.afazenda.controller.PesquisarFuncionariosController;
import singletons.UsuarioAtual;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Created by patrick on 02/05/17.
 */
public class MenuFuncionarios {
    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JButton gerenciarCargosButton;
    private JButton gerenciarFuncionariosButton;
    private JButton gerenciarContratosButton;
    private JButton voltarButton;
    private JPanel MainPanel;

    private static JFrame frame;

    public MenuFuncionarios() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        voltarButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuPrincipal().iniciarJanela();
                frame.dispose();
            }
        });
        gerenciarCargosButton.addActionListener(new ActionListener() {
            @Override
            /**
             N o funciona
             */
            public void actionPerformed(ActionEvent actionEvent) {
                new PesquisarCargos().iniciarJanela();
                frame.dispose();
            }
        });
        gerenciarFuncionariosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new PesquisarFuncionariosController().iniciarJanela();
                frame.dispose();

            }
        });
        gerenciarContratosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuContrato().iniciarJanela();
                frame.dispose();
            }
        });
        configuracoesButton.addActionListener(new ActionListener() {
            @Override
            /**
             N o funciona
             */
            public void actionPerformed(ActionEvent e) {
                new Configura  esDeUsuario().iniciarJanela();
                frame.dispose();
            }
        });
        sairButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new UsuarioAtual().logoffFuncionario();
                new LoginController().iniciarJanela();
                frame.dispose();
            }
        });
    }

    public void iniciarJanela() {
        frame = new JFrame("Funcion rios");
        frame.setContentPane(new MenuFuncionarios().MainPanel);
```

```
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);

    }

}
```

Código 66: MenuFuncionarios.java

6.2.18 Menu Principal

```
package br.com.afazenda.view;

import br.com.afazenda.controller.CadastroFuncionarioController;
import br.com.afazenda.controller.LoginController;
import br.com.afazenda.model.vo.Funcionario;
import singletons.UsuarioAtual;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Created by patrick on 02/05/17.
 */
public class MenuPrincipal {

    private JLabel nomeUsuarioAtualLabel;
    private JButton configuracoesButton;
    private JButton sairButton;
    private JButton agricolaButton;
    private JButton estoqueButton;
    private JButton vendasButton;
    private JButton animalButton;
    private JButton clientesButton;
    private JButton funcionariosButton;
    private JPanel MainPanel;
    private static JFrame frame;

    public void iniciarJanela() {
        frame = new JFrame("Menu Principal");
        frame.setContentPane(new MenuPrincipal().MainPanel);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }

    public MenuPrincipal() {
        nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
        funcionariosButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuFuncionarios().iniciarJanela();
                frame.dispose();

            }
        });
        clientesButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {

                new MenuCliente().iniciarJanela();
                frame.dispose();
            }
        });
        estoqueButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuEstoque().iniciarJanela();
                frame.dispose();
            }
        });
        vendasButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new CriarVenda().iniciarJanela();
                frame.dispose();
            }
        });
        agricolaButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent actionEvent) {
                new MenuAgricola().iniciarJanela();
                frame.dispose();
            }
        });
```

```
77          animalButton.addActionListener(new ActionListener() {
78              @Override
79              public void actionPerformed(ActionEvent actionEvent) {
80                  new MenuAnimais().iniciarJanela();
81                  frame.dispose();
82              }
83          });
84          sairButton.addActionListener(new ActionListener() {
85              @Override
86              public void actionPerformed(ActionEvent actionEvent) {
87                  new UsuarioAtual().logoffFuncionario();
88                  new LoginController().iniciarJanela();
89                  frame.dispose();
90              }
91          });
92          configuracoesButton.addActionListener(new ActionListener() {
93              @Override
94              public void actionPerformed(ActionEvent actionEvent) {
95                  new Configura  esDeUsuario().iniciarJanela();
96                  frame.dispose();
97              }
98          });
99
100     }
101
102
103 }
```

Código 67: MenuPrincipal.java

6.2.19 Pesquisar Cargos

```
1   package br.com.afazenda.view;
2
3   import br.com.afazenda.controller.LoginController;
4   import singletons.UsuarioAtual;
5
6   import javax.swing.*;
7   import java.awt.event.ActionEvent;
8   import java.awt.event.ActionListener;
9
10  /**
11   * Created by patrick on 02/05/17.
12   */
13  public class PesquisarCargos {
14      private JButton configura  esButton;
15      private JButton sairButton;
16      private JTextField textField1;
17      private JButton adicionarButton;
18      private JList list1;
19      private JButton deletarButton1;
20      private JButton voltarButton;
21      private JPanel MainPanel;
22      private JLabel nomeUsuarioAtualLabel;
23
24      private static JFrame frame;
25
26      public PesquisarCargos() {
27          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
28          voltarButton.addActionListener(new ActionListener() {
29              @Override
30              public void actionPerformed(ActionEvent actionEvent) {
31                  new MenuFuncionarios().iniciarJanela();
32                  frame.dispose();
33              }
34          });
35          configura  esButton.addActionListener(new ActionListener() {
36              @Override
37              public void actionPerformed(ActionEvent e) {
38                  new Configura  esDeUsuario().iniciarJanela();
39                  frame.dispose();
40              }
41          });
42          sairButton.addActionListener(new ActionListener() {
43              @Override
44              public void actionPerformed(ActionEvent e) {
45                  new UsuarioAtual().logoffFuncionario();
46                  new LoginController().iniciarJanela();
47                  frame.dispose();
48              }
49          });
50      }
51
52      public void iniciarJanela() {
53          frame = new JFrame("Funcion rios");
54          frame.setContentPane(new PesquisarCargos().MainPanel);
55          frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
56          frame.pack();
```

```
57          frame.setVisible(true);
58      }
59  }
```

Código 68: PesquisarCargos.java

6.2.20 Pesquisar Contrato

```
1   package br.com.afazenda.view;
2
3   import br.com.afazenda.controller.CadastroContratoController;
4   import br.com.afazenda.controller.LoginController;
5   import br.com.afazenda.model.dao.ContratoImp;
6   import br.com.afazenda.model.vo.Contrato;
7   import singletons.UsuarioAtual;
8   import utils.ConnectionFactory;
9
10  import javax.swing.*;
11  import java.awt.event.ActionEvent;
12  import java.awt.event.ActionListener;
13  import java.sql.SQLException;
14  import java.util.ArrayList;
15
16  /**
17   * Created by patrick on 03/05/17.
18   */
19  @SuppressWarnings("Duplicates")
20  public class PesquisarContrato {
21      private static JFrame frame;
22      private JPanel MainPanel;
23      private JLabel nomeUsuarioAtualLabel;
24      private JButton configuracoesButton;
25      private JButton sairButton;
26      private JTextField CPFtextField;
27      private JButton buscarButton;
28      private JList list1;
29      private JTextArea textArea1;
30      private JButton voltarButton;
31      private JButton inativarButton;
32      private JButton novoButton;
33      private JButton listarTodosButton;
34      private ArrayList<Contrato> contratos;
35      private ContratoImp contratoImp = new ContratoImp(ConnectionFactory.getConnection());
36
37
38      public PesquisarContrato() {
39
40          try {
41              contratos = contratoImp.listarContratos();
42              preencherListaTodos();
43          } catch (SQLException e) {
44              e.printStackTrace();
45          }
46
47          nomeUsuarioAtualLabel.setText(UsuarioAtual.getInstance().getFuncionarioAtual().getName());
48          voltarButton.addActionListener(new ActionListener() {
49              @Override
50              public void actionPerformed(ActionEvent actionEvent) {
51                  new MenuContrato().iniciarJanela();
52                  frame.dispose();
53              }
54          });
55          novoButton.addActionListener(new ActionListener() {
56              @Override
57              public void actionPerformed(ActionEvent actionEvent) {
58                  new CadastroContratoController().iniciarJanela();
59                  frame.dispose();
60              }
61          });
62          configuracoesButton.addActionListener(new ActionListener() {
63              @Override
64              public void actionPerformed(ActionEvent e) {
65                  new Configura  esDeUsuario().iniciarJanela();
66                  frame.dispose();
67              }
68          });
69
70          sairButton.addActionListener(new ActionListener() {
71              @Override
72              public void actionPerformed(ActionEvent e) {
73                  new UsuarioAtual().logoffFuncionario();
74                  new LoginController().iniciarJanela();
75                  frame.dispose();
76              }
77          });
78
79          listarTodosButton.addActionListener(new ActionListener() {
80              @Override
```

```
81              public void actionPerformed(ActionEvent e) {
82
83                  preencherListaTodos();
84
85              }
86          });
87
88
89          inativarButton.addActionListener(new ActionListener() {
90              @Override
91              public void actionPerformed(ActionEvent e) {
92                  int index = list1.getSelectedIndex();
93                  try {
94                      contratoImp.inativarContrato(contratos.get(index));
95                      preencherListaTodos();
96                  } catch (SQLException e1) {
97                      e1.printStackTrace();
98                  }
99
100             }
101         });
102         buscarButton.addActionListener(new ActionListener() {
103             @Override
104             public void actionPerformed(ActionEvent e) {
105                 preencherListaPorCPF();
106             }
107         });
108     }
109
110     public static void main(String[] args) {
111         new PesquisarContrato().iniciarJanela();
112     }
113
114     public void iniciarJanela() {
115         frame = new JFrame("Funcion rios");
116         frame.setContentPane(new PesquisarContrato().MainPanel);
117         frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
118         frame.pack();
119         frame.setVisible(true);
120
121     }
122
123     private void preencherListaTodos() {
124         try {
125             contratos = contratoImp.listarContratos();
126             DefaultListModel listModel = new DefaultListModel();
127
128             for (Contrato contrato : contratos) {
129                 String estado;
130                 estado = contrato.isAtivo() ? "ativo" : "inativo";
131
132                 String s = "Id␣do␣contrato:␣" + contrato.getId() + "␣|␣CPF:␣" + contrato.getFuncionario().getCpf() + ""
      +
133                         "␣|␣Cargo:␣" + contrato.getCargo().getNome() + "␣|␣Status:␣" + estado;
134                 listModel.addElement(s);
135             }
136             list1.setModel(listModel);
137
138
139         } catch (SQLException e1) {
140             e1.printStackTrace();
141         }
142     }
143
144     private void preencherListaPorCPF() {
145         try {
146             String CPF = CPFtextField.getText();
147             contratos = contratoImp.listarContratosPorCPF(CPF);
148             DefaultListModel listModel = new DefaultListModel();
149
150             for (Contrato c : contratos) {
151                 String estado;
152                 estado = c.isAtivo() ? "ativo" : "inativo";
153
154                 String s = "Id␣do␣contrato:␣" + c.getId() + "␣|␣CPF:␣" + c.getFuncionario().getCpf() + "" +
155                         "␣|␣Cargo:␣" + c.getCargo().getNome() + "␣|␣Status:␣" + estado;
156                 listModel.addElement(s);
157             }
158             list1.setModel(listModel);
159
160
161         } catch (SQLException e1) {
162             e1.printStackTrace();
163         }
164     }
165
166 }
```

Código 69: PesquisarContrato.java

6.2.21 Relatório Vendas

```java
package br.com.afazenda.view;

import javax.swing.*;

/**
 * Created by Marina on 04/05/2017.
 */
public class RelatorioVendas {
    private JPanel MainPanel;
    private JLabel nomeUsuarioAtualLabel;
    private JButton configura  esButton;
    private JButton sairButton;
    private JList list1;
    private JButton voltarButton;
    private JLabel outputLabel;
}
```

Código 70: RelatorioVendas.java

6.3 Banco de Dados

6.3.1 Banco de Dados

```sql
CREATE SCHEMA fazenda_bd;

SET search_path TO Fazenda_bd;

CREATE TABLE IF NOT EXISTS especie (
  idespecie  SERIAL PRIMARY KEY,
  nome       VARCHAR(45) NOT NULL,
  quantidade INT         NOT NULL -- lembrar da lista de produtos que    uma classe separada
);


CREATE TABLE IF NOT EXISTS endereco (
  idendereco SERIAL PRIMARY KEY,
  logradouro VARCHAR(45) NOT NULL,
  numero     INT         NOT NULL,
  bairro     VARCHAR(45) NOT NULL,
  cidade     VARCHAR(45) NOT NULL,
  estado     VARCHAR(2)  NOT NULL,
  cep        VARCHAR(10) NOT NULL
);

CREATE TABLE IF NOT EXISTS cliente (
  cpf          VARCHAR(14) PRIMARY KEY,
  nome         VARCHAR(100) NOT NULL,
  telefone     VARCHAR(20)  NOT NULL,
  idendereco   INT          NOT NULL,
  email        VARCHAR(100),
  dtnascimento VARCHAR(10),
  CONSTRAINT fk_endereco
  FOREIGN KEY (idendereco)
  REFERENCES endereco (idendereco)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS funcionario (
  cpf          VARCHAR(14) PRIMARY KEY,
  nome         VARCHAR(100) NOT NULL,
  telefone     VARCHAR(20),
  email        VARCHAR(100),
  idendereco   INT          NOT NULL,
  ativo        BOOLEAN,
  dtnascimento VARCHAR(10),
  CONSTRAINT fk_endereco
  FOREIGN KEY (idendereco)
  REFERENCES endereco (idendereco)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS cargo (
  idcargo SERIAL PRIMARY KEY,
  nome    VARCHAR(45) NOT NULL
);

CREATE TABLE IF NOT EXISTS contrato (
  idcontrato    SERIAL,
  dataInicio    VARCHAR(10),
  dataFim       VARCHAR(10),
  ativo         BOOLEAN,
```

```
  idfuncionario VARCHAR(14),
  idcargo       INT NOT NULL,
  PRIMARY KEY (idcontrato, idfuncionario, idcargo),
  CONSTRAINT fk_funcionario
  FOREIGN KEY (idfuncionario)
  REFERENCES funcionario (cpf)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION,
  CONSTRAINT fk_cargo
  FOREIGN KEY (idcargo)
  REFERENCES cargo (idcargo)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS venda (--- LEMBRAR DO PRODUTOS
  idvenda     SERIAL PRIMARY KEY,
  datavenda   VARCHAR(10)      NOT NULL,
  valor       DOUBLE PRECISION NOT NULL,
  cliente     VARCHAR(14)      NOT NULL,
  funcionario VARCHAR(14)      NOT NULL,
  CONSTRAINT fk_cliente
  FOREIGN KEY (cliente)
  REFERENCES cliente (cpf)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION,
  CONSTRAINT fk_funcionario
  FOREIGN KEY (funcionario)
  REFERENCES funcionario (cpf)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS animal (
  idanimal SERIAL PRIMARY KEY,
  especie  INT NOT NULL,
  CONSTRAINT fk_especie
  FOREIGN KEY (especie)
  REFERENCES especie (idespecie)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION
);

CREATE TABLE IF NOT EXISTS produto (
  idproduto       SERIAL PRIMARY KEY,
  nome            VARCHAR(45)      NOT NULL,
  preco           DOUBLE PRECISION NOT NULL,
  origem          CHAR             NOT NULL,
  unidadedemedida VARCHAR(45),
  datarecebimento VARCHAR(10),
  quantidade      INT
);


CREATE TABLE IF NOT EXISTS cultura_agricola (
  idcultura       SERIAL PRIMARY KEY,
  nome            VARCHAR(45) NOT NULL,
  periodoplantio  VARCHAR(45) NOT NULL,
  periodocolheita VARCHAR(45) NOT NULL
);

CREATE TABLE IF NOT EXISTS plantio (
  idplantio     SERIAL PRIMARY KEY,
  cultura       INT         NOT NULL,
  dataplantacao VARCHAR(10) NOT NULL,
  CONSTRAINT fk_cultura
  FOREIGN KEY (cultura)
  REFERENCES cultura_agricola (idcultura)
  ON UPDATE NO ACTION
  ON DELETE NO ACTION
);

CREATE TABLE item_venda (
  iditemvenda SERIAL PRIMARY KEY NOT NULL,
  idproduto   INT                NOT NULL,
  idvenda     INT                NOT NULL,
  quantidade  INT                NOT NULL,


  FOREIGN KEY (idproduto)
  REFERENCES produto (idproduto),
  FOREIGN KEY (idvenda)
  REFERENCES venda (idvenda)

  ON UPDATE NO ACTION
  ON DELETE NO ACTION

);

-- FUNCTION: fazenda_bd.insere_endereco(character varying, integer, character varying, character varying, character
     varying, character varying)

-- DROP FUNCTION fazenda_bd.insere_endereco(character varying, integer, character varying, character varying, character
      varying, character varying);
```

```
153
154 -- criado por: Alice Rodrigues
155
156 CREATE OR REPLACE FUNCTION fazenda_bd.insere_endereco(
157   plogradouro CHARACTER VARYING,
158   pnumero     INTEGER,
159   pbairro     CHARACTER VARYING,
160   pcidade     CHARACTER VARYING,
161   pestado     CHARACTER VARYING,
162   pcep        CHARACTER VARYING)
163   RETURNS INTEGER
164 LANGUAGE 'plpgsql'
165 COST 100
166 VOLATILE
167 AS $function$
168 DECLARE
169   vidEndereco INTEGER;
170
171 BEGIN
172   SELECT idendereco
173   INTO vidEndereco
174   FROM fazenda_bd.endereco
175   WHERE UPPER(logradouro) = UPPER(plogradouro) AND
176         numero = pnumero AND
177         UPPER(bairro) = UPPER(pbairro) AND
178         UPPER(cidade) = UPPER(pcidade) AND
179         UPPER(estado) = UPPER(pestado) AND
180         cep = pcep;
181   IF (vidEndereco IS NULL)
182   THEN
183
184     SELECT max(idendereco)
185     INTO vidEndereco
186     FROM fazenda_bd.endereco;
187     IF (vidEndereco IS NULL)
188     THEN
189       vidEndereco := 1;
190     ELSE
191       vidEndereco := vidEndereco + 1;
192     END IF;
193
194     INSERT INTO fazenda_bd.endereco (idendereco,
195                                      logradouro,
196                                      numero,
197                                      bairro,
198                                      cidade,
199                                      estado,
200                                      cep)
201     VALUES (vidEndereco,
202             plogradouro,
203             pnumero,
204             pbairro,
205             pcidade,
206             pestado,
207             pcep);
208   END IF;
209   RETURN vidEndereco;
210 END;
211
212 $function$;
213
214 ALTER FUNCTION fazenda_bd.insere_endereco( CHARACTER VARYING, INTEGER, CHARACTER VARYING, CHARACTER VARYING, CHARACTER
      VARYING, CHARACTER VARYING )
215 OWNER TO postgres;
216
217 -- FUNCTION: fazenda_bd.insere_venda(date, double precision, character varying, character varying)
218
219 -- DROP FUNCTION fazenda_bd.insere_venda(date, double precision, character varying, character varying);
220
221 CREATE OR REPLACE FUNCTION fazenda_bd.insere_venda(
222   pdatavenda   DATE,
223   pvalor       DOUBLE PRECISION,
224   pcliente     CHARACTER VARYING,
225   pfuncionario CHARACTER VARYING)
226   RETURNS INTEGER
227 LANGUAGE 'plpgsql'
228 COST 100
229 VOLATILE
230 AS $function$
231
232 DECLARE
233   vidVenda INTEGER;
234
235 BEGIN
236   SELECT max(idvenda) + 1
237   INTO vidVenda
238   FROM fazenda_bd.venda;
239
240   IF (vidVenda IS NULL)
241   THEN
242     vidVenda := 1;
243   ELSE
244     vidVenda := vidVenda + 1;
245   END IF;
```

```
246
247     INSERT INTO fazenda_bd.venda (idvenda,
248                                   datavenda,
249                                   valor,
250                                   cliente,
251                                   funcionario)
252     VALUES (vidVenda,
253             pdatavenda,
254             pvalor,
255             pcliente,
256             pfuncionario);
257     RETURN vidVenda;
258   END;
259
260   $function$;
261
262   ALTER FUNCTION fazenda_bd.insere_venda( DATE, DOUBLE PRECISION, CHARACTER VARYING, CHARACTER VARYING )
263   OWNER TO postgres;
```

Código 71: fazenda.sql

7 Diagramas

7.1 Diagrama de Classes

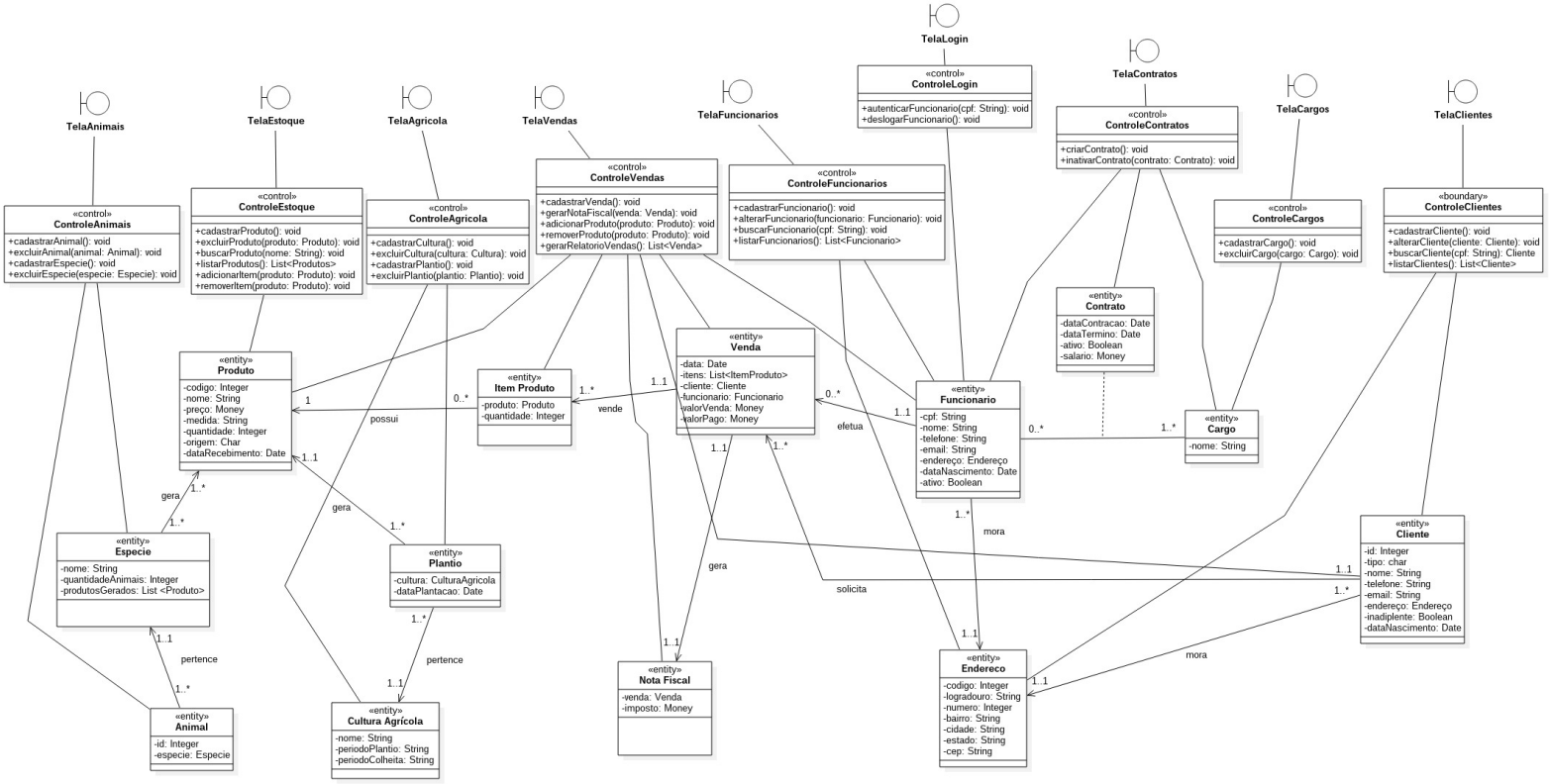


Figura 3: Diagrama de Classes

7.2 Diagramas de Atividade

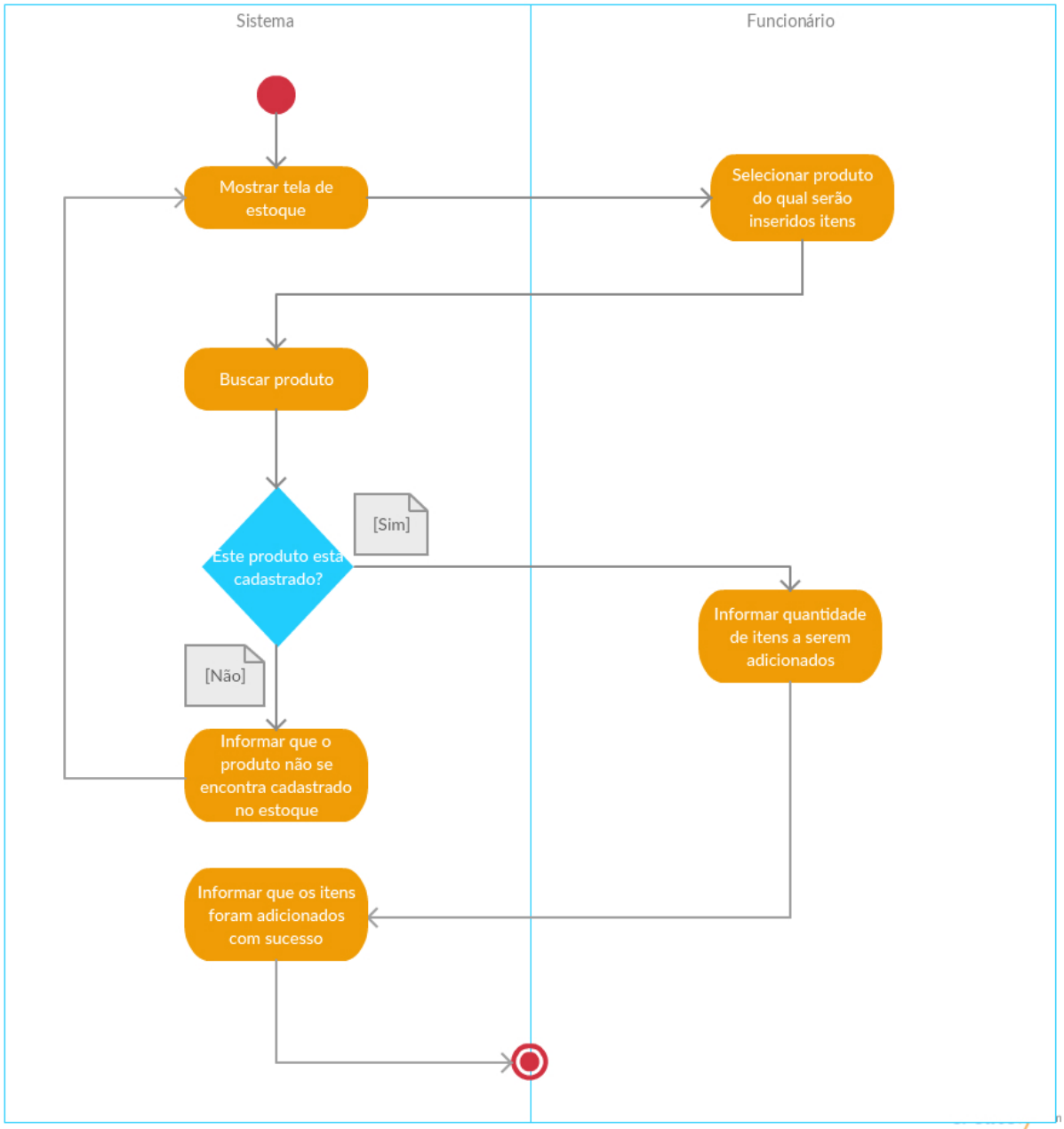


Figura 4: Adicionar Itens

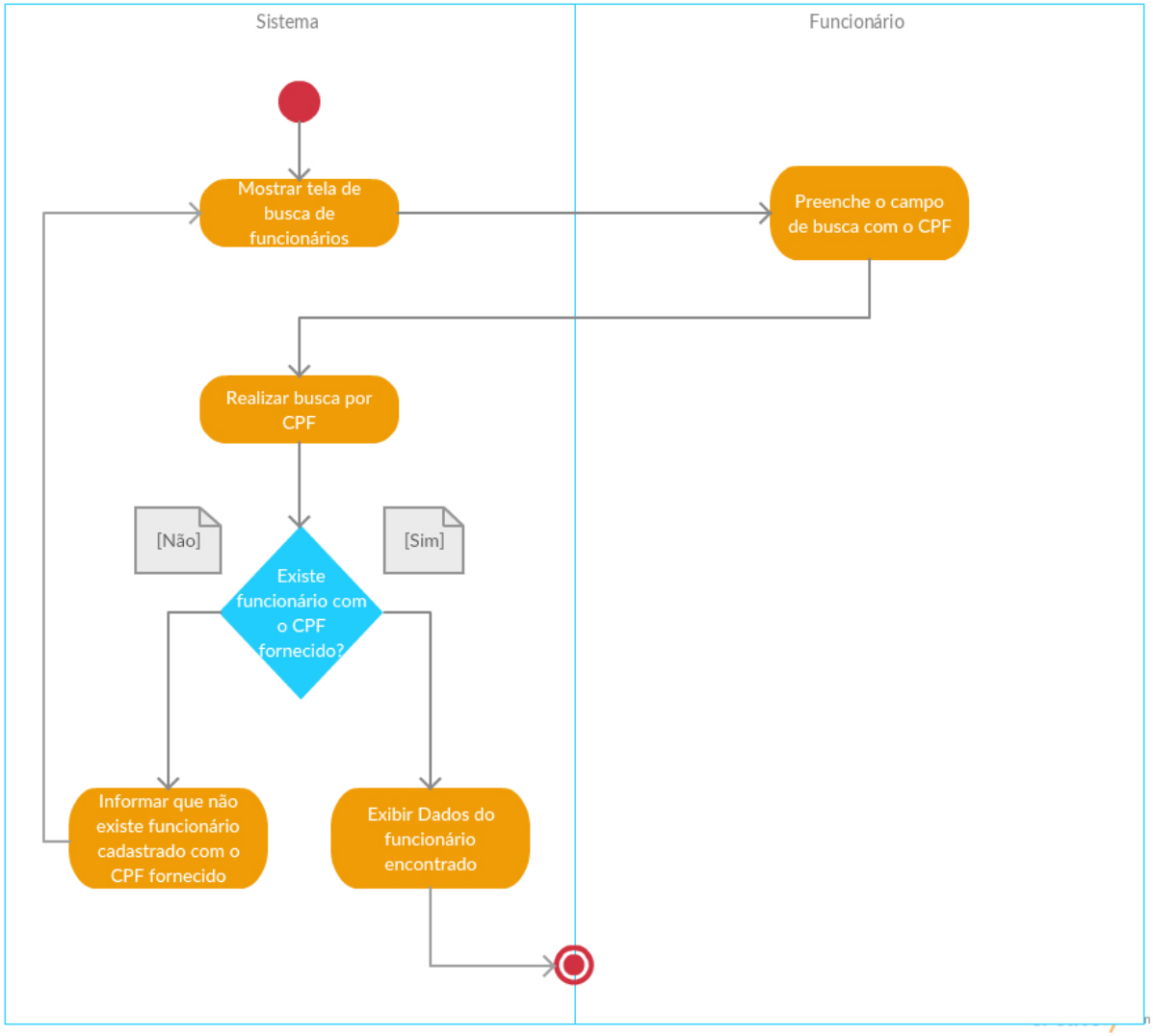


Figura 5: Buscar Funcionário por CPF

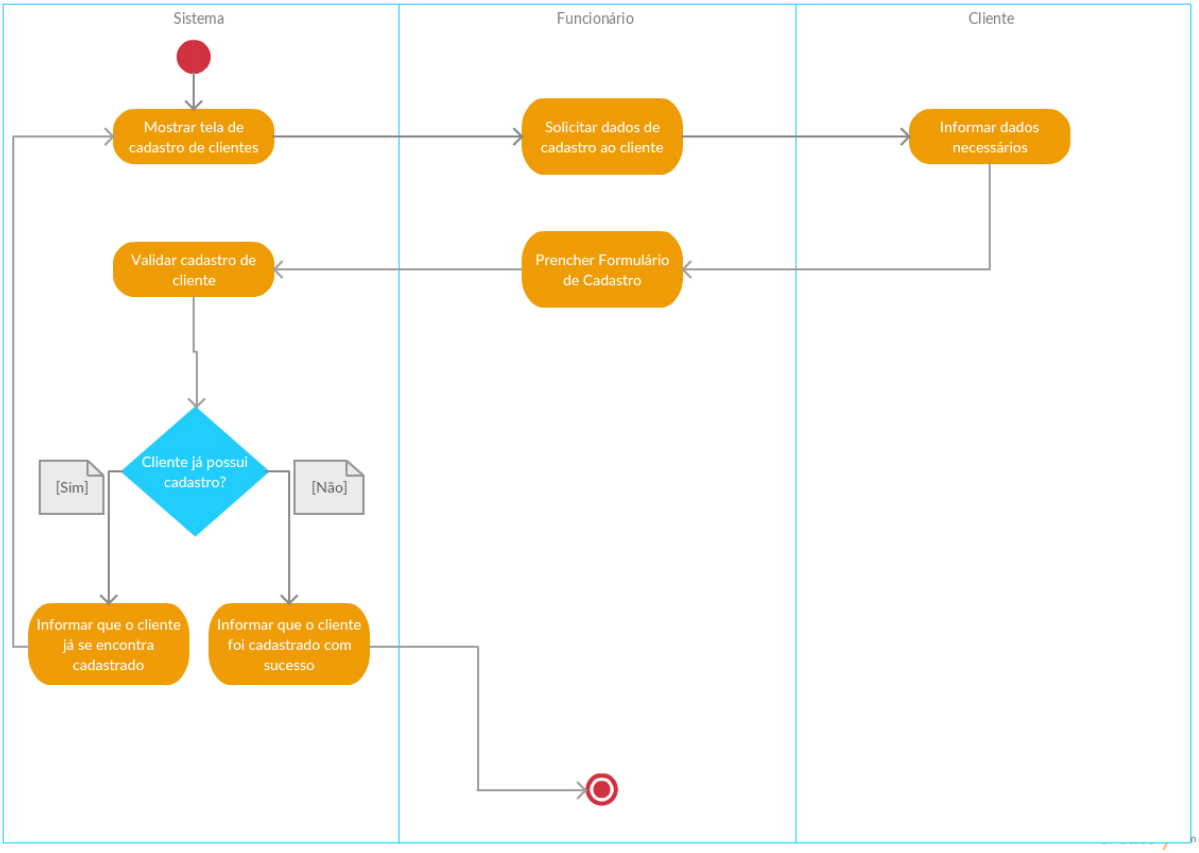


Figura 6: Cadastrar Cliente

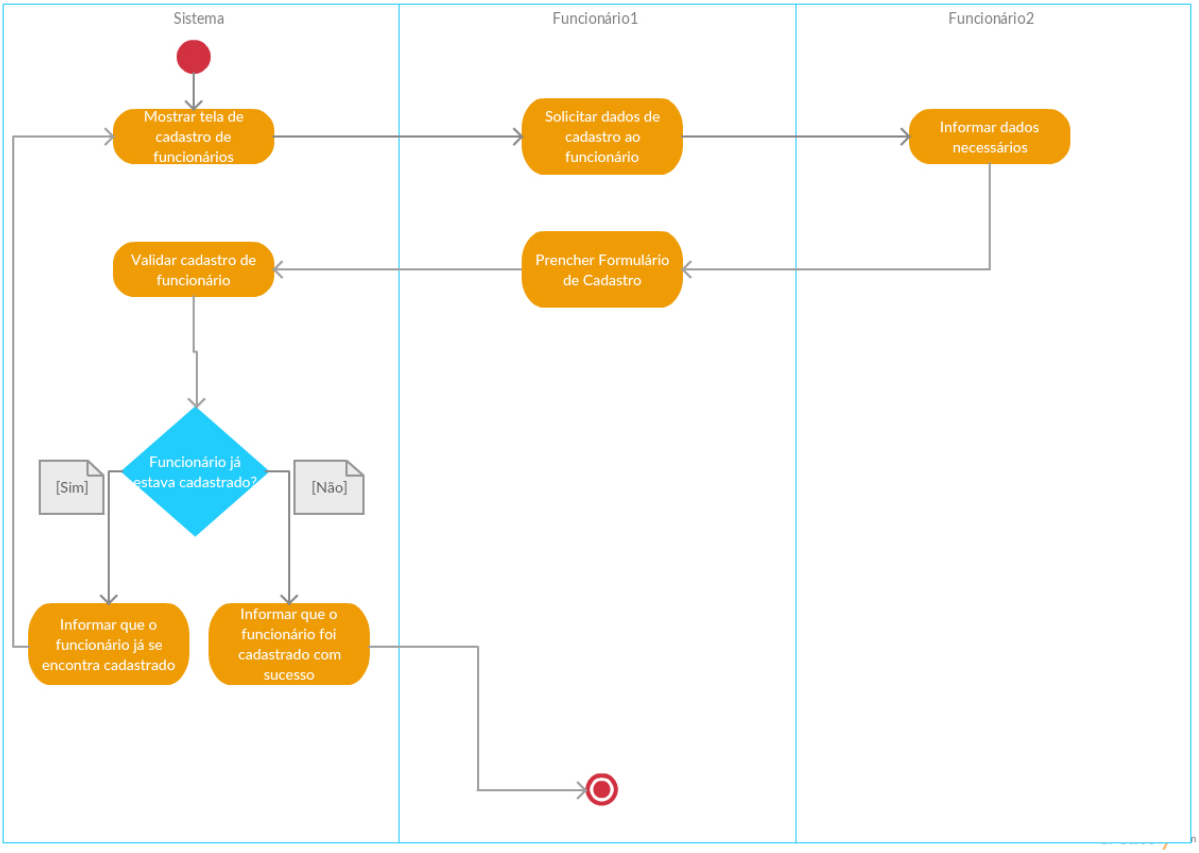


Figura 7: Cadastrar Funcionário

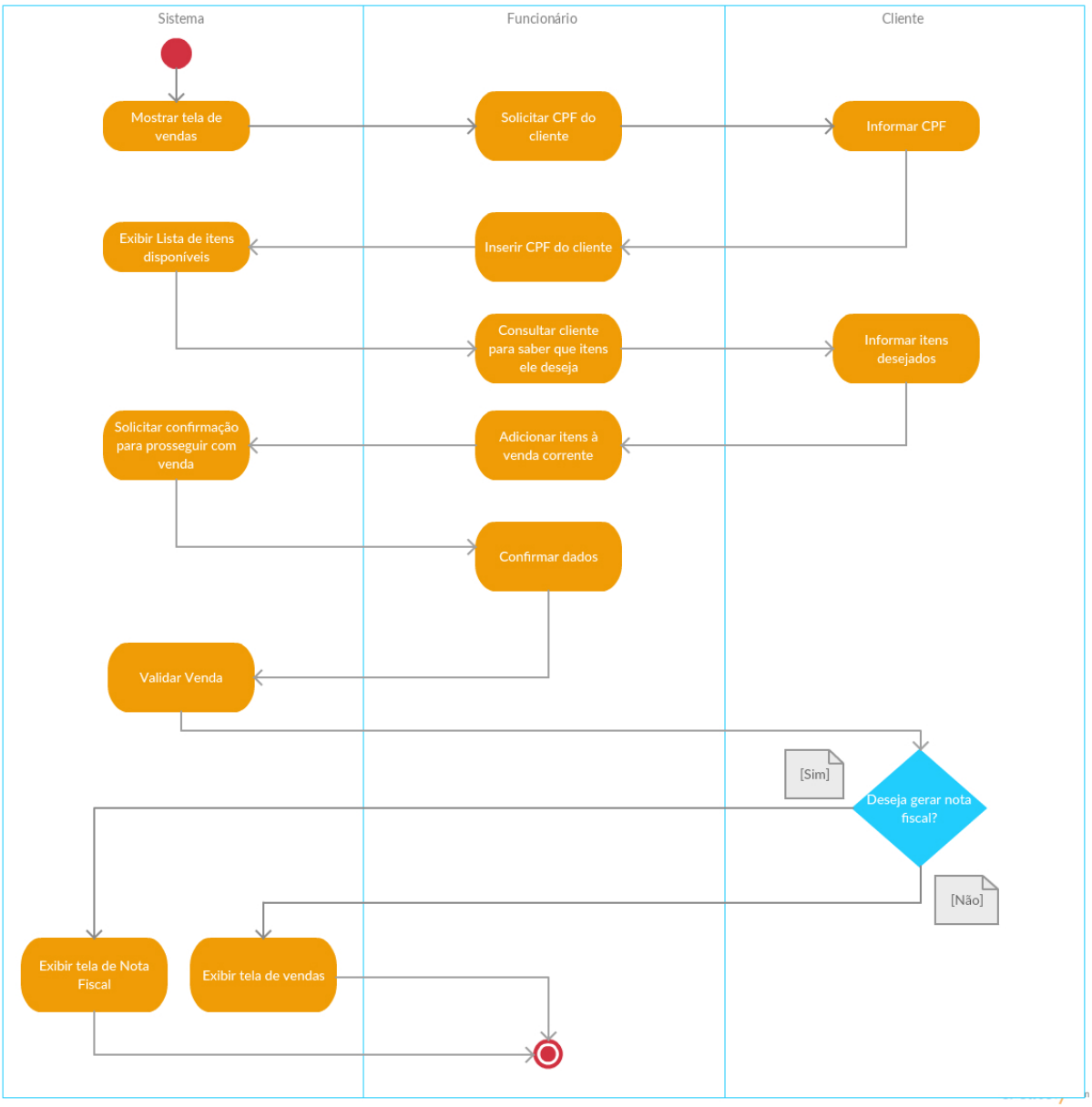


Figura 8: Criar Venda

7.3 Diagramas de Sequência

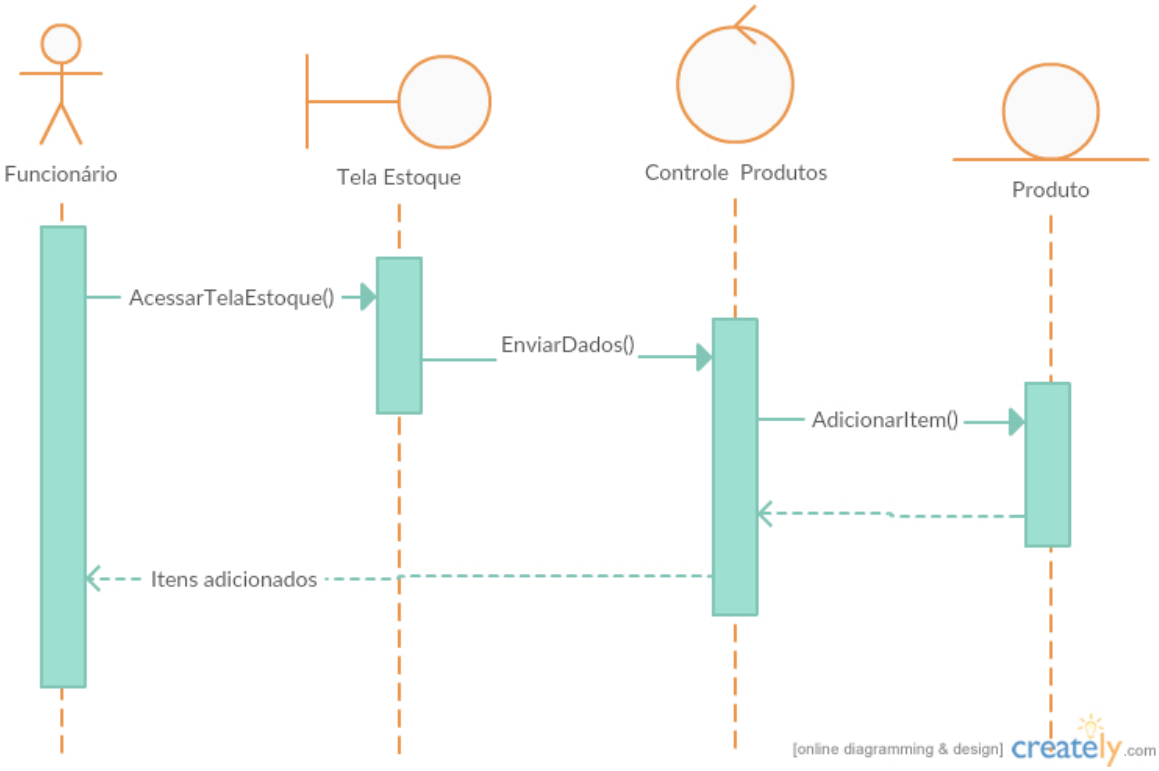


Figura 9: Adicionar Item

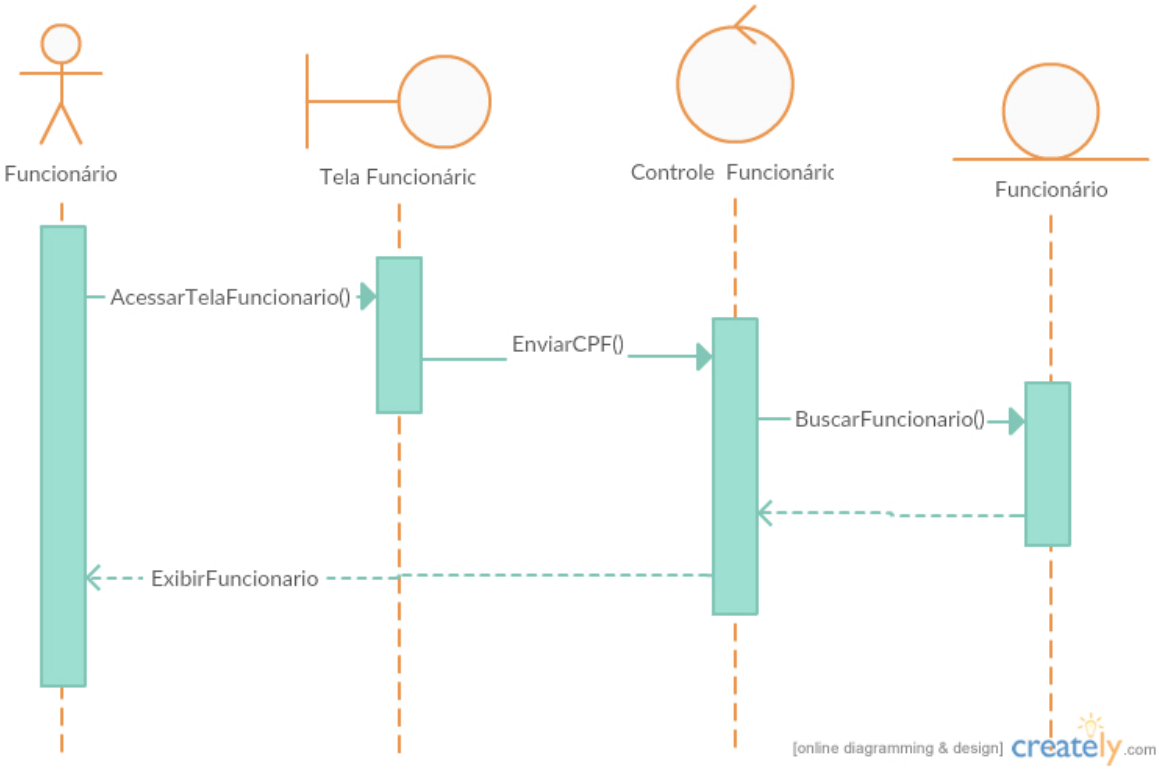


Figura 10: Buscar Funcionário

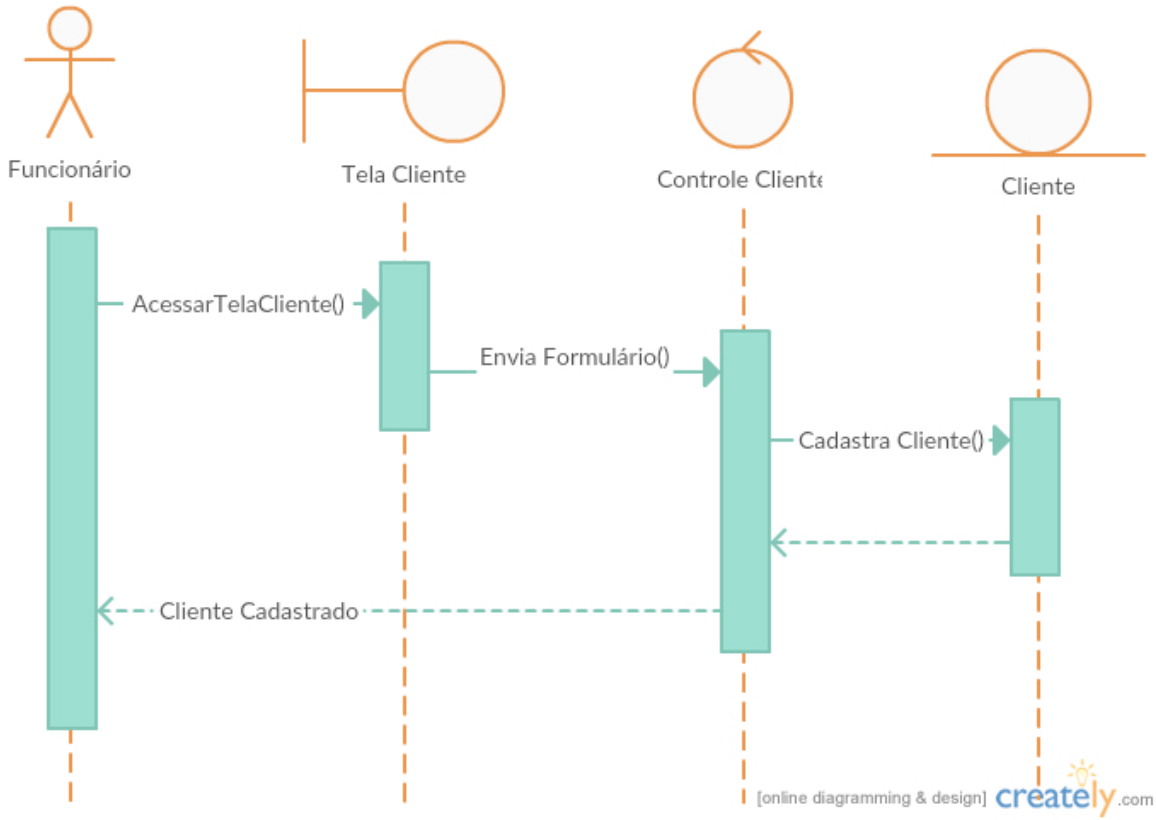


Figura 11: Cadastrar Cliente

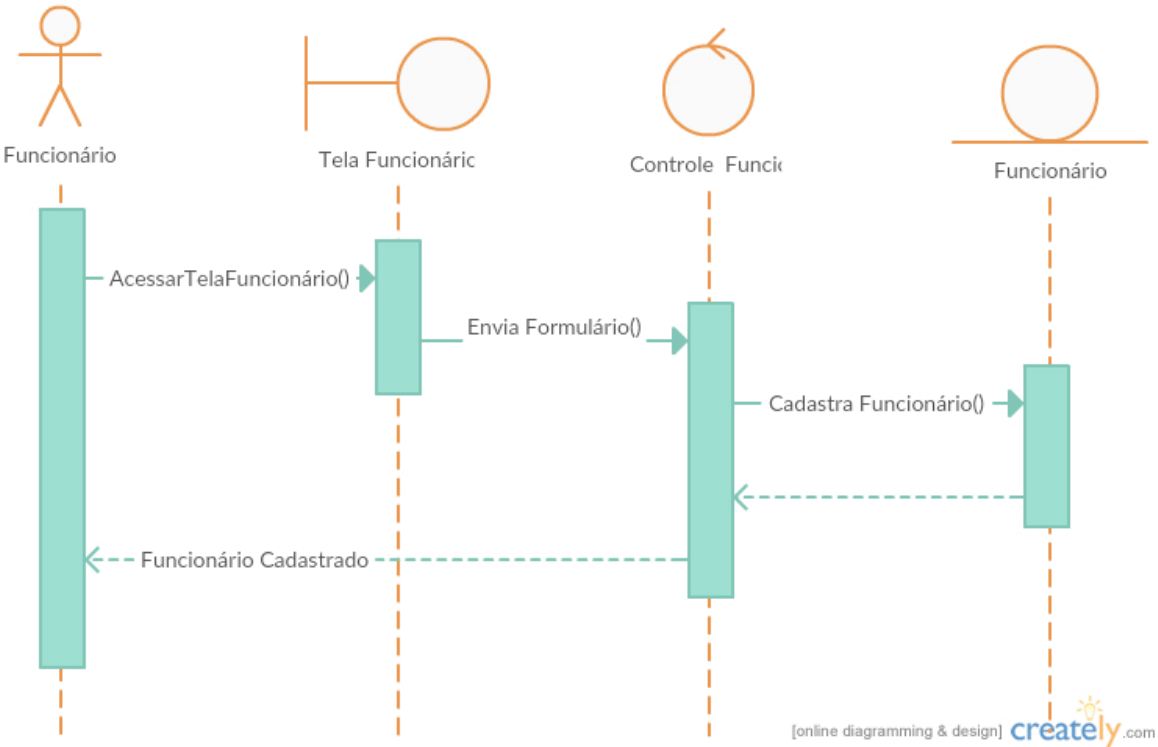


Figura 12: Cadastrar Funcionário

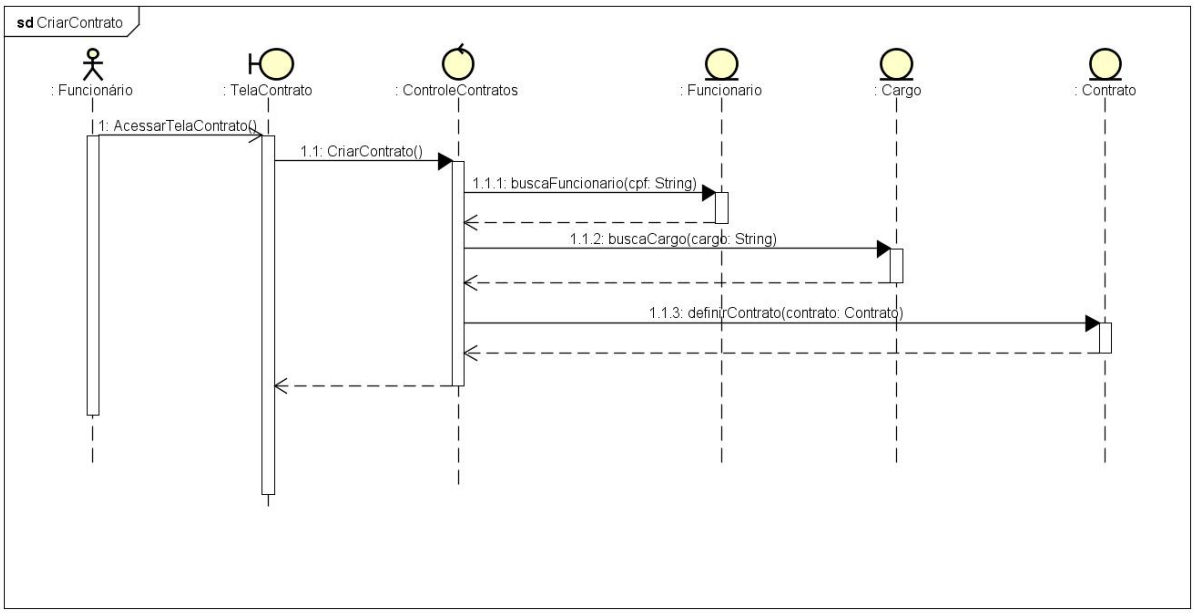


Figura 13: Criar Contrato

7.4 Diagrama de Casos de Uso

Figura 14: Diagrama de Casos de Uso