Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

THAIS DIZERO

APLICAÇÃO JAVA PARA GERENCIAMENTO DE LOCAÇÃO DE TRAJES SOCIAIS

Assis
2016

THAIS DIZERO

APLICAÇÃO JAVA PARA GERENCIAMENTO DE LOCAÇÃO DE TRAJES SOCIAIS

Projeto de pesquisa apresentado ao curso de Análise de sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Thais Dizero
Orientador: Prof. MSc. Guilherme de Cleva Farto

Assis
2016

FICHA CATALOGRÁFICA

DIZERO, Thais.
Aplicação Java para gerenciamento de locação de trajes sociais / Thais Dizero. Fundação Educacional do Município de Assis – FEMA – Assis, 2016.
Numero p.70

Orientador: Prof. MSc. Guilherme de Cleva Farto
Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA

1. Locação de Trajes Sociais. **2.** Java. **3.** JasperReports

CDD: 001.61
Biblioteca da FEMA

APLICAÇÃO JAVA PARA GERENCIAMENTO DE LOCAÇÃO DE TRAJES SOCIAIS

THAIS DIZERO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Prof. MSc. Guilherme de Cleva Farto

Analisador: Prof. Dr. Luiz Carlos Begosso

Assis
2016

DEDICATÓRIA

Dedico este trabalho para meus pais
que me deram todo o apoio e entenderam
a minha ausência em determinadas
circunstâncias, e a meus colegas que
me ajudaram a sempre seguir em frente.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais que me deram apoio e estímulo para realizar este sonho.
Ao orientador pela segurança passada em todo período de desenvolvimento deste trabalho, aos professores pelos conhecimentos passados.
E sem esquecer os colegas de classe pelos momentos de descontração e em especial ao Diego, João Roberto, Letícia, Sabrina entre outros que sempre estiveram ao meu lado me fazendo ter mais vontade de persistir e não desistir desse sonho.

RESUMO

Este projeto descreve o desenvolvimento de um sistema de gerenciamento de locação de trajes sociais, focando nos módulos de reservas e cadastro de clientes.

O objetivo do sistema é a diminuição do volume de papéis e tempo gasto na execução do serviço que é feito manualmente implementando assim uma aplicação em Java para melhorar o gerenciamento de uma loja de locação de trajes sociais, onde todo o controle era feito manualmente.

O projeto foi desenvolvido utilizando a linguagem *Java* adotando o padrão MVC, em conjunto com o *framework Hibernate*, utilizando a biblioteca JasperReports para a criação dos relatórios. Com isto, a empresa terá benefícios como rapidez para efetuar pesquisas e reservas, cadastro de produtos e de clientes bem como a informatização do controle financeiro.

O software proposto e desenvolvido para este trabalho visa melhorar o controle de informações com o intuito de diminuir o volume de papel que é utilizado com contratos e agendamentos feitos manualmente.

Palavras-chave: Desenvolvimento de aplicação *Desktop Java;* Locação de trajes sociais; Casamentos; Formaturas; Debutantes*.*

ABSTRACT

This project describes the development of a rental management system of social costumes, focusing on modules reserves and customer base.

The goal of the system is the reduction in the volume of paper and time spent performing the service that is done manually so implementing a Java-application to improve the management of a rental store social costumes, where all the control was done manually.

The project was developed using the Java language adopting the MVC pattern, together with the Hibernate framework, using the JasperReports library for creating reports. With this, the company will benefit as quickly to conduct research and reservation, product registration and customer as well as the computerization of financial control.

The proposed and developed software for this work is to improve the control information in order to reduce the volume of paper that is used with contracts and appointments made manually.

Keywords: Desktop Application Development Java; Leasing of social costumes; weddings; graduations; Debutantes.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 – INTRODUÇÃO

Empresas de pequeno porte que atuam na área de locação de trajes costumam ser simples, sendo assim, não utilizam de um software para fazer o controle da loja. O software Ltrajes tem como objetivo auxiliar no cadastro de produtos, clientes e na organização das reservas, evitando assim uma possível duplicidade de informações e atividades realizadas na empresa.

Conforme pesquisa da ABEOC BRASIL, Associação Brasileira de Empresas de Eventos, divulgada no final de maio de 2015, o mercado de eventos sociais cresceu nos últimos anos e estima-se que tenha atingido quase 17 bilhões de reais no ano de 2014 sendo a região sudeste a responsável por metade dos gastos com festas e cerimônias, totalizando 8,6 bilhões de reais (ABEOC BRASIL, 2015).

A indústria do casamento movimenta um custo médio de 35 mil reais por cerimônia, com maior movimento nos meses de abril e setembro e menor movimento nos meses de dezembro, janeiro e fevereiro devido às festas de fim de ano e o carnaval. Segundo a Pesquisa nacional de amostra de Domicilio (PNAD) e do Instituto Brasileiro de Geografia e Estatística (IBGE), o índice de casamento no Brasil está aumentando consideravelmente. Em 2008, foram registrados 959, 901 de casamentos, 5% a mais do que 2007. Já as debutantes rendem 4,3 bilhões para o país sendo a maioria de classe C com 50% no total, superando as da classe A/B em 16%. Entre os universitários, a massa de renda chega a 30 bilhões de reais. Somente na região metropolitana de São Paulo pode chegar a 4,4 bilhões de reais (ABEOC BRASIL, 2015).

O software desenvolvido para este trabalho visa melhorar o controle de informações diminuindo o volume de papel que é utilizado com contratos e agendamentos feitos manualmente. Este trabalho foi inspirado em uma empresa denominada como Rosa aluguel de trajes sociais, localizada na cidade de Andirá - PR que oferece produtos com qualidade priorizando o bom atendimento.

1.1 OBJETIVOS

O objetivo principal deste trabalho é implementar uma aplicação em Java que forneça funcionalidades para melhorar o gerenciamento de uma loja de locação de trajes sociais onde o controle é feito manualmente com o uso de papéis, sendo assim o processo de organização da empresa se tornará mais eficaz.

Atualmente, a organização e a execução de reservas são feitas por meio de papéis e manualmente com o auxílio de uma agenda contendo diversas informações como, dia, mês em que será utilizado o produto, nome do locatário, uma pequena descrição do produto e seu valor.

Como parte burocrática há um contrato que também é preenchido manualmente, nele é informado o nome, endereço, documentos do locatário, a descrição do produto e seu valor, a forma de pagamento, data de devolução, algumas regras que o locatário deve seguir, e a assinatura do locatário.

Com este software a empresa terá benefícios como rapidez para efetuar pesquisas e reservas, cadastros de produtos e de clientes e um controle financeiro informatizado.

O sistema é composto por cadastro de clientes, produtos, fornecedores e etc, uma pesquisa de clientes, produtos e reserva, onde será possível consultar quais produtos estão alugados, podendo ser consultado por código do produto, data da reserva ou nome do cliente, evitando assim o volume de papel que é armazenado na empresa e também o gasto com tais papéis.

1.2 JUSTIFICATIVAS

Com o aumento do fluxo de casamentos, festas de 15 anos e formaturas, empresas que atuam na área de locação de trajes se deparam com a falta de auxílio para a organização de seus serviços.

Antigamente a parte de organização de reservas era feita manualmente e agora por meio de software a empresa passará a ter facilitadores para está parte, gerando assim ganhos em todos os aspectos.

1.3 PÚBLICO ALVO

Com a elaboração deste software empresas que estão no mesmo nicho de mercado terá facilidade para controlar seus serviços. O software ajudará controlar todo o processo de entrada e saída de produtos da loja e cadastros de clientes mantendo assim mais organizado e seguro. Com isso o produto final deste projeto será oferecido como uma solução prática para diversas empresas que atuam neste ramo.

.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido em seis partes onde o Capítulo 1, apresenta a introdução, justificativas, objetivos com o intuito de mostrar a motivação para o desenvolvimento do trabalho. O Capítulo 2 destaca as tecnologias e ferramentas utilizadas para o desenvolvimento do projeto. O Capítulo 3 aborda toda a parte de modelagem do sistema para maior entendimento do funcionamento do software. No Capítulo 4 é possível visualizar o tempo de desenvolvimento do trabalho e orçamento dos equipamentos utilizados. O Capítulo 5 aborda as parte de interfaces e programação, destacando como o software está organizado, finalizando com o Capítulo 6 onde está a conclusão e sugestão de trabalhos futuros.

2 – TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

Neste capítulo serão apresentadas as ferramentas utilizadas para o desenvolvimento do sistema Ltrajes. A tecnologia Java com *Swing* fora escolhida para o desenvolvimento do software por ser uma plataforma e tecnologia orientada a objetos de alto nível.

Para o desenvolvimento fora escolhido a IDE Netbeans, pois possui código aberto e pode ser usado por qualquer desenvolvedor além de ser executado em várias plataformas tais como, *Windows, Linux, MacOS* e *Solaris* (GONÇALVES, 2008).

A IDE fornece ferramentas profissionais para a criação de aplicativos em *desktop, web* e *mobile*, seus principais recursos são: editor de código fonte integrado, com recursos para aplicações *Web*, visualizador de classes integrado ao de interfaces, suporte ao *Java Enterprise Edition* e suporte a *Database* (banco de dados) (GONÇALVES, 2008).

Para a camada de persistência, será adotado o *Postgresql* que é um banco de dados relacional *open source*. Um de seus atrativos é possuir recursos comuns a banco de dados de grande porte, fornecendo recursos, inclusive, para operações robustas. Além disso, trata-se de um banco de dados versátil, seguro e gratuito (MAYMALA, 2015).

Para o protótipo de interface fora escolhido a ferramenta de *designer Invision* onde basta fazer o *upload* do projeto e adicionar *hotspots* para transformar as telas estáticas em clicáveis (INVISIONAPP, 2015).

Segundo Gabardo (2012, p, 18), ”O *Model view controller (*MVC) é um padrão de designer de software e é utilizado para separar as camadas de lógica e negócios da de apresentação. Quando o MVC é referenciado como *designer* de software é porque o MVC vem sendo adotado como uma solução periódica para um problema conhecido.

Para testes unitários em Java fora escolhido o *framework* JUnit que é *open-source* criado por Eric Gamma e Kent Beck, a proposta de Gama e Beck era criar uma ferramenta que auxiliasse na produção de testes, e aumentaria a produtividade em relação à codificação dos testes. JUnit é um *framework* que facilita o desenvolvimento e execução de testes unitários em código Java orientado a objeto ele verifica se cada unidade de código funciona da forma esperada, facilita a criação, execução automática de testes e a apresentação de resultados. Esta ferramenta fornece uma completa API, conjunto de

classes, para construir os testes e aplicações gráficas e em modo console para executar os testes criados (MEDEIROS, 2009).

2.1 – JAVA

Java é uma plataforma orientada a objetos que foi criada como ferramenta de programação de um projeto de *Sun Microsystems* chamado "*The Green Project*", iniciado por Patrick Naughton e James Gosling em 1991. Em 1992, foi criada a primeira demonstração do projeto, um sistema executando em *handheld* com capacidade de controle remoto com interface *touchscreen* interativa. *Handheld* foi denominado como *7(*star seven*), por ser a forma de atender chamadas telefônicas entre os telefones dos integrantes da equipe (CLARO et al., 2008).

O *7 controlava uma grande variedade de dispositivos de uso doméstico apresentando uma interface com animação. O sistema criado para *handheld* foi executado em um novo processador independente de linguagem de programação que era chamada de *Oak* que do inglês significa carvalho. Em maio de 1995, *John Gage*, diretor da *Sun Microsystems*, e *Marc Andreessen*, executivo da *Netscape*, anunciaram o lançamento da plataforma Java. Seu *slogan* definido na forma de uma xícara de café originou-se por causa do consumo excessivo de café ingerido pela sua equipe de desenvolvimento. Tal plataforma foi então inserida no *Netscape* que era o principal *browser* de Internet da época. Capaz de criar aplicativos *desktops* e *Web*, Java cria software robustos, completos e independentes (MENDES, 2009).

2.1.1 – Principais características

A plataforma Java não incorpora o uso de herança múltipla, ponteiro entre outros conceitos que são encontrados em C e C++. Além disso, é considerada simples por permitir o desenvolvimento de softwares para diversos sistemas operacionais e arquitetura de hardware sem necessidade de se preocupar com detalhes de infraestrutura (CLARO et al., 2008).

O modelo de orientação a objetos existe desde 1970, porém só ganhou maior visibilidade depois do sucesso da linguagem Java. A linguagem Java foi criada seguindo os padrões da orientação a objetos, trazendo consigo os conceitos de herança, polimorfismo e encapsulamento. Diferente da programação estruturada a orientação a objetos adota uma forma mais próxima do mecanismo humano para lidar com a complexidade de um sistema (MENDES, 2009).

O conceito *de Multithread* permite projetar e implementar aplicações paralelas de forma eficiente. São instruções executadas dentro de um mesmo processo com comportamento em tempo real e maior capacidade de resposta (MENDES, 2009).

A linguagem Java interpretada gera um arquivo depois da complicação no formato *bytecode* podendo ser executado em várias arquiteturas como: *Windows*, *Linux*, *Mac* e *Unix*, que tenha a máquina virtual Java instalada. O formato *bytecode* é realizado no momento da execução que é gerenciada pela *Java Virtual Machine* (JVM) (CLARO et al., 2008).

Com independência de arquitetura a linguagem Java é projetada para sistemas implementados em plataformas de hardware e software diferentes. Com isso, o software pode ser executado em servidores como *UNIX* da *HP* e *UNIX* da *IBM*. O compilador da plataforma Java gera *bytecodes* permitindo assim que um programa Java seja executado e qualquer arquitetura (MENDES, 2009).

Java também incorpora um mecanismo que faz com que o programa não tenha contato com o computador real. O programa Java não tem acesso aos dispositivos de entrada e saída, memória e sistemas de arquivos, isso é feito pela *Java Virtual Machine* (JVM) tornando assim um sistema mais seguro (CLARO et al., 2008).

2.1.2 – Máquina Virtual Java

Máquina virtual é um software que simula uma máquina física e executa vários programas, gerencia processos, arquivos e memórias através de um software onde todo o processo é feito virtualmente independente do hardware que está sendo utilizado. O Java tem sua execução relacionada com o sistema operacional, sua ligação é direta com a JVM onde é feita a portabilidade do código. A JVM é desenvolvida em código nativo, pois

interage diretamente com o sistema operacional. A JVM é responsável pela execução de pilhas e gerenciamento de memória e *threads*. Possui uma função que é responsável pela limpeza de memória virtual chamado de *Garbage Collector.* O *Garbage Collector* deixa uma quantidade de lixo para que assim possa fazer coletas maiores, diminuindo o tempo gasto. O *Garbage Collector* é um dos serviços mais importantes da JVM, pois exige pouco esforço para usá-lo (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPAAT; KUNG, 2012).

Na JVM a área total usada para armazenar objetos é chamada de *Heap*, seu tamanho de memória é controlado pelas opções –xms e –xmx, onde o-xms especifica o tamanho inicial do *heaps* e o-xmx o tamanho máximo. (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPAAT; KUNG, 2012).

Para programar em Java é necessário dois componentes o *Java Development Kit* (JDK) que possui pacotes que possibilitam o desenvolvimento de aplicações Java, já possuindo a JVM, e o *Java Runtime Enviroment* (JRE) que é onde a JVM está contida (SILVEIRA; SILVEIRA; LOPES; MOREIRA; STEPAAT; KUNG, 2012).

Com isso percebemos que a *Java Virtual Machine* faz todo o trabalho livrando assim o desenvolvedor de se preocupar com imprevistos na aplicação.

2.2 – INTERFACES GRÁFICAS COM SWING

Java *Swing* foi criado em 1997 com componentes GUI que a partir da versão 1.2 da plataforma 2 se tornaram padrões. *Swing* é uma complementação do *Abstract Window Toolkit* (AWT). O AWT contém todas as classes para criação de interfaces do usuário.

Em 1996, foi criada pela *Netscape* a interface gráfica – *Internet Foundation Class* (IFC). Logo depois a *Sun* trabalhou junto para aperfeiçoar a interface onde passou a se chamar *Swing*, nome oficial até hoje das interfaces baseadas ponto a ponto. *Swing* faz parte do *Java Foundation Class* (JFC) que é vasto e contém muito mais coisas do que apenas o *Swing*, entretanto tem componentes que são mais lentos para serem exibidos do que os do AWT, mas isso não se torna um problema nas máquinas atuais (CLARO et al., 2008).

Já em questão de vantagens o *Swing* se destaca por conter maior e melhor conjunto de elementos para interface, menor dependência de plataforma, menos “*bugs*” e por ser robusto com mais características e maior portabilidade. É importante destacar que AWT e

Swing não devem ter seus componentes misturados, pois a manipulação de componentes dentro do AWT é diferente do *Swing*, por exemplo, no AWT um *frame* é adicionado utilizando o método ADD, já no *Swing* todos os componentes devem ser adicionados dentro de um *CONTENT PANE* (CLARO et al., 2008).

Com o uso do Java *Swing* API o posicionamento e tamanho dos componentes se tornam mais complicados para ser manipulado, isso acontece porque o controle é feito pelo *Layout Manager's* que é um objeto associado a um componente de *background*. Seus principais *Layouts Manager's* são os *Borderlayout* que divide a área de um componente de *background* em cinco regiões, cada região pode conter apenas um componente.

A figura 1 ilustra o *Borderlayout* que altera o tamanho de cada componente para torná-los compatíveis com cada região (SIERRA et al., 2005).

Figura 1: Borderlayout (DEITEL, 2010).

A figura 2 ilustra o *Flowlayout* que coloca os componentes da esquerda para direita não alterando o seu tamanho preferencial, quando o tamanho horizontal não for o suficiente automaticamente é inserido para próxima linha (SIERRA, Kathy; BATES, Bert, 2005).

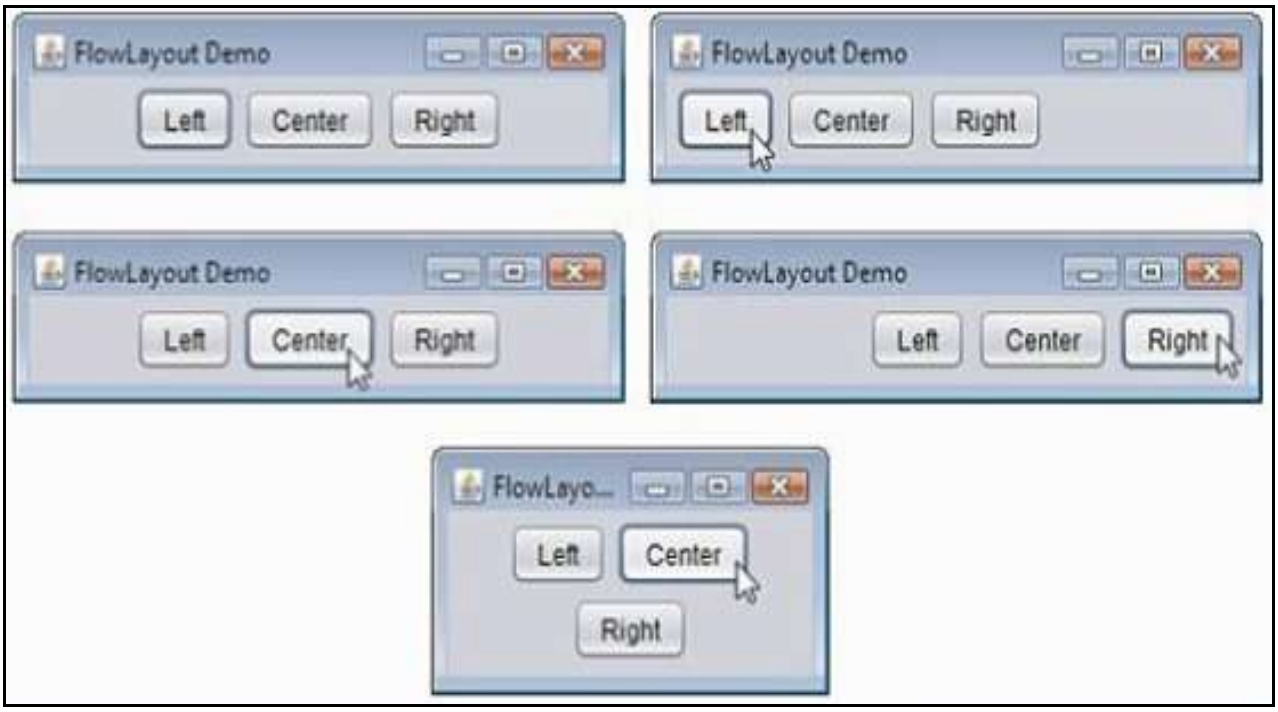


Figura 2: Flowlayout (DEITEL, 2010).

A figura 3 ilustra o *Gridlayout* que divide o *background* em células semelhando aos das tabelas (SIERRA, Kathy; BATES, Bert, 2005).

Figura 3: Gridlayout (DEITEL, 2010).

2.3 – JAVA DATABASE CONNECTIVITY (JDBC)

JDBC é uma API com acesso simples a várias aplicações Java que permite integração entre base de dados e aplicações Java, foram inspiradas no *Open Database Coonnectivity* (ODBC) que foi desenvolvido para acessar a base de dados do *Windows*. Com isso a JDBC API tem a pretensão de ser a mais simples para que assim possa oferecer mais flexibilidade aos programadores. Para a ligação com a base de dados é necessário ter um *driver* JDBC que é um conjunto de classes que fazem interface específica de base de dados, com vários *drivers* que suportam a maioria de bases de dados como: *Oracle*, *SQL Server*, *Sybase* e *Mysql* (THOMAS, 2002).

A JDBC manda instruções SQL para qualquer banco de dados relacional, porém tem que haver um *driver* para que isso seja possível. A JDBC conta com quatro tipos de *drivers* o tipo um JDBC-ODBC, tipo dois, tipo três e tipo quatro. O JDBC-ODBC foi o primeiro tipo, porém se tornou inutilizável por conter *drivers* nativos (CLARO et al.2008).

A figura 4 mostra que a aplicação Java tem duas camadas, a primeira ODBC e a segunda JDBC, caracterizada pela lentidão atualmente não é mais utilizada.

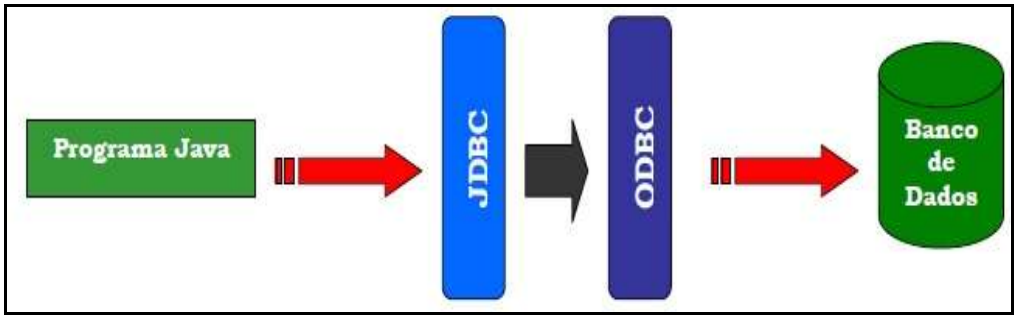


Figura 4: Exemplo do *Driver* tipo um JDBC- ODBC (CLARO; SOBRAL, 2008).

A figura 5 ilustra o *driver* tipo dois onde possui um *software* cliente do banco de dados em cada estação, isso significa que toda vem que um usuário *web* for executar a aplicação ele terá que baixar o cliente BD (CLARO et al., 2008).

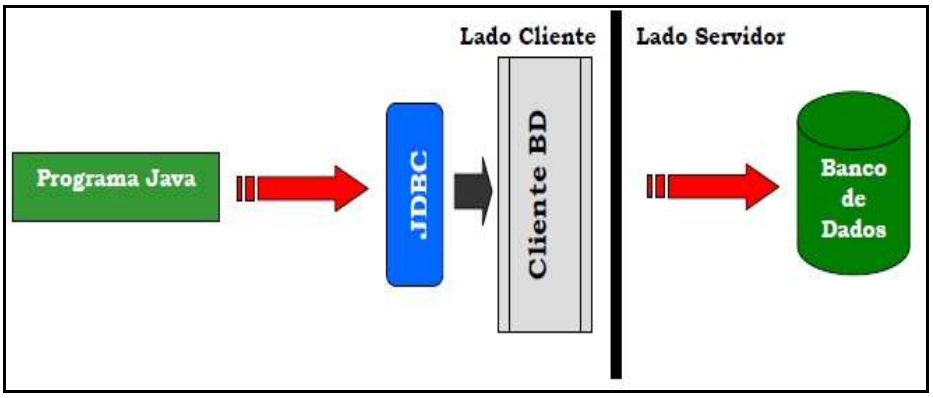


Figura 5: Exemplo do *Driver* Tipo dois (CLARO; SOBRAL, 2008).

A figura 6 ilustra o *driver* tipo três que utiliza uma classe onde cria uma porta de comunicação com o banco de dados, ou seja, cada cliente do Banco de Dados estará instalado no lado do servidor. (CLARO et al., 2008).

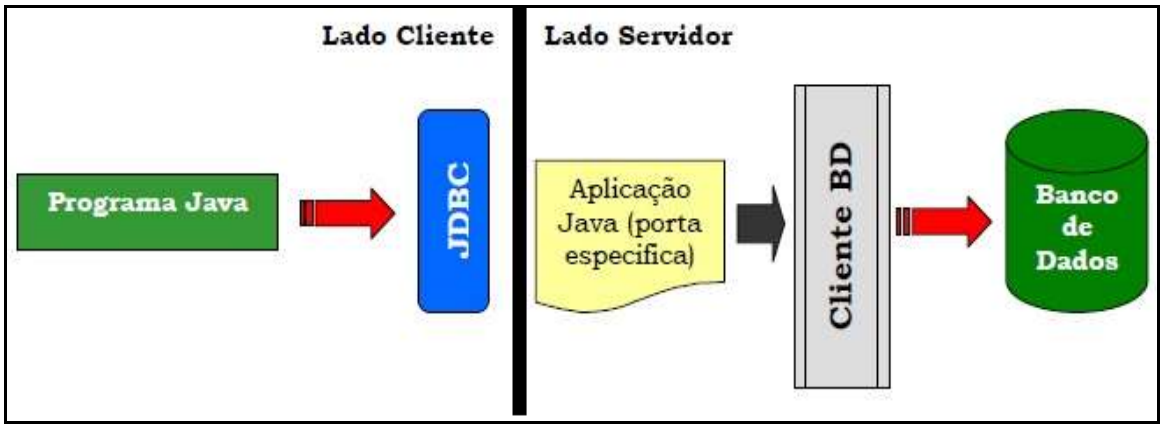


Figura 6: Exemplo do *Driver* tipo três (Thin Client) (CLARO; SOBRAL, 2008).

A figura 7 ilustra o *driver* tipo quatro, também chamando de *Thin Client,* contém um protocolo nativo onde o *driver* é puro Java, o acesso é feito diretamente ao banco de dados, não fazendo uso de um cliente BD (CLARO et al., 2008).

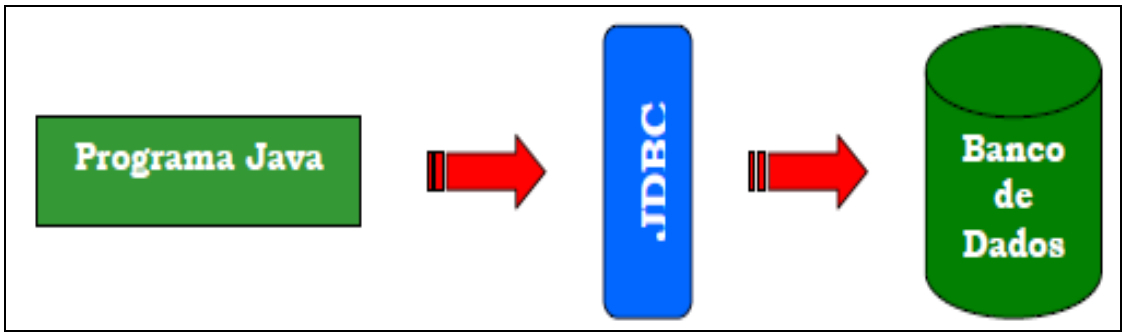


Figura 7: Exemplo do *Driver* tipo quatro (CLARO et al., 2008).

A figura 8 ilustra o JDBC onde existem classes de conexão com o BD com o método ***Class.forName (DriverClasseName)*** identifica qual *driver* está sendo utilizado e exibe o nome através de um URL específico, como segue um exemplo abaixo:

```
jdbc:<subprotocolo>:<subname>

onde podemos ter especificamente:

jdbc:odbc:JdbcOdbcDriver
```

Figura 8: Exemplo de URL (CLARO et al., 2008).

A figura 9 ilustra a classe *Drivermanager* onde é selecionado o *driver* apropriado caso exista mais de um. Para requisitar uma conexão com a base de dados é utilizado a URL, uma identificação de usuário, uma senha e o nome da instancia da base de dados, como segue:

```
jdbc:odbc:cursoJDBC,"","

ou em MySQL

jdbc:mysql://localhost/JAVADB
```

Nome da instância do BD

Figura 9: Exemplo de conexão (CLARO et al., 2008).

Para utilizar o JDBC é preciso seguir alguns passos importantes como:

- Carregar o *driver* JDBC usando o método *Class.forName*().

- Estabelecer a ligação à base de dados onde à única dificuldade é especificar o URL correto.
- Criar um objeto *statement* que envia comandos SQL para a base de dados.
- Processar os resultados.
- Fechar a ligação.

2.4 – BANCO DE DADOS POSTGRESQL

Em 1986, nasce na Universidade *Berkeley* nos (EUA) o projeto *Postgres* conhecido hoje como *Postgresql*, teve seu modelo e suas regras desenvolvidas por uma equipe orientada pelo professor *Michael Stonebrakes* com apoio de vários órgãos como *Army Research* (ARO) e o *National Science Foudation* (NSF) (MILANI, 2008).

Em 1987, a primeira versão demonstrativa ficou pronta e foi apresentada em diversos meios. A primeira versão estável foi lançada em 1989 com vários *bugs* que foram corrigidos anualmente com novas versões. Em 1991, teve seu código adquirido pela *Illustra Information Technologies* no qual se juntou com a *Informix* do professor *Michael Stonebraker* que foi comprada pela IBM por um bilhão de dólares em 2001(MILANI, 2008).

Com uma versão estável e confiável o *Postegresql* conta com recursos de banco de dados pagos e supre a necessidade de pequenas, médias e grandes aplicações sendo assim, seu banco de dados não tem limites de tamanho, sua única limitação é quanto ao *hardware* do computador onde serão armazenadas suas informações onde o limite Máximo é de 32 terabytes por tabela (MILANI, 2008).

Postgresql é compatível com as principais linguagens utilizadas por programadores tais como: C/C++, JAVA/JSA, PHP, ASP. NET entre outros. Extremamente portável pode ser instalado nos seguintes sistemas operacionais: *Linux, Unix, Mac, Windows* (MILANI, 2008).

Postgresql contém diversos recursos por ser um SGBD relacional completo com suporte a operações atomicidade, consistência, isolamento e durabilidade (ACID), tem vantagem sobre outros bancos de dados, pois oferece os recursos necessários para realizar a replicação entre servidores sendo totalmente gratuito. O *Postgresql* é disponível para a

utilização de *cluster* que significa basicamente usar dois ou mais computadores configurados tornando assim maior sua capacidade de armazenamento e utilização. Por conter o recurso *multithread* é possível gerenciar várias conexões com o banco de dados, com isso o acesso à mesma informação pode ser feito por mais de uma pessoa sem que haja atrasos ou fila de espera.

No *Postgresql* já está incluído o suporte para SSL, que possibilita criar conexões seguras para movimentar informações sigilosas, havendo também uma extensibilidade para utilizar criptografia como o SHA1 e o MD5. *Postgresql* adota os padrões impostos pela ANSI SQL em suas implementações de funcionalidades. Usando a versão de 2003 do ANSI SQL os novos recursos estão sendo desenvolvidos em conformidade com as versões 92 e 99 do ANSI SQL. *Postgresql* foi pioneiro na implantação de recursos (GIN e DTRACE) fazendo se destacar diante dos demais bancos de dados (MILANI, 2008).

O *Postgresql* suporta grandes tamanhos de informações em suas tabelas, onde o tamanho máximo de um banco de dados é ilimitado, o tamanho de uma tabela é o equivalente a 32 terabytes, com tamanho máximo de linhas de 1,6 terabytes, e 1 gigabytes de campo, com tamanho máximo de linhas e índices por tabela ilimitado e tamanho máximo de colunas por tabela de 250 a 1600 (MILANI, 2008).

2.5 – RELATÓRIO – JASPERREPORTS

Relatório é um conjunto de informações que mostra os resultados obtidos por uma determinada atividade, no sistema Ltrajes foi utilizado o gerador de relatório iReport que é uma IDE de código aberto escrito 100% em Java, juntamente com a biblioteca JasperReports que possui diversos formatos tais como PDF, HTML, XML, etc (MORAIS, 2016).

JasperReports fora escolhida por ser a ferramenta mais utilizada atualmente, por ser *open Source* e gratuita. Seu *layout* é definido em um arquivo XML com extensão .jrxml, a compilação do arquivo XML é feito através do método compileReport que é encontrado na classe net.sf.jasperreports.engine.JasperCompileManager. onde possui todas as informações do relatório (GONÇALVES, 2009).

3 – ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Neste capítulo serão mostradas as ferramentas que foram utilizadas para o levantamento de requisitos, análise e a modelagem de diagramas para demonstrar melhor a aplicação Java para gerenciamento de locação de trajes sociais Ltrajes.

3.1 – MAPA MENTAL

Para melhor abstração do sistema fora desenvolvido um Mapa Mental ou *Mind Map,* um diagrama responsável por facilitar o entendimento de uma ideia lógica assim como organizar tópicos relacionados a partir de um determinado foco inicial (SOUZA, 2004).

A Figura 10 ilustra as atividades do sistema Ltrajes:

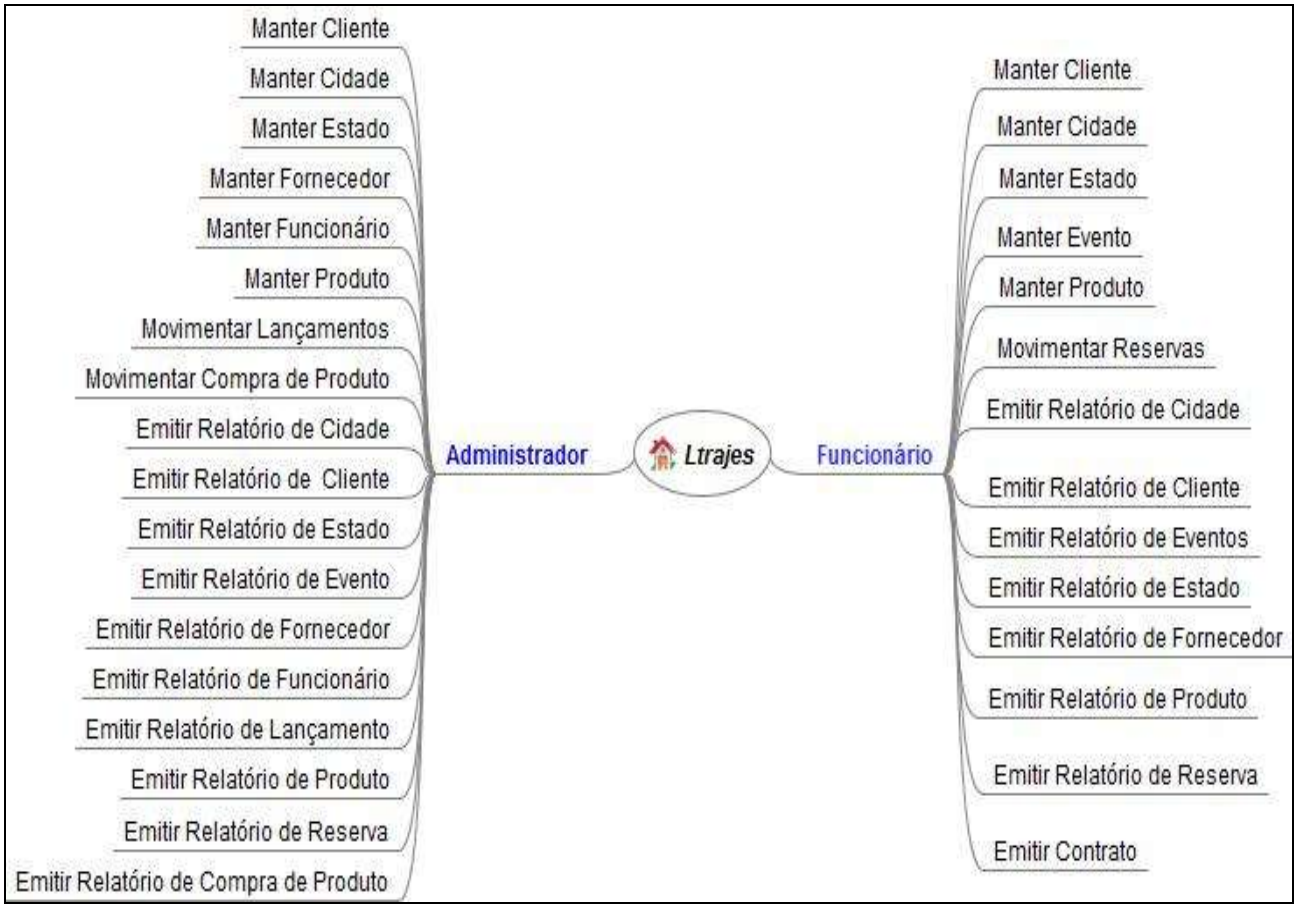


Figura 10: Mapa Mental – Aplicação LTRAJES.

3.2 – LISTA DE REQUISITOS

Entre diversas funcionalidades da aplicação, destacam-se os seguintes requisitos:

- Efetuar Login;
- Manter Cliente;
- Manter Cidade;
- Manter Estado;
- Manter Evento;
- Manter Funcionário;
- Manter Fornecedor;
- Manter Produto;
- Movimentar Lançamento;
- Movimentar Compra de Produto;
- Movimentar Reserva;
- Emitir Relatório de Cliente;
- Emitir Relatório de Cidade;
- Emitir Relatório de Estado;
- Emitir Relatório de Evento;
- Emitir Relatório de Evento por tipo;
- Emitir Relatório de Evento por cidade;
- Emitir Relatório de Funcionário;
- Emitir Relatório de Fornecedor;
- Emitir Relatório de Produto;
- Emitir Relatório de Produto por descrição;
- Emitir Relatório de Produto por fornecedor;

- Emitir Relatório de Reservas;
- Emitir Relatório de Reservas por data;
- Emitir Relatório de Lançamento por tipo;
- Emitir Relatório de Lançamento por data;
- Emitir Relatório de Compras de Produtos;
- Emitir Relatório de Compras de Produtos por data;
- Emitir Contrato;

3.3 – DIAGRAMA E ESPECIFICAÇÃO DE CASOS DE USO

Para melhor visualização dos requisitos da aplicação foram elaborados alguns diagramas como complemento da documentação e que descrevem a interação do sistema com o usuário. Como elemento primário da UML, utiliza-se o Diagrama de Casos de Uso para especificar o comportamento de um sistema ou de parte de um sistema. Para melhor descrever as funcionalidades da aplicação, serão utilizadas narrativas para especificar os diagramas descrevendo em forma textual a interação (GÓES, 2014).

A Figura 11 apresenta o Diagrama de Caso de Uso para a entidade “Administrador”:

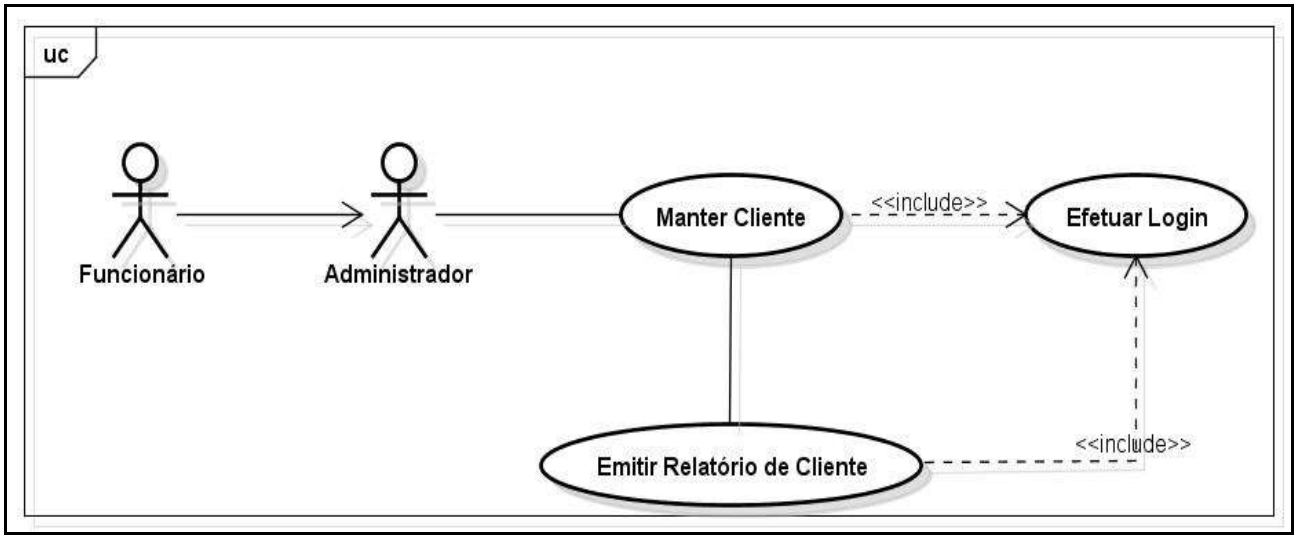


Figura 11: Diagrama de Caso De Uso – Administrador e funcionário.

A Figura 12 apresenta o Diagrama de Caso de Uso comum para as entidades “Administrador” e “Funcionário” com representação “Manter”:

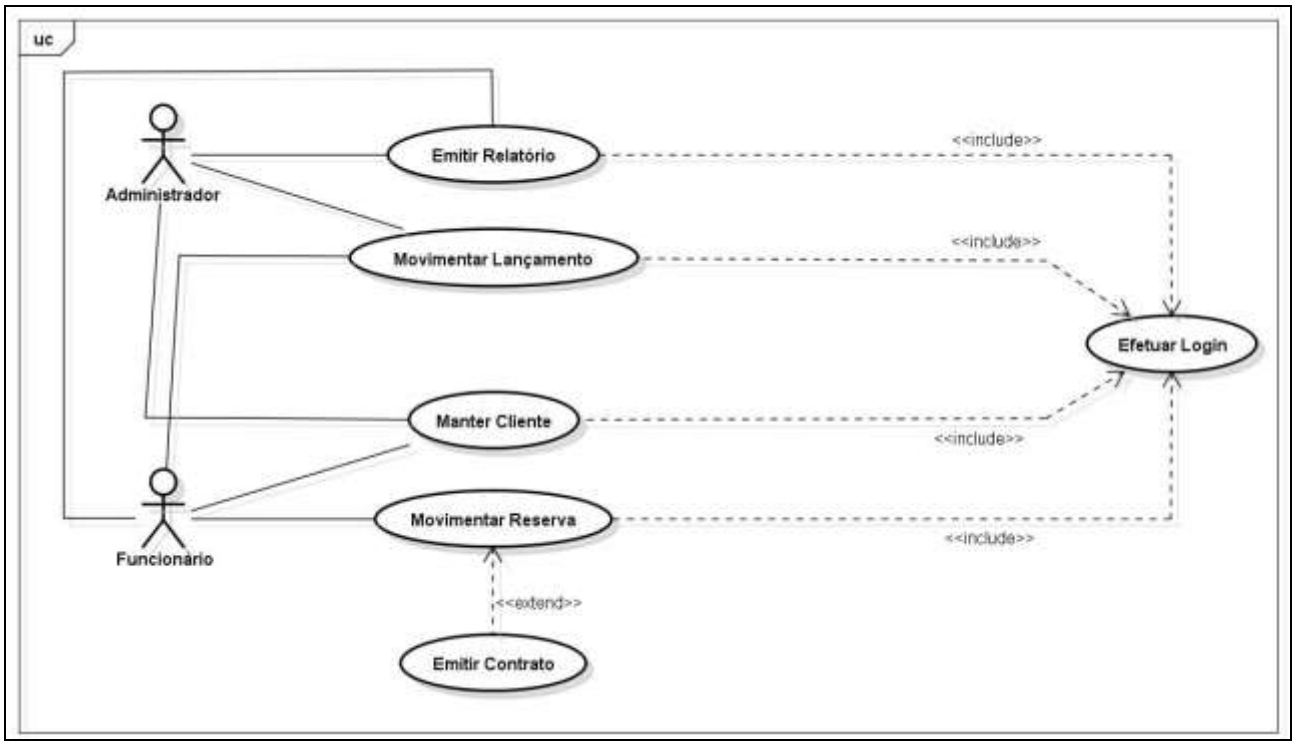


Figura 12: Diagrama de Caso de Uso – Administrador e Funcionário.

A Figura 13 apresenta o Diagrama de Caso de Uso comum para as entidades “Administrador” e “Funcionário” com representação “Movimentar e Emitir”:

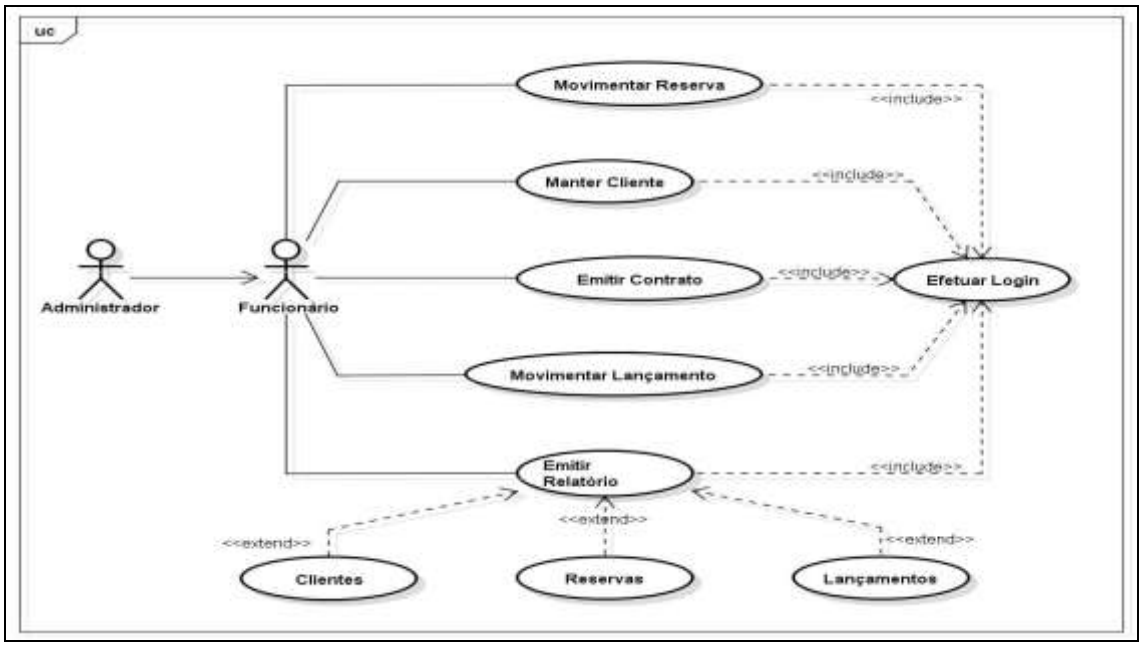


Figura 13: Diagrama de Caso de Uso comum – Administrador e Funcionário.

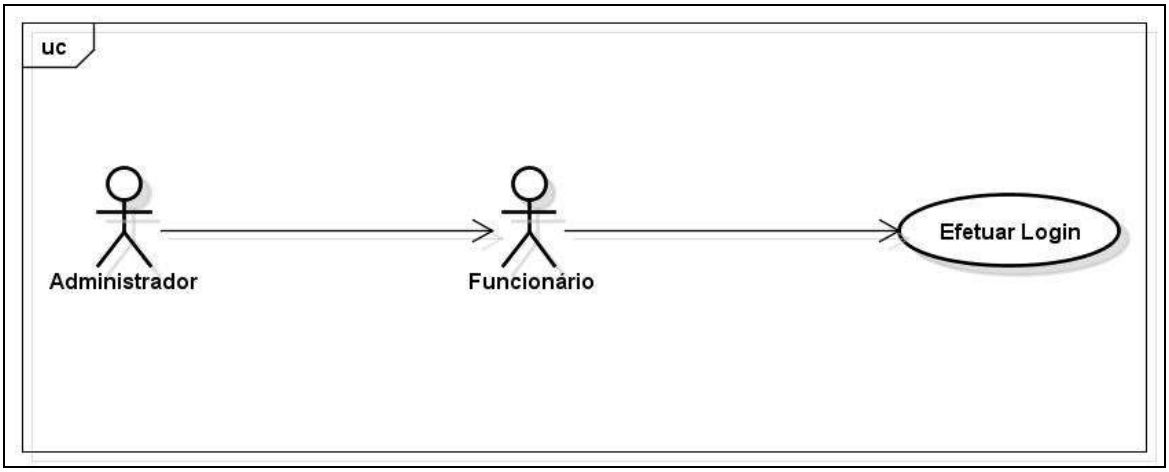


Figura 14: Diagrama de Caso de Uso Efetuar Login.

Nome do Caso de uso	Efetuar Login
Atores	Administrador e Funcionário
Pré-Condições	Não existe
Cenário Principal	1- O sistema solicita os dados para efetuar login. 2- O funcionário ou administrador informa os dados. 3- O funcionário ou administrador clica em acessar. 4- O sistema faz a validação dos dados.

	5- O funcionário se conecta no sistema.
Cenário Alternativo	Não existe
Casos de Teste	1- Caso os dados estejam corretos o sistema entra na tela principal. 2- Caso os dados não estejam corretos o sistema mostra uma mensagem “Usuário ou senha Inválidos”.

Tabela 1: Efetuar Login

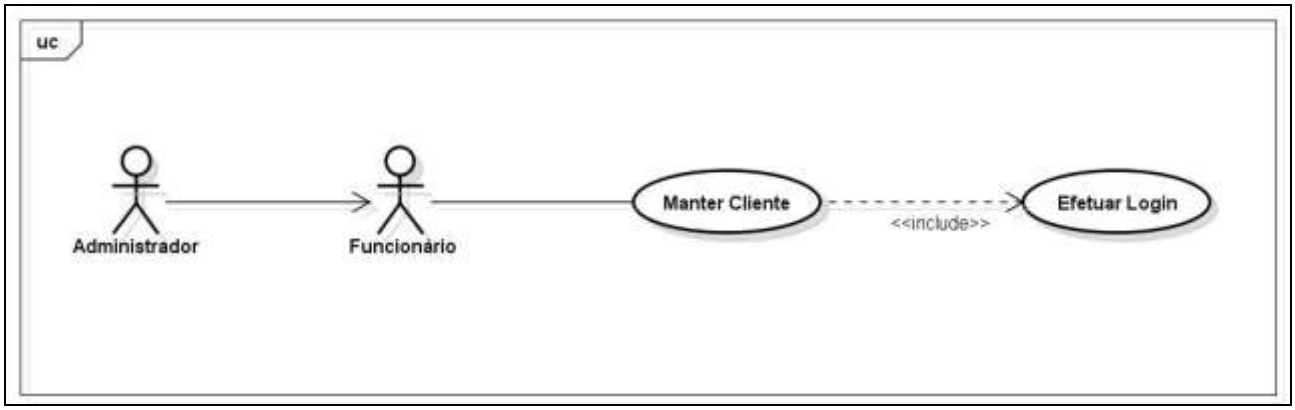


Figura 15: Diagrama de Caso de Uso Manter Cliente.

Nome do Caso de uso	Manter Cliente
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- Funcionário ou administrador informa os dados do cliente. 2- O sistema verifica se não há campo vazio. 3- O sistema verifica se já existe cadastro com o mesmo cpf. 4- O sistema cadastra o cliente.
Cenário Alternativo	A1- Funcionário ou administrador pode cancelar a operação durante o processo.
Casos de Teste	1- Caso tenha campos vazios o sistema informa qual campo está vazio. 2- Caso o CPF esteja cadastrado sistema mostra uma mensagem “Cliente já cadastrado”.

Tabela 2: Manter Cliente

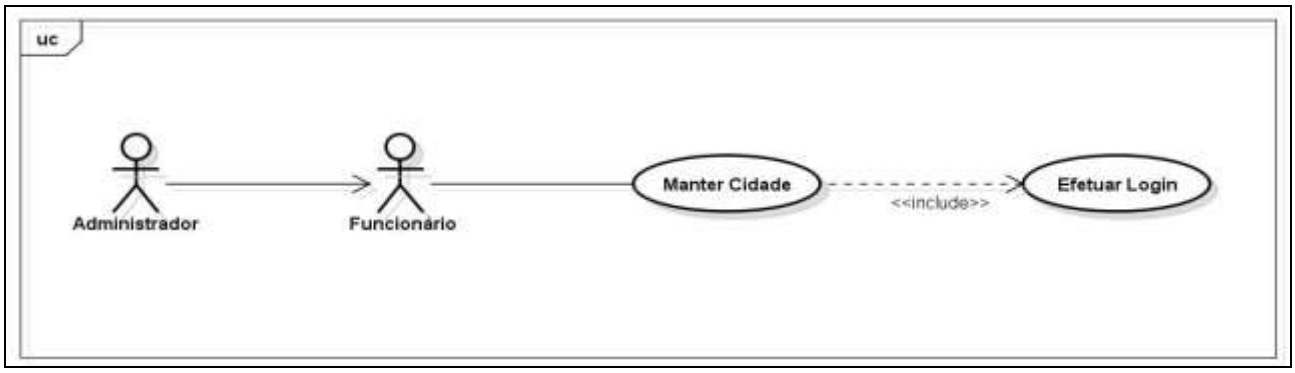


Figura 16: Diagrama de Caso de Uso Manter Cidade.

Nome do Caso de uso	Manter Cidade
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- Funcionário ou administrador Informa os dados da cidade. 2- O sistema verifica se há campos em branco. 3- O sistema verifica se já está cadastrada a cidade. 4- O sistema cadastra a cidade.
Cenário Alternativo	A1- Funcionário ou administrador pode cancelar a operação durante o processo
Casos de Teste	1- Caso os campos estejam vazios o sistema mostra qual campo deve ser preenchido. 2- Caso o nome já esteja cadastrado o sistema mostra uma mensagem “Cidade já cadastrada.

Tabela 3: Manter Cidades.

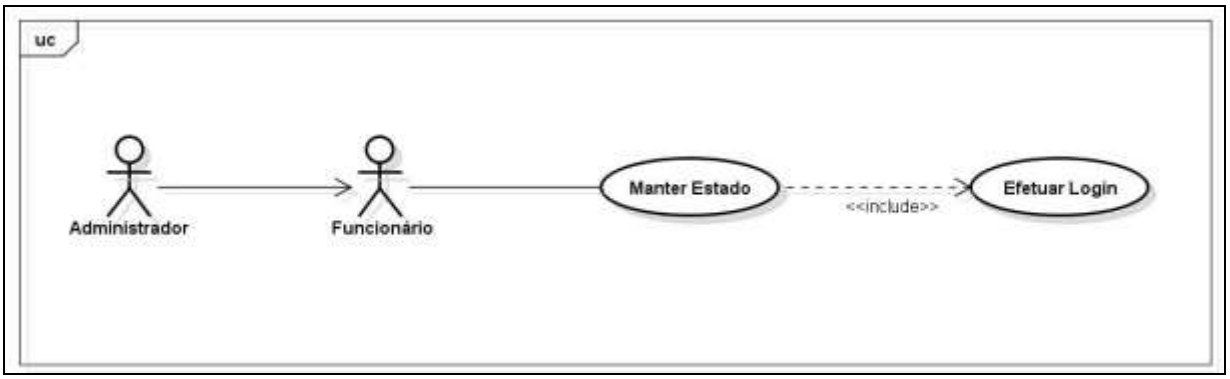


Figura 17: Diagrama de Caso de Uso Manter Estado.

Nome do Caso de uso	Manter Estado
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- Funcionário ou administrador informa os dados do estado. 2- O sistema verifica se há campos em branco. 3- O sistema verifica se já existe o estado cadastrado. 4- O sistema cadastra o estado.
Cenário Alternativo	A1- Funcionário ou administrador pode cancelar a operação durante o processo
Casos de Teste	1- Caso haja campos em branco o sistema informa qual campo deve ser preenchido. 2- Caso a sigla já esteja cadastrada o sistema mostra uma mensagem “UF já cadastrado”.

Tabela 4: Manter Estado.

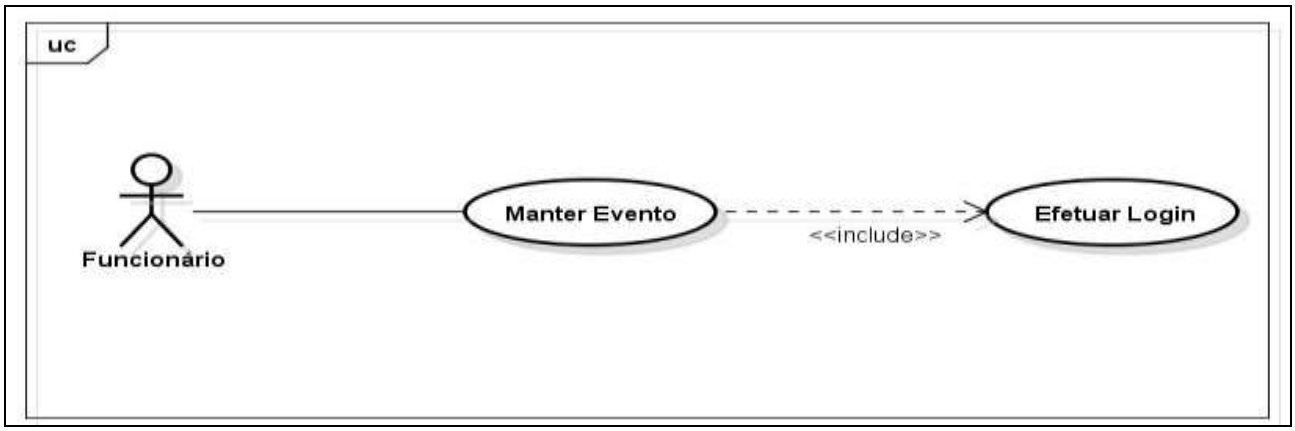


Figura 18: Diagrama de Caso de Uso Manter Evento.

Nome do Caso de uso	Manter Evento
Atores	Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- Funcionário informa os dados do evento. 2- Verifica se não há campos em branco. 3- O sistema verifica se o evento já foi cadastrado. 4- O sistema cadastra o evento.

Cenário Alternativo	A1- Funcionário pode cancelar a operação durante o processo.
Casos de Teste	1- Caso haja campos em branco o sistema mostra qual deve ser preenchido. 2- Caso o evento já esteja cadastrado o sistema emiti uma mensagem "Evento já cadastrado".

Tabela 5: Manter Evento.

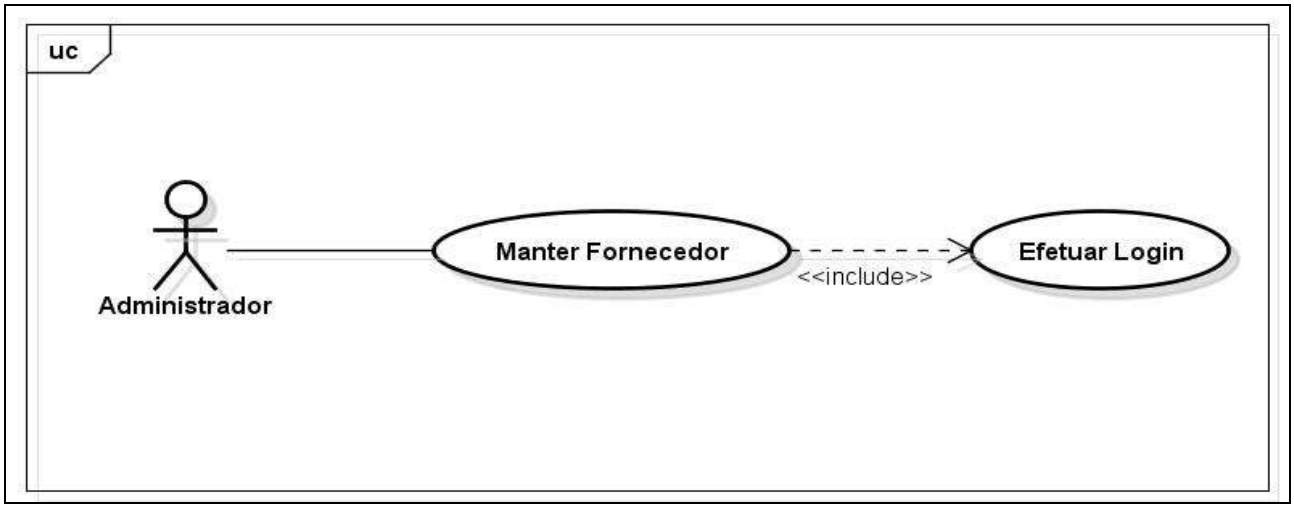


Figura 19: Diagrama de Caso de Uso Manter Fornecedor.

Nome do Caso de uso	Manter Fornecedor
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1- Administrador informa os dados do fornecedor. 2- O sistema verifica se há campos em branco. 3- O sistema verifica se o cnpj informado já existe. 4- O sistema cadastra o fornecedor.
Cenário Alternativo	A1- Administrador pode cancelar a operação durante o processo
Casos de Teste	1- Caso os campos estejam em branco, o sistema mostra qual campo deve ser preenchido. 2- Caso o cnpj já esteja cadastrado, o sistema emiti uma mensagem "Fornecedor já cadastrado".

Tabela 6: Manter Fornecedor.

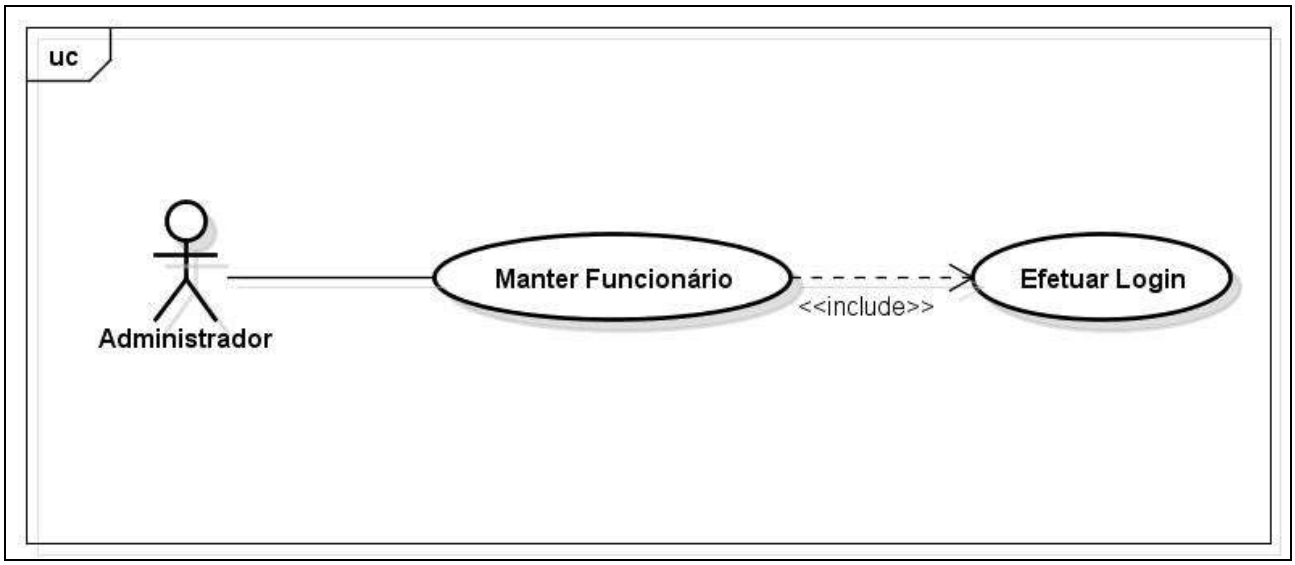


Figura 20: Diagrama de Caso de Uso Manter Funcionário.

Nome do Caso de uso	Manter Funcionário
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1- Administrador informa os dados do funcionário. 2- O sistema verifica se não há campos em branco. 3- O sistema verifica se já existe os dados no banco de dados. 4- O sistema cadastra o funcionário.
Cenário Alternativo	A1- Administrador pode cancelar a operação durante o processo
Casos de Teste	1- Caso haja campos em branco o sistema informa qual campo deve ser preenchido. 2- Caso os dados já existam o sistema informa que o funcionário já está cadastrado. 3- Caso os campos informados estejam corretos finaliza a operação.

Tabela 7: Manter Funcionário.

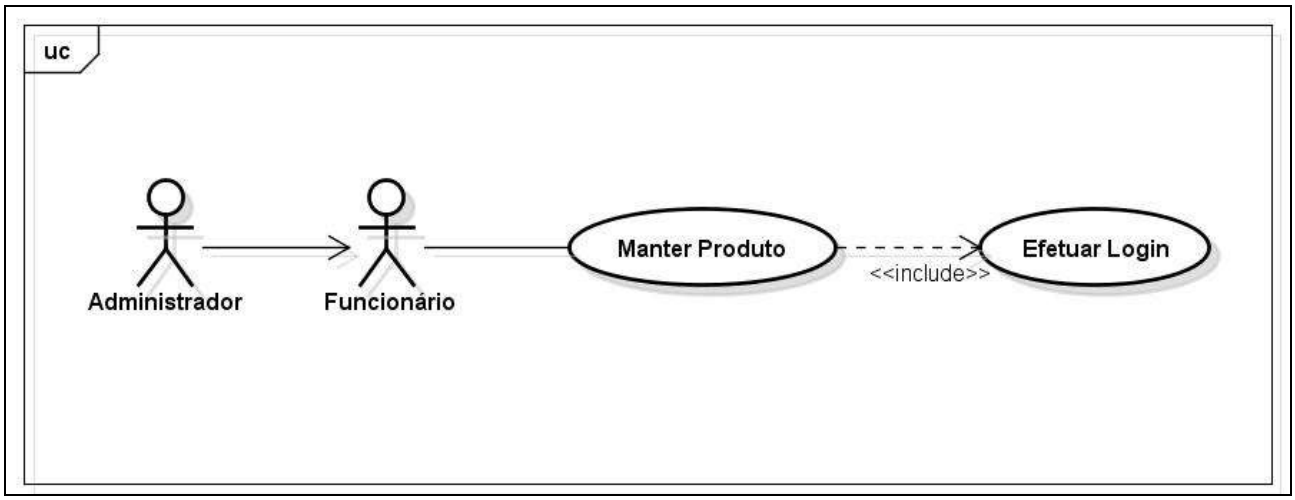


Figura 21: Diagrama de Caso de Uso Manter Produto.

Nome do Caso de uso	Manter Produto
Atores	Administrador e Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- Funcionário ou administrador informa os dados do produto. 2- O sistema verifica se não há campos em branco. 3- O sistema verifica se o produto já não está cadastrado. 4- O sistema cadastra o produto.
Cenário Alternativo	A1- Funcionário ou administrador pode cancelar a operação durante o processo.
Casos de Teste	1- Caso haja algum campo em branco, o sistema informa qual campos deve ser preenchido. 2- Caso o produto já exista o sistema mostra uma mensagem “Produto já cadastrado”.

Tabela 8: Manter Produto.

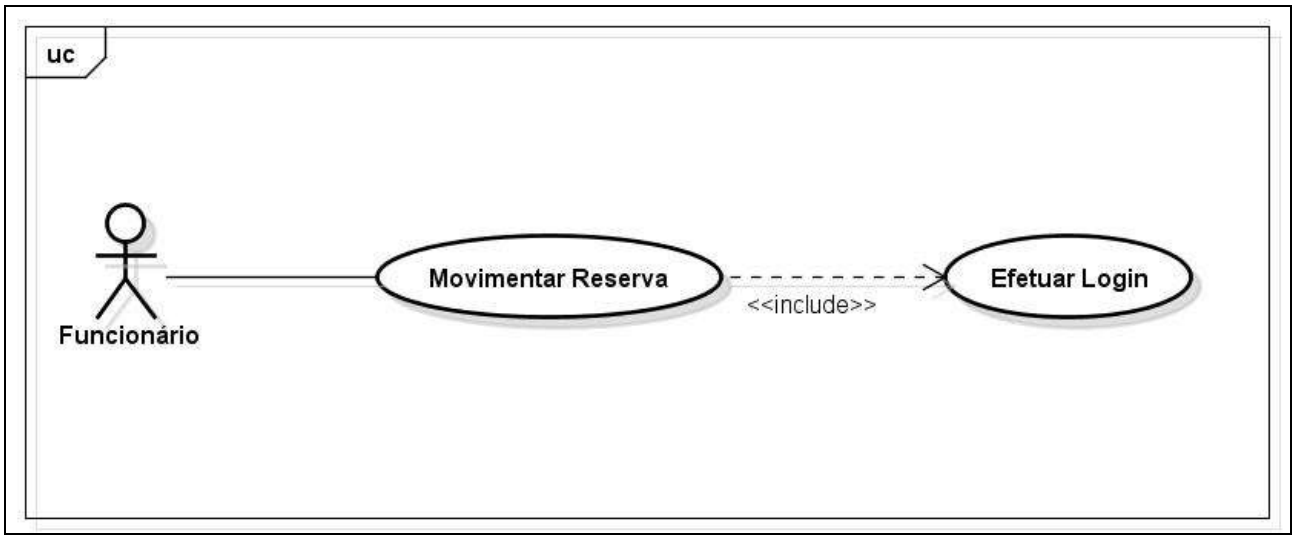


Figura 22: Diagrama de Caso de Uso Movimentar Reserva.

Nome do Caso de uso	Movimentar Reserva
Atores	Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O funcionário informa os dados da reserva. 2- O sistema faz a validação das informações. 3- O sistema efetua a reserva.
Cenário Alternativo	A1- O funcionário pode interromper o processo a qualquer momento clicando no botão “Cancelar”.
Casos de Teste	1- Caso o cliente não esteja cadastrado, o funcionário pode cadastrar-lo clicando em “Cadastrar”.

Tabela 9: Movimentar Reserva.

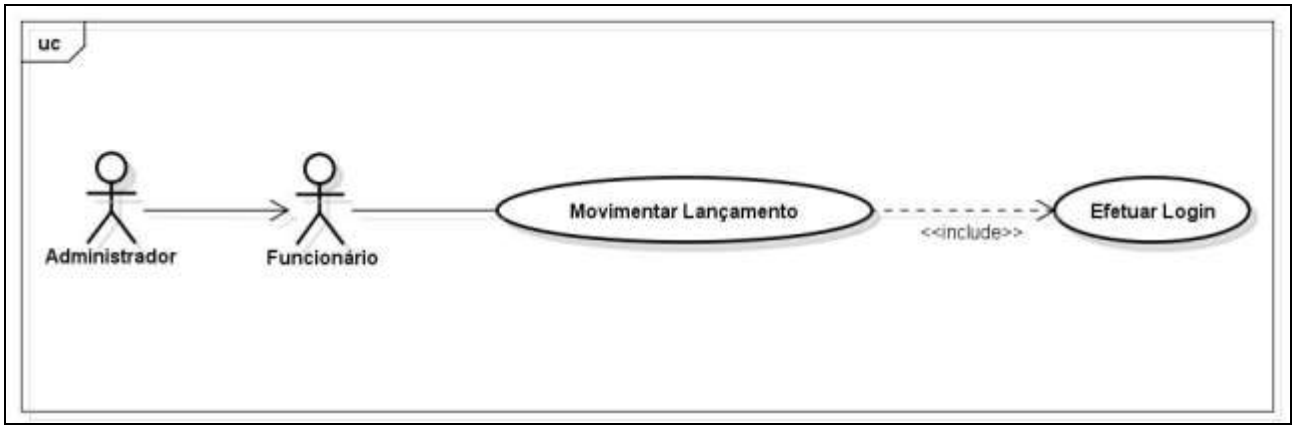


Figura 23: Diagrama de Caso de Uso Movimentar Lançamento.

Nome do Caso de uso	Movimentar Lançamento
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1- Administrador seleciona a opção Movimentações. 2- No submenu seleciona lançamentos. 3- O sistema mostra em abas as contas a pagar e receber. 4- O lançamento pode ser feito atravez do botão “Lançar Conta”. 5- O sistema pede as informações necessarias para realizar o lançamento.
Cenário Alternativo	A1- O administrador pode interromper o processo a qualquer momento clicando no botão fechar. A2- Se clicado em “limpar” o sistema limpa os campos.
Caso de Teste	1- O sistema pede para informar se é contas a receber ou a pagar.

Tabela 10: Movimentar Lançamento.

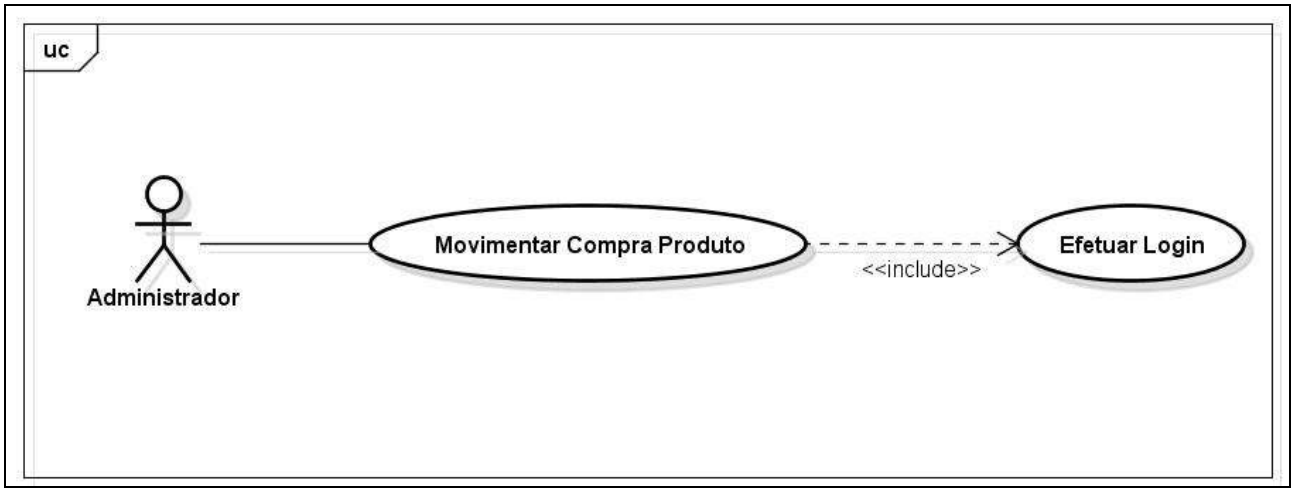


Figura 24: Diagrama de Caso de Uso Movimentar Compra de Produto.

Nome do Caso de uso	Movimentar Compra de Produto
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1- Administrador seleciona a opção Movimentações. 2- No submenu seleciona movimentações. 3- No proximo submenu seleciona

	realizar compra de produto. 4- O sistema pede as informações necessarias para realizar a compra.
Cenário Alternativo	A1- O Administrador pode interromper o processo a qualquer momento clicando no botão cancelar.
Caso de Teste	1- O administrador deixa de informar algum campo obrigatório.

Tabela 11: Movimentar Compra de Produto.

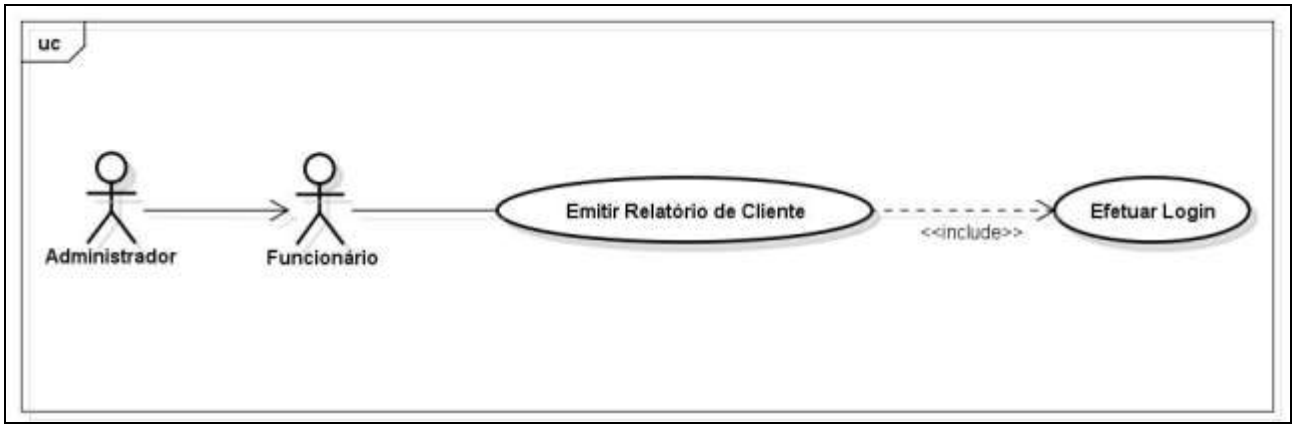


Figura 25: Diagrama de Caso de Uso Emitir Relatório de Clientes.

Nome do Caso de uso	Emitir relatório de Clientes
Atores	Administrador e Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O sistema informa os dados necessarios para relatório. 2- O administrador ou funcionário seleciona a opção “visualizar” 3- O administrador ou funcionário seleciona a opção “imprimir”. 4- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador ou funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador ou funcionário cancela a operação.

Tabela 12: Emitir Relatório de Clientes.

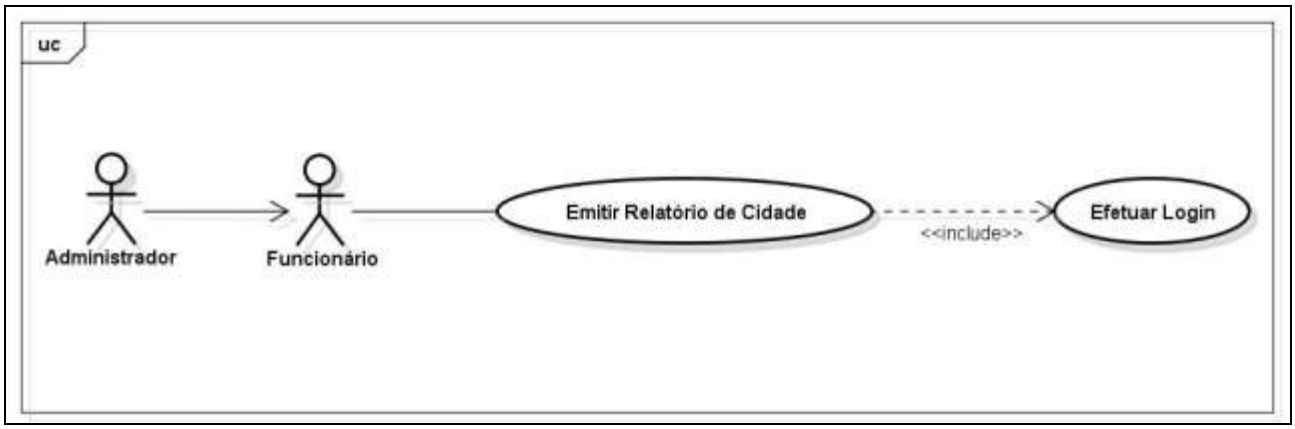


Figura 26: Diagrama de Caso de Uso Emitir Relatório de Cidade.

Nome do Caso de uso	Emitir relatório de Cidade
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O sistema informa os dados necessarios para relatório. 2- O administrador ou funcionário seleciona a opção “visualizar”. 3- O administrador ou funcionário seleciona a opção “imprimir”. 4- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador ou funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador ou funcionário cancela a operação.

Tabela 13: Emitir Relatório de Cidade.

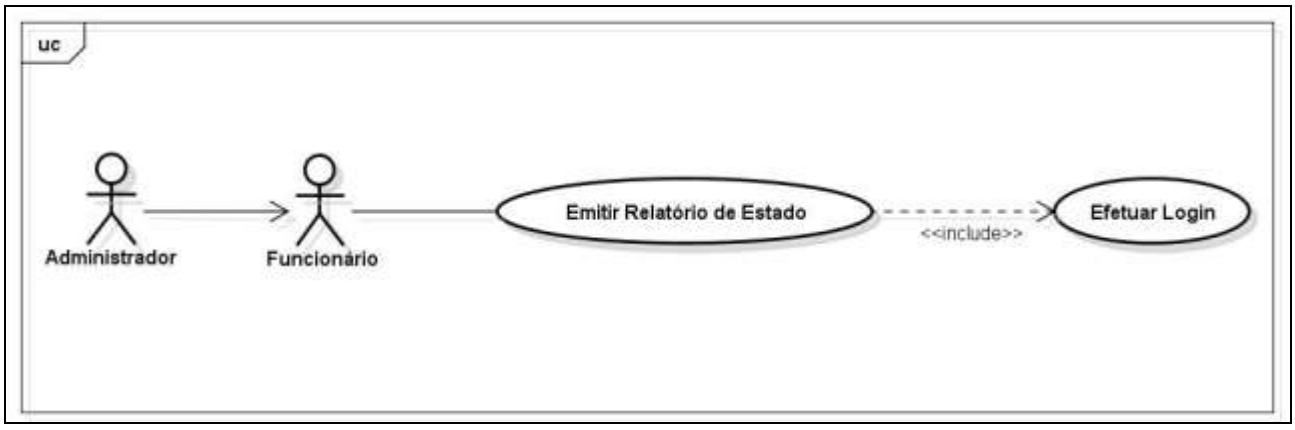


Figura 27: Diagrama de Caso de Uso Emitir Relatório de Estado.

Nome do Caso de uso	Emitir relatório de Estado
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O sistema informa os dados necessarios para relatório. 2- O administrador ou funcionário seleciona a opção “visualizar”. 3- O administrador ou funcionário seleciona a opção “imprimir”. 4- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador ou funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador ou funcionário cancela a operação.

Tabela 14: Emitir Relatório de Estado.

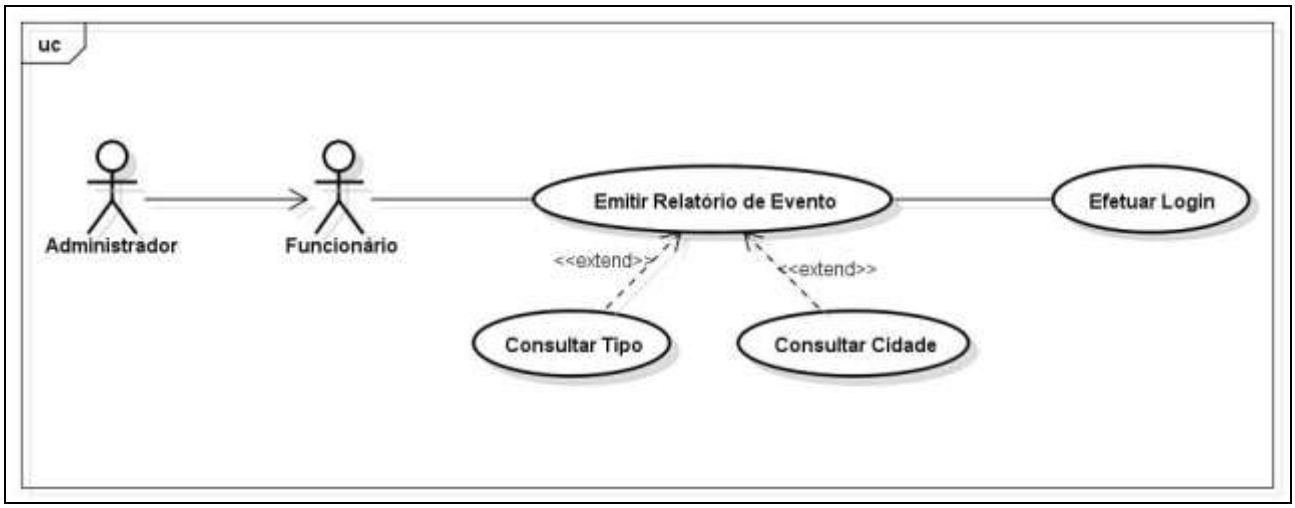


Figura 28: Diagrama de Caso de Uso Emitir Relatório de Evento.

Nome do Caso de uso	Emitir relatório de Evento
Atores	Administrador e Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O administrador ou funcionário pode escolher entre emitir relatório de todos os registros ou filtra-los. 2- Ao clicar em relatório por filtro o administrador ou funcionário escolhe qual tipo de consulta que deseja realizar e preenche os dados e clica em buscar. 3- O sistema preenche o relatório e

	mostra na tela. 4- O administrador ou funcionário seleciona a opção “imprimir”. 5- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador ou funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador ou funcionário cancela a operação.

Tabela 15: Emitir Relatório de Evento.

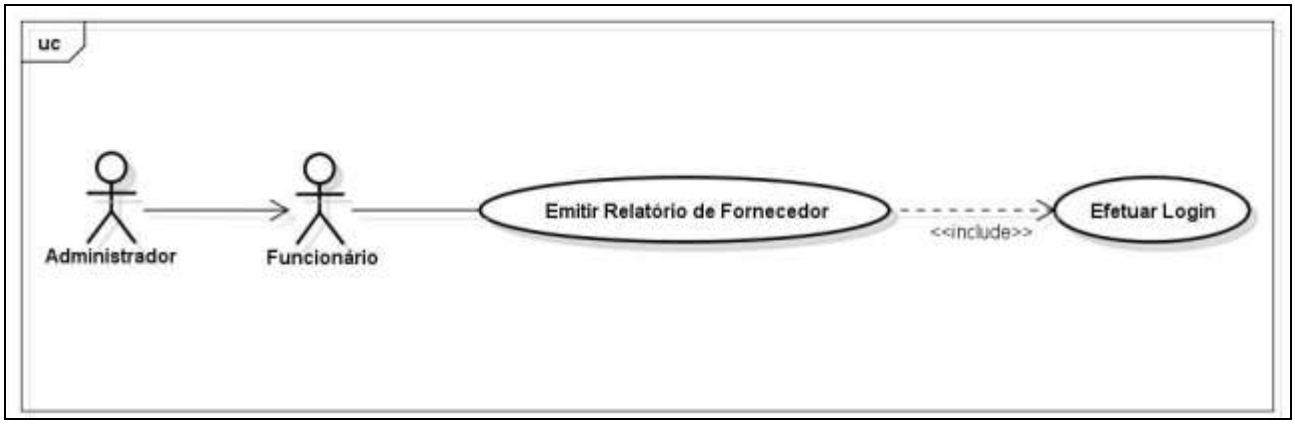


Figura 29: Diagrama de Caso de Uso Emitir Relatório de Fornecedor.

Nome do Caso de uso	Emitir relatório de Fornecedor
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O sistema informa os dados necessarios para relatório. 2- O administrador ou funcionário seleciona a opção “visualizar”. 3- O administrador ou funcionário seleciona a opção “imprimir”. 4- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador ou funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador ou funcionario podem cancela a operação.

Tabela 16: Emitir Relatório de Fornecedor.

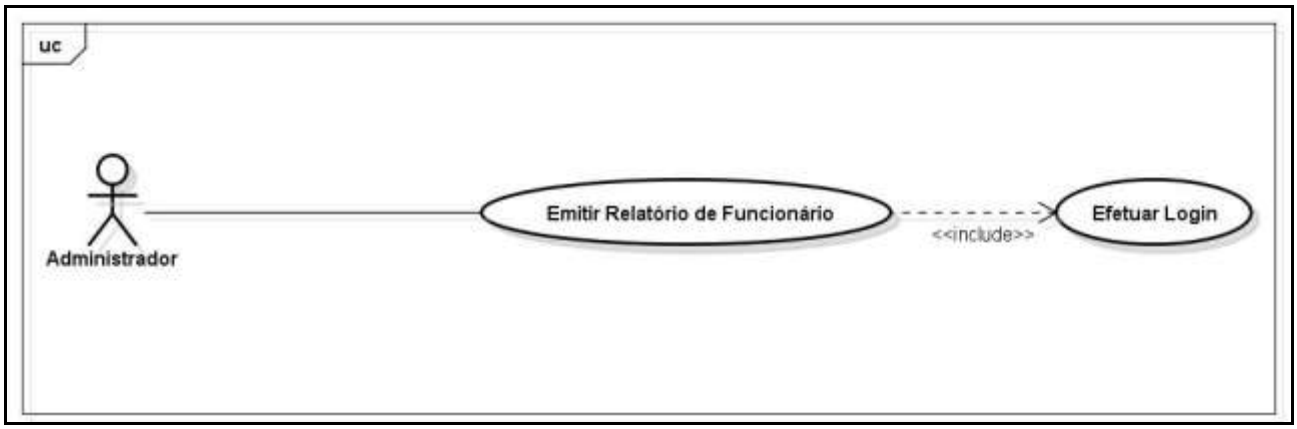


Figura 30: Diagrama de Caso de Uso Emitir Relatório de Funcionário.

Nome do Caso de uso	Emitir relatório de Funcionário
Atores	Administrador
Pré-Condições	Efetuar Login
Cenário Principal	1- O sistema informa os dados necessários para relatório. 2- O Administrador seleciona a opção “visualizar”. 3- O Administrador seleciona a opção “imprimir”. 4- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador cancela a operação

Tabela 17: Emitir Relatório de Funcionário.

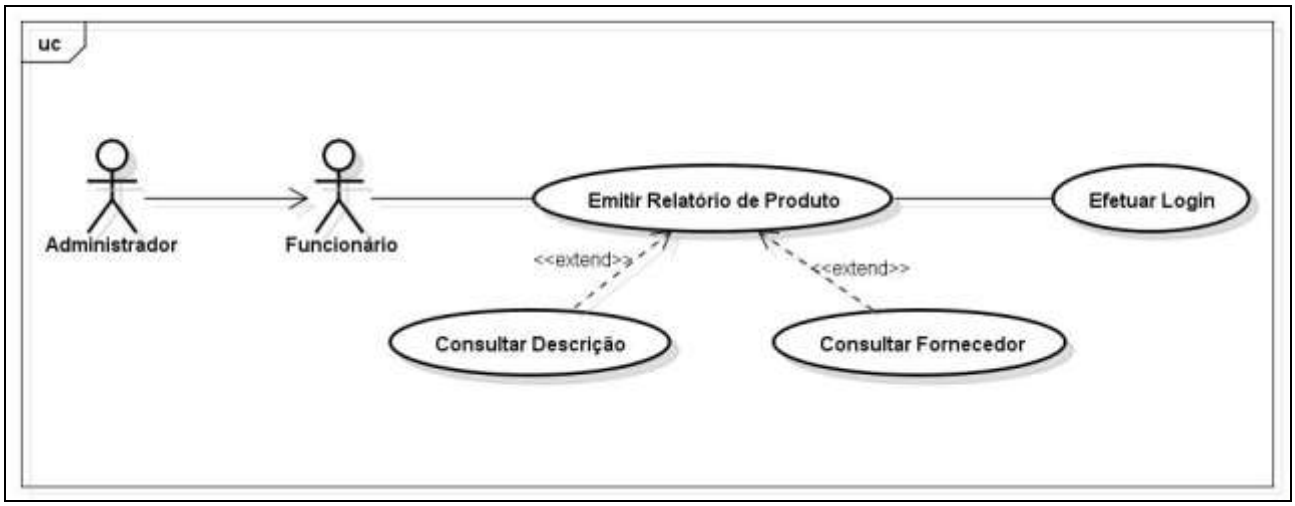


Figura 31: Diagrama de Caso de Uso Emitir Relatório de Produto.

Nome do Caso de uso	Emitir relatório de Produto
Atores	Administrador e Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O administrador ou funcionário pode escolher entre emitir relatório de todos os registros ou filtra-los. 2- Ao clicar em relatório por filtro o administrador ou funcionário escolhe qual tipo de consulta que deseja realizar e preenche os dados e clica em buscar. 3- O sistema preenche o relatório e mostra na tela. 4- O Administrador ou funcionário seleciona a opção “imprimir”. 5- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador e funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador e funcionário cancela a operação.

Tabela 18: Emitir Relatório de Produto.

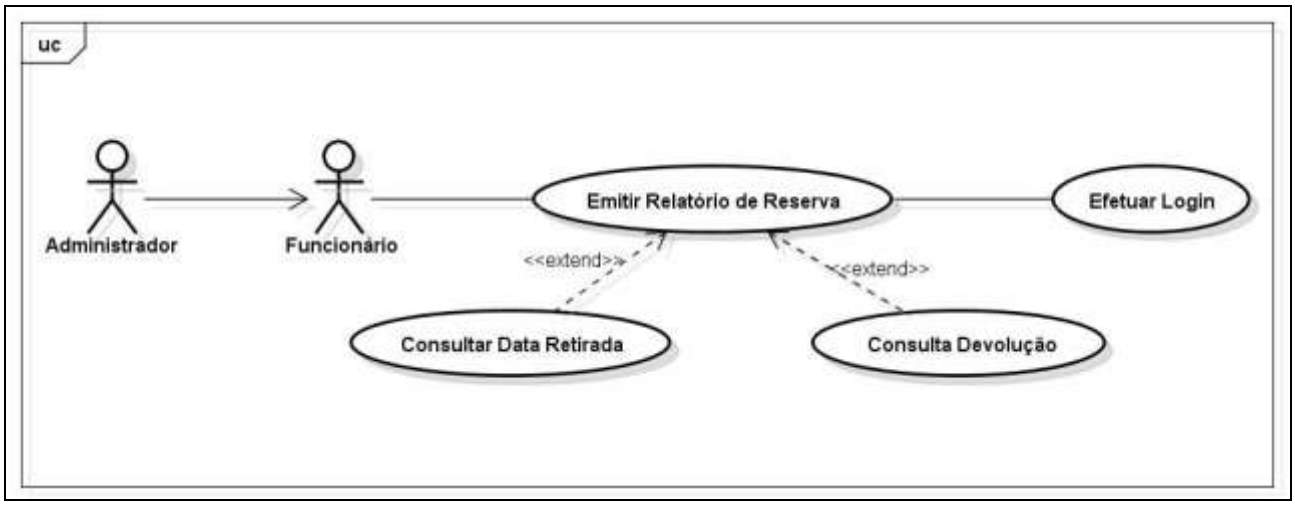


Figura 32: Diagrama de Caso de Uso Emitir Relatório de Reservas

Nome do Caso de uso	Emitir relatório de Reservas
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O administrador ou funcionário pode escolher entre emitir relatório de todos os registros ou

	filtra-los. O administrador e funcionário seleciona a opção “visualizar”. 2- Ao clicar em relatório por filtro o administrador ou funcionário escolhe qual tipo de consulta que deseja realizar e preenche os dados e clica em buscar. 3- O sistema preenche o relatório e mostra na tela. 4- O Administrador ou funcionário seleciona a opção “imprimir”. 5- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador e funcionário pode visualizar o relatório e não imprimir
Casos de Teste	1- Administrador e funcionário cancela a operação.

Tabela 19: Emitir Relatório de Reservas.

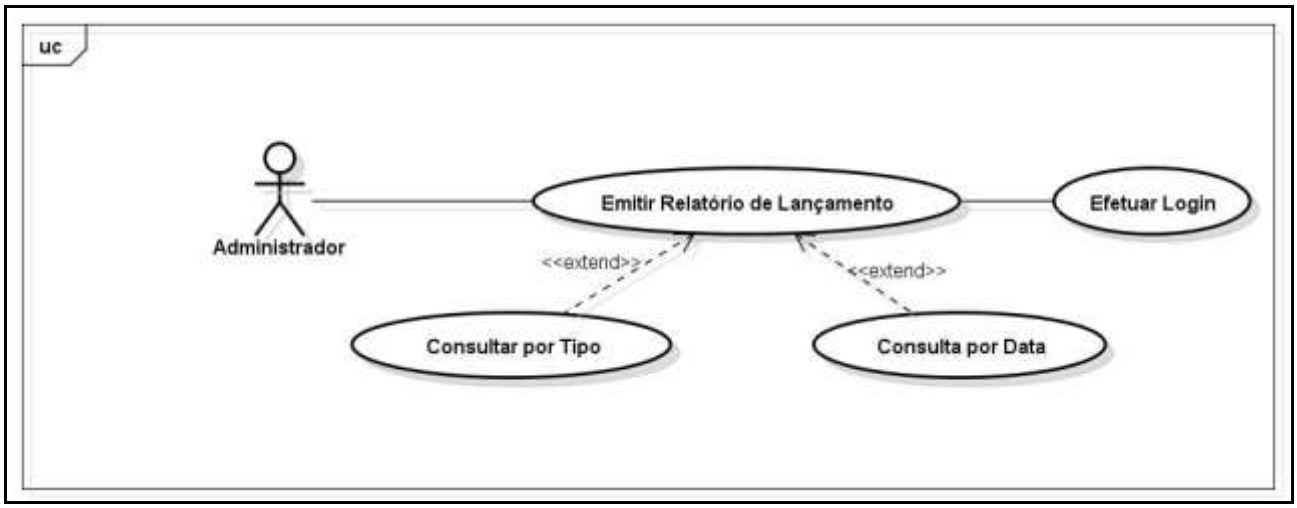


Figura 33: Diagrama de Caso de Uso Emitir Relatório de Lançamento.

Nome do Caso de uso	Emitir relatório de Lançamento
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O administrador ou funcionário pode escolher entre emitir relatório de todos os registros ou filtra-los. 2- Ao clicar em relatório por filtro o administrador ou funcionário escolhe qual tipo de consulta que

	deseja realizar e preenche os dados e clica em buscar. 3- O sistema preenche o relatório e mostra na tela. 4- O administrador ou funcionário seleciona a opção “imprimir”. 5- O sistema imprime o relatório com sucesso.
Cenário Alternativo	A1- Administrador e funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador e funcionário cancela a operação.

Tabela 20: Emitir Relatório de Lançamento.

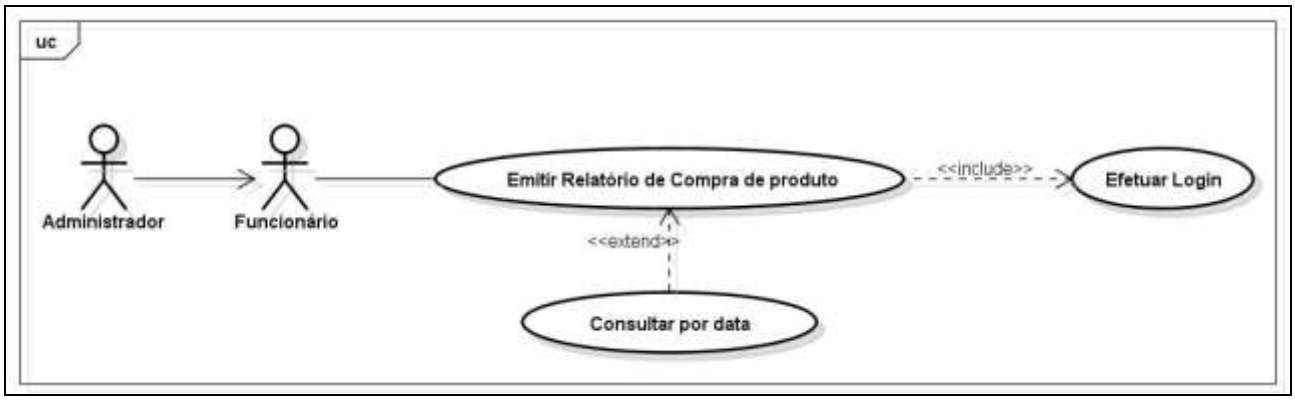


Figura 34: Diagrama de Caso de Uso Emitir Relatório de Compra de Produto.

Nome do Caso de uso	Emitir relatório de Compra de Produto
Atores	Administrador e funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- O administrador ou funcionário pode escolher entre emitir relatório de todos os registros ou filtra-los. 2- Ao clicar em relatório por filtro o administrador ou funcionário escolhe qual tipo de consulta que deseja realizar e preenche os dados e clica em buscar. 3- O sistema preenche o relatório e mostra na tela. 4- O administrador ou funcionário seleciona a opção “imprimir”. 5- O sistema imprime o relatório com sucesso.

Cenário Alternativo	A1- Administrador e funcionário pode visualizar o relatório e não imprimir.
Casos de Teste	1- Administrador e funcionário cancela a operação.

Tabela 21: Emitir Relatório de Compra de Produto.

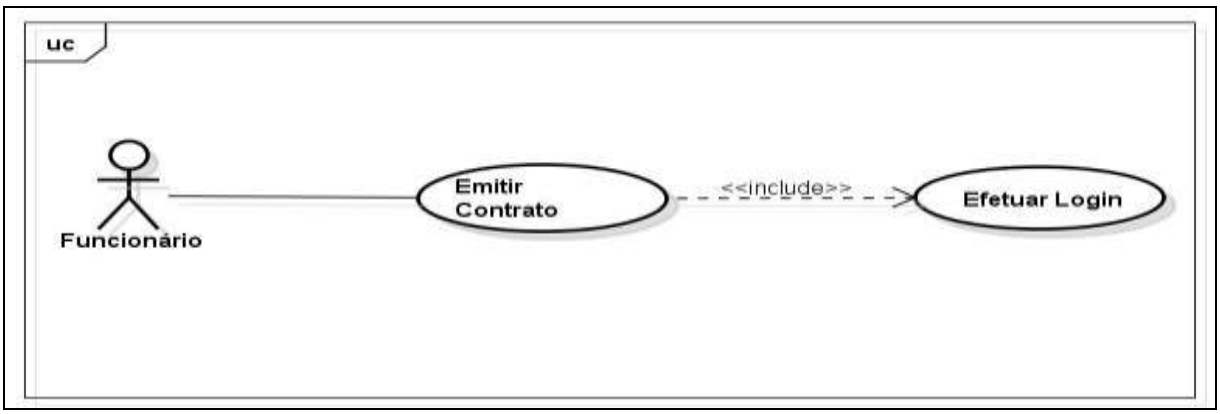


Figura 35: Diagrama de Caso de Uso Emitir Contrato.

Nome do Caso de uso	Emitir Contrato
Atores	Funcionário
Pré-Condições	Efetuar Login
Cenário Principal	1- Após ter efetuado a reserva, o sistema mostra uma mensagem “Contrato gerado com sucesso”. 2- O ficará em uma pasta especifica do computador, e o funcionário porderá abri-lo apartir daí. 3- O funcionário entao por escolher se irá imprimir a via do cliente ou nao.
Cenário Alternativo	A1- O funcionário pode visualizar o contrato e não imprimir.
Casos de Teste	1- O funcionário cancela a operação.

Tabela 22: Emitir Contrato.

Os diagramas de caso de uso apresentados representam as funcionalidades e suas respectivas narrativas do sistema Ltrajes para as entidades “Administrador” e “Funcionário”.

3.4 – DIAGRAMA DE ATIVIDADES

O diagrama de atividades descreve passo a passo o caminho percorrido para a conclusão de uma atividade. Mostra o fluxo de controle de uma atividade para outra representando os aspectos dinâmicos sendo utilizada para modelar uma pequena parte do código de um sistema (GUEDES, 2011).

Neste diagrama é possível compreender como irá funcionar a realização da reserva, onde o cliente irá fazer a escolha do produto e o funcionário deverá prosseguir com a movimentação da reserva. A figura 36 ilustra o diagrama de atividade para o cenário “Movimentar Reserva”.

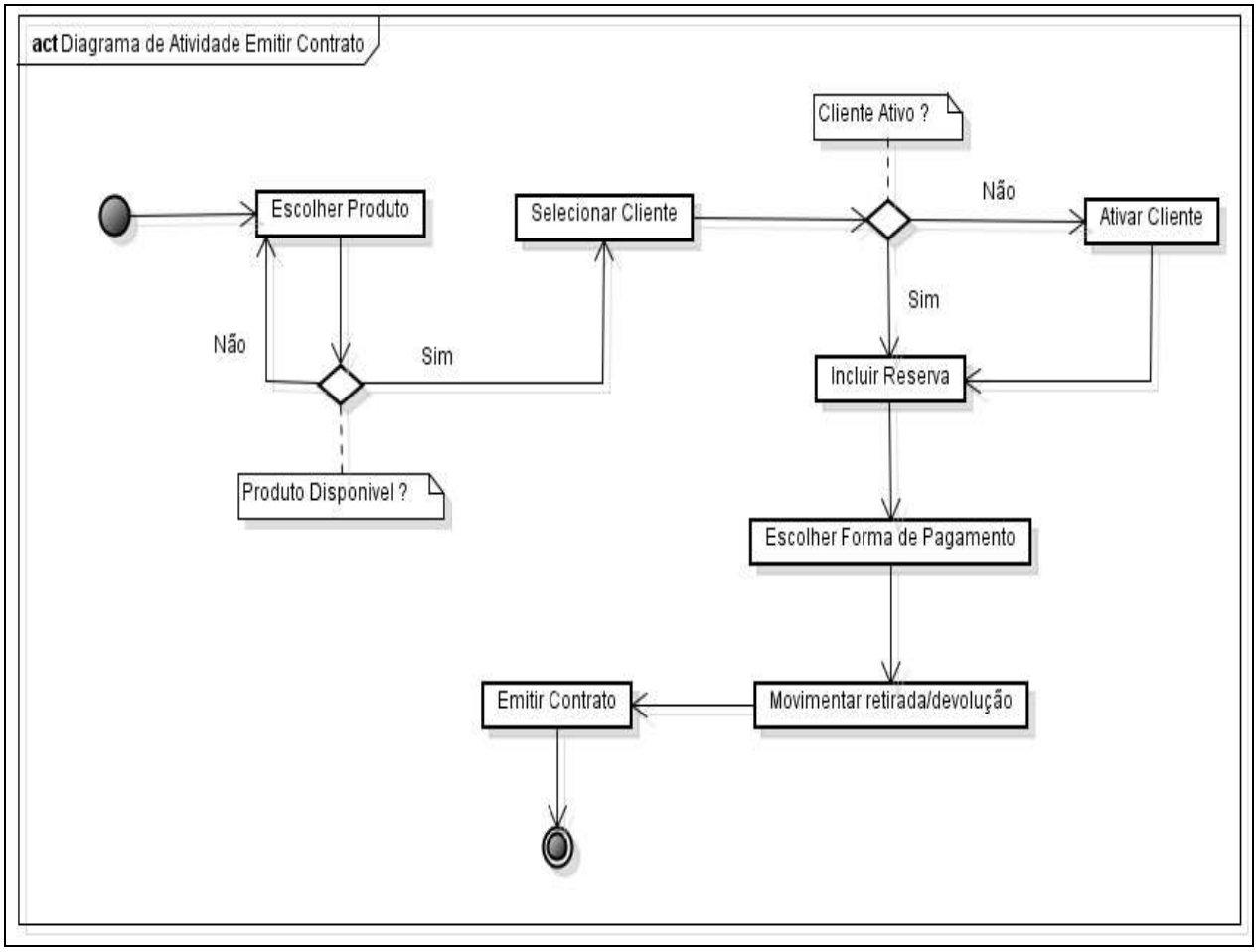


Figura 36: Diagrama de Atividades Movimentar Reserva.

Neste diagrama é possível compreender como irá funcionar a realização de cadastro de clientes que poderá ser efetuado tanto pelo funcionário quanto pelo administrador do sistema. A figura 37 ilustra o diagrama de atividade para o cenário “Cadastrar Clientes”.

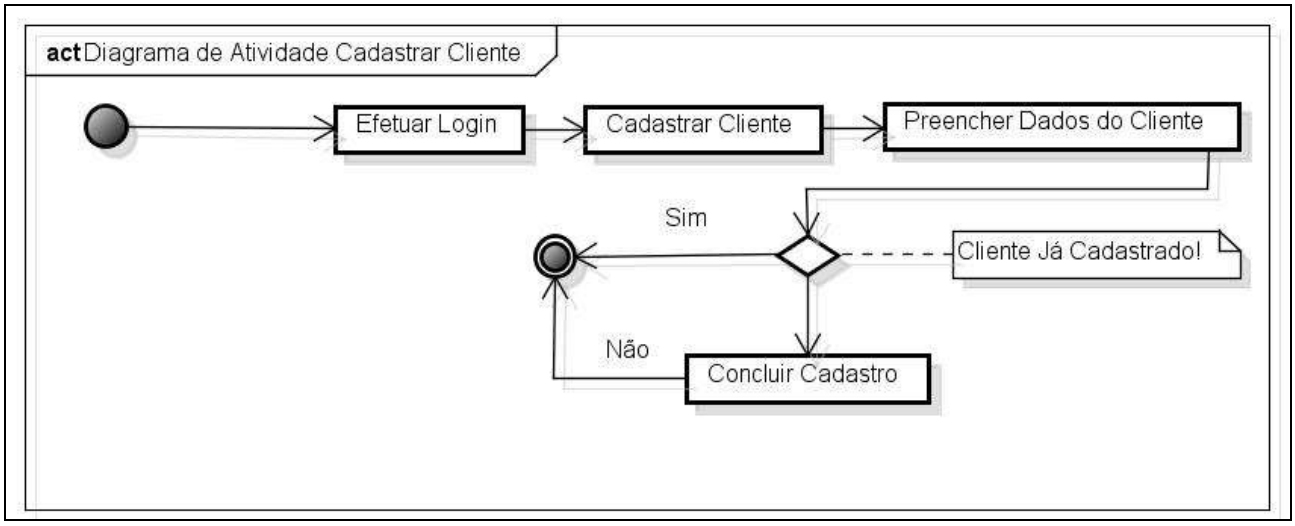


Figura 37: Diagrama de Atividades Cadastrar Clientes.

Os diagramas de atividade apresentados ilustram o processo de uma determinada atividade.

3.5 – DIAGRAMA DE SEQUÊNCIA

Diagrama de sequência é utilizado para mostrar a interação do sistema com o objeto que se dá a funcionalidade do caso de uso. Possibilita a troca de informações utilizando os objetos para enviar mensagens que contém informações relacionadas a determinadas atividades (NUNES et al., 2011).

O diagrama de sequência representa a lógica do sistema para que seja efetuada a consulta de cliente a verificação de disponibilidade do produto e a efetivação da reserva. A figura 38 ilustra o diagrama de sequência para o cenário “Consultar Cliente”.

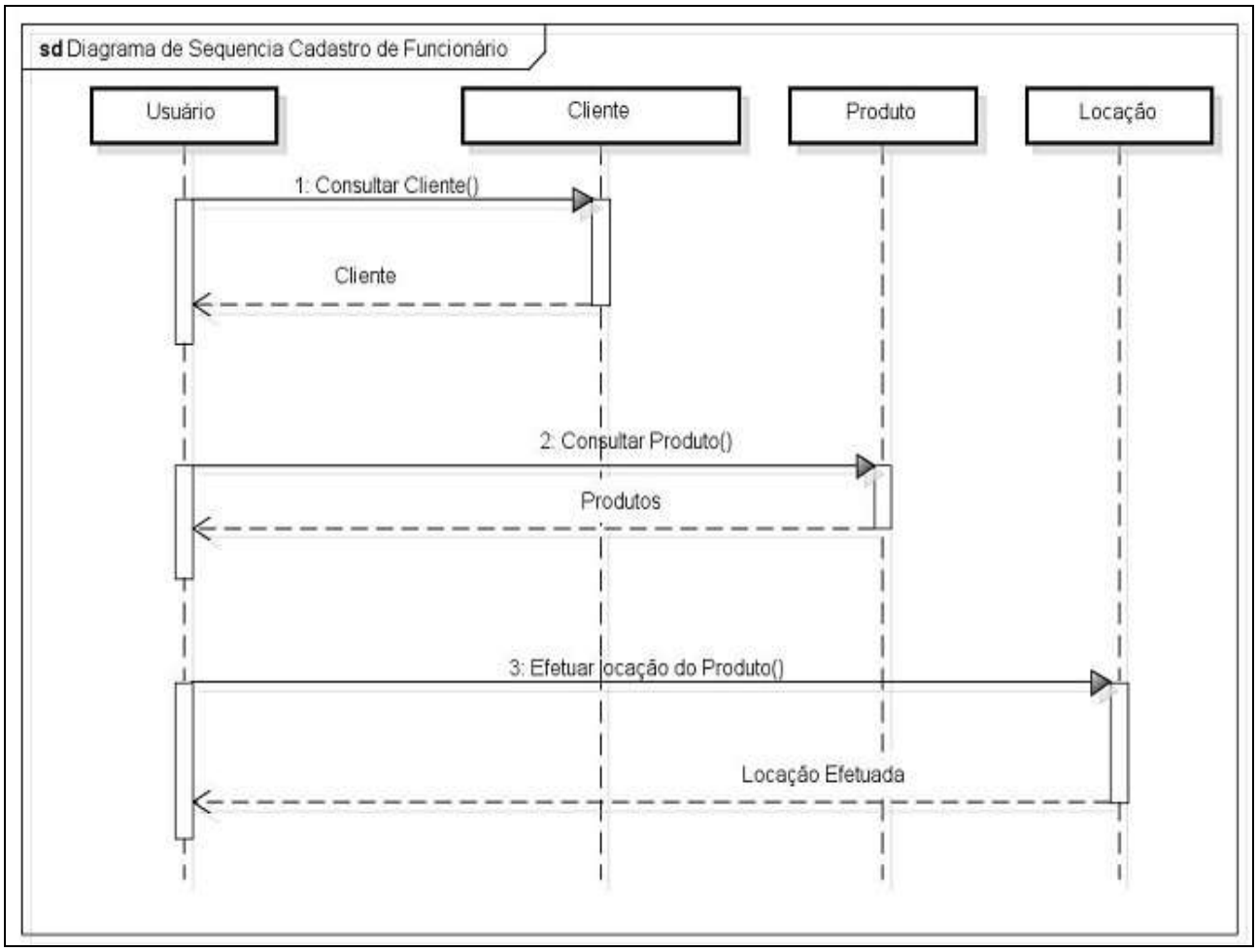


Figura 38: Diagrama de Sequência Consultar Cliente.

Neste diagrama é representado como o sistema irá funcionar para a efetivação do cadastro de funcionários. A figura 39 ilustra o diagrama de sequência para o cenário “Cadastrar Funcionário”.

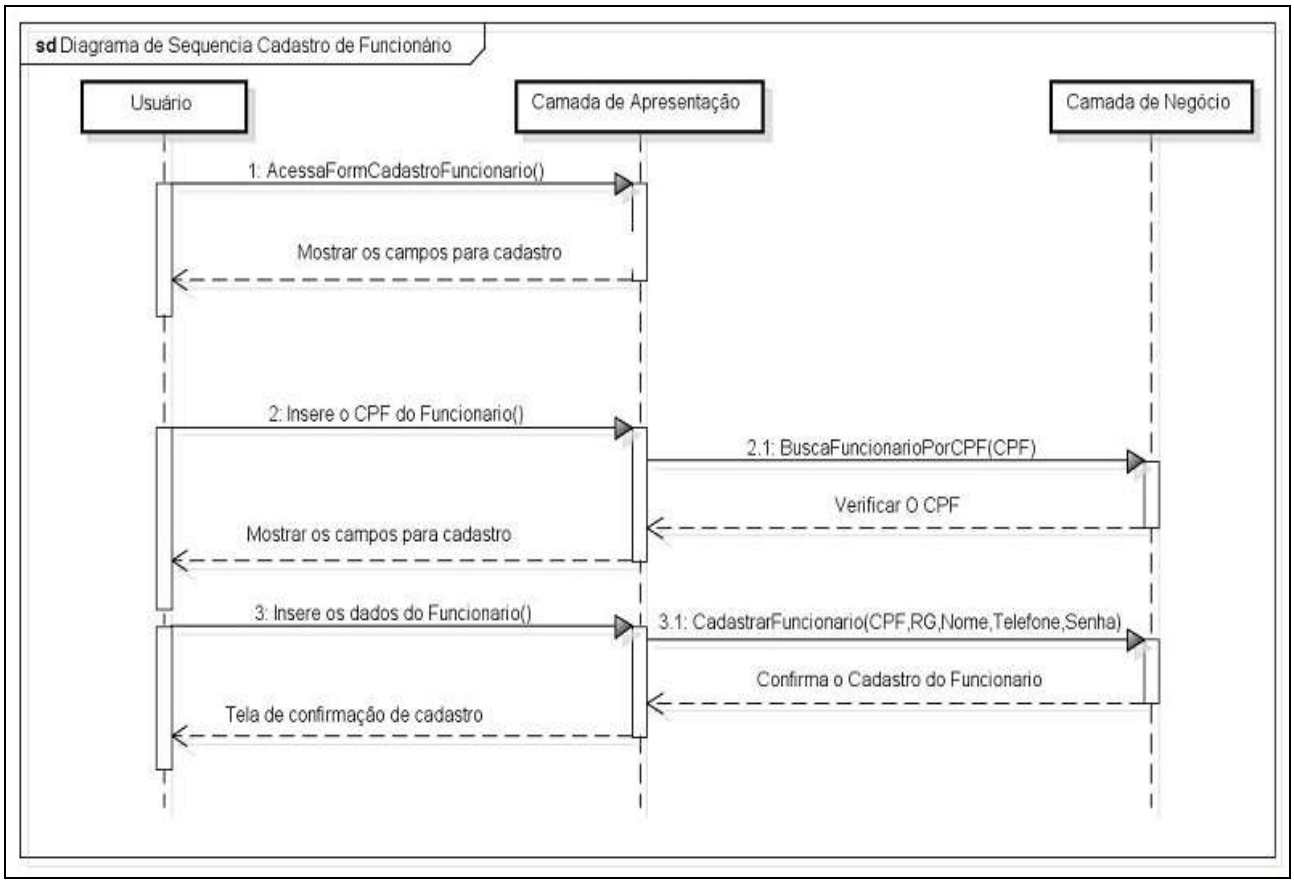


Figura 39: Diagrama de Sequência Cadastro de Funcionário.

Os diagramas de sequência apresentados ilustram a troca de mensagens para a execução de uma determinada atividade solicitada pela entidade.

3.6 – DIAGRAMA DE CLASSES

O diagrama de classes detalha os atributos, métodos e relacionamentos de cada classe que será utilizado no sistema, composto pelos elementos abstratos de modelação o diagrama de classes descreve o modelo de um sistema com o objetivo de suportar os requisitos funcionais, sendo importante para a documentação e para o desenvolvimento do sistema enfatizando os dados necessários para a sua construção (NUNES et al., 2011).

A figura 40 ilustra o diagrama de classes onde detalha os relacionamento, métodos e atributos de cada classe.

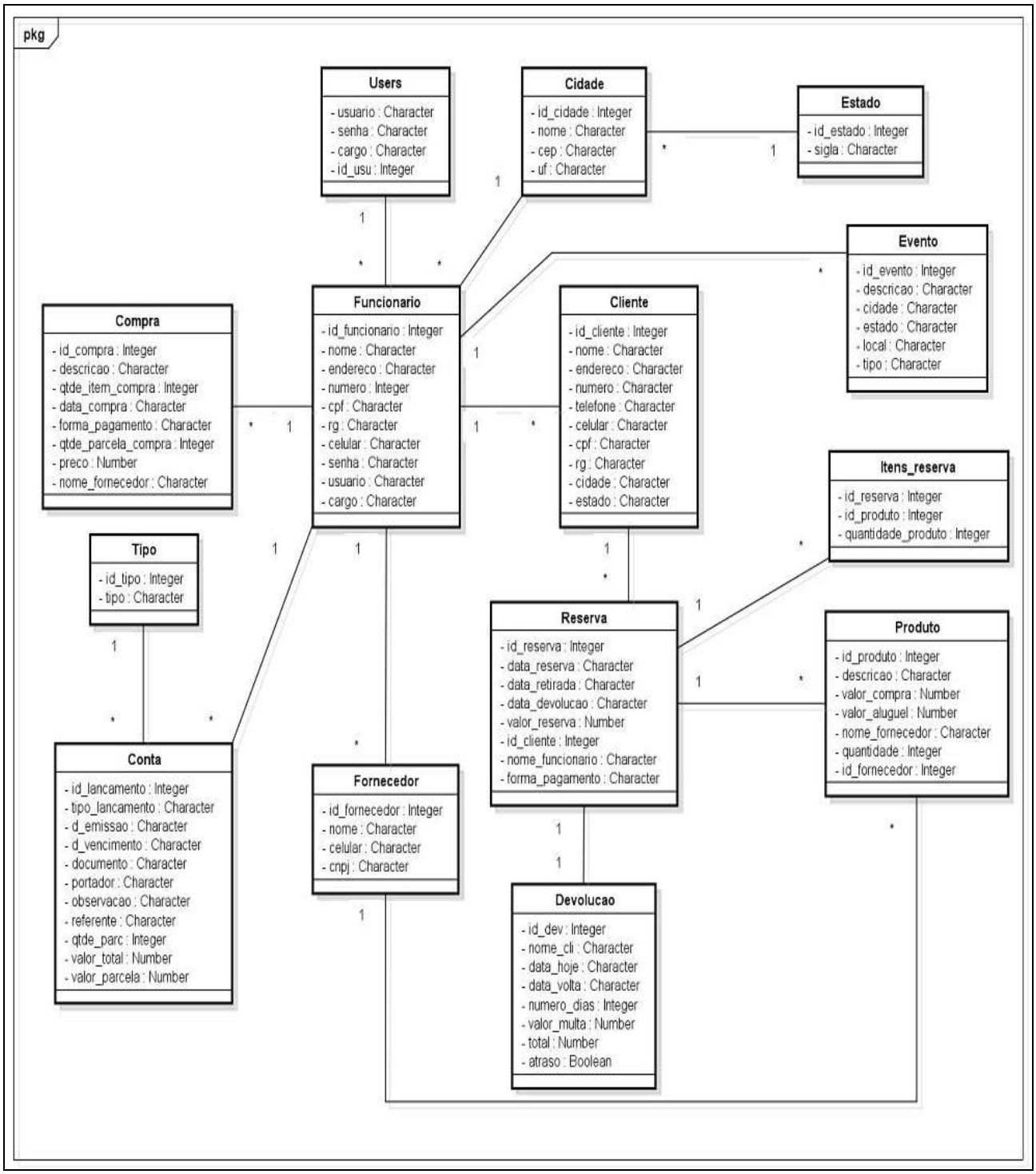


Figura 40: Diagrama de Classes.

Com este diagrama apresentamos as definições das classes, atributos e relacionamentos existentes no sistema LTRAJES.

3.7 – MODELO ENTIDADE – RELACIONAMENTO

O Diagrama entidade-relacionamento é uma apresentação abstrata da estrutura que será utilizada no banco de dados da aplicação, mostram quais serão as chaves primarias, estrangeiras e suas referencias, facilitando assim o entendimento por parte do programador (SILBERSCHATZ, 2012).

A figura 41 ilustra o diagrama de entidade – relacionamento que é fundamental para o entendimento de como será modelado o banco de dados, facilitando o desenvolvimento do programador.

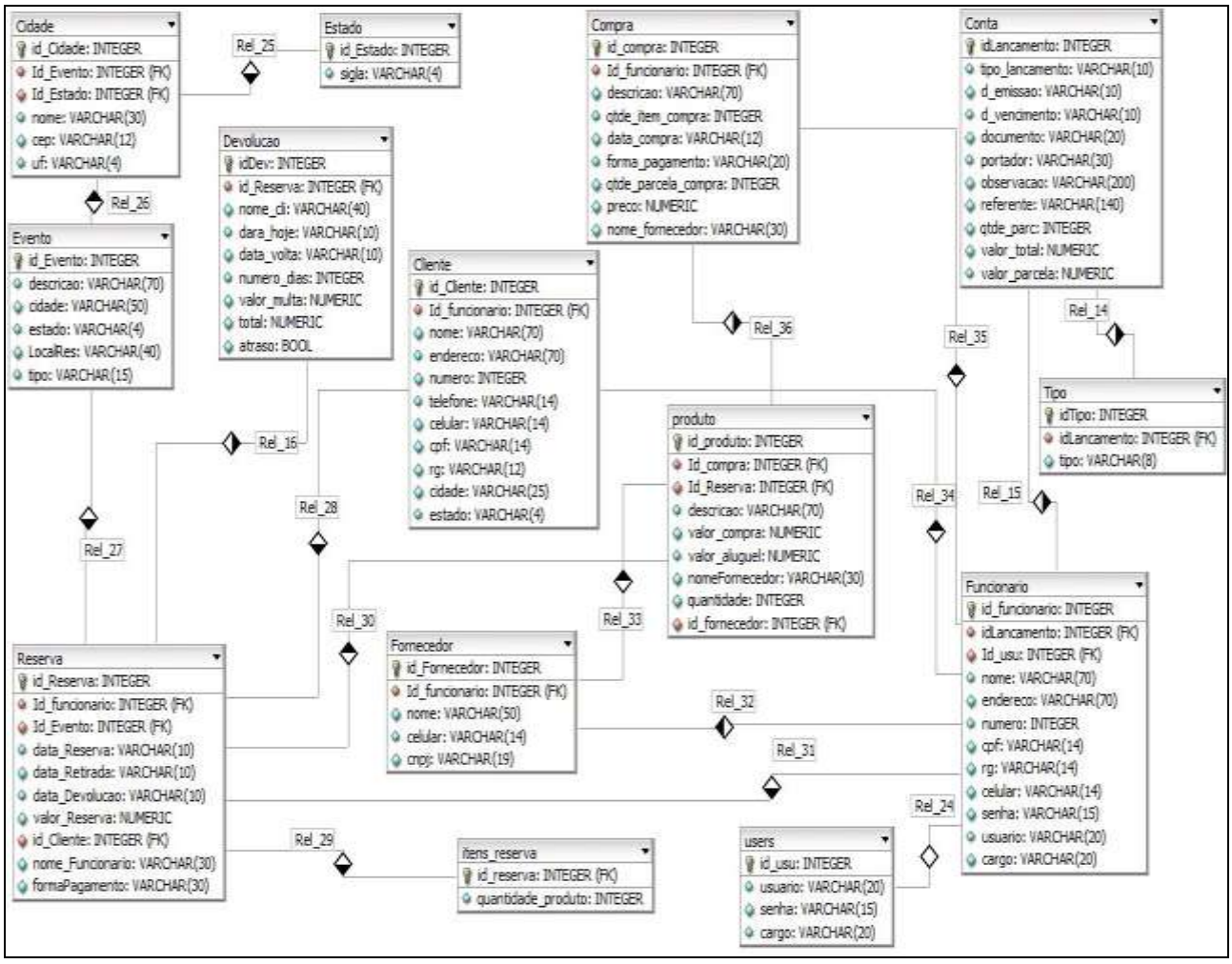


Figura 41: Diagrama de Entidade e Relacionamento.

O diagrama apresentado ilustra as tabelas e relacionamentos existentes no banco de dados do sistema Ltrajes.

4 – ESTRUTURA DO PROJETO

Neste capítulo será apresentada a estrutura do projeto que será adotada para o desenvolvimento do trabalho de conclusão de curso. As etapas do desenvolvimento da aplicação são ilustradas na Figura 42 que apresenta o diagrama *Work Breakdown Structure* (WBS) também conhecido como Estrutura Analítica do Projeto (EAP) que será utilizado para organizar e orientar o trabalho a ser desenvolvido.

4.1 – ESTRUTURA ANALÍTICA DO PROJETO

A implementação foi dividida em etapas e para defini-las utilizou-se o diagrama de Estrutura Analítica de Projetos (EAP) do inglês *Work Breakdown Structure* (WBS). A EAP tem, como principal propósito, organizar a estrutura analítica do projeto, dividindo o trabalho em partes pequenas com alto nível de detalhamento do projeto com a finalidade de facilitar o seu entendimento (CARVALHO, 2011).

A figura 41 ilustra o diagrama de EAP que detalha o projeto com a finalidade de facilitar o entendimento.

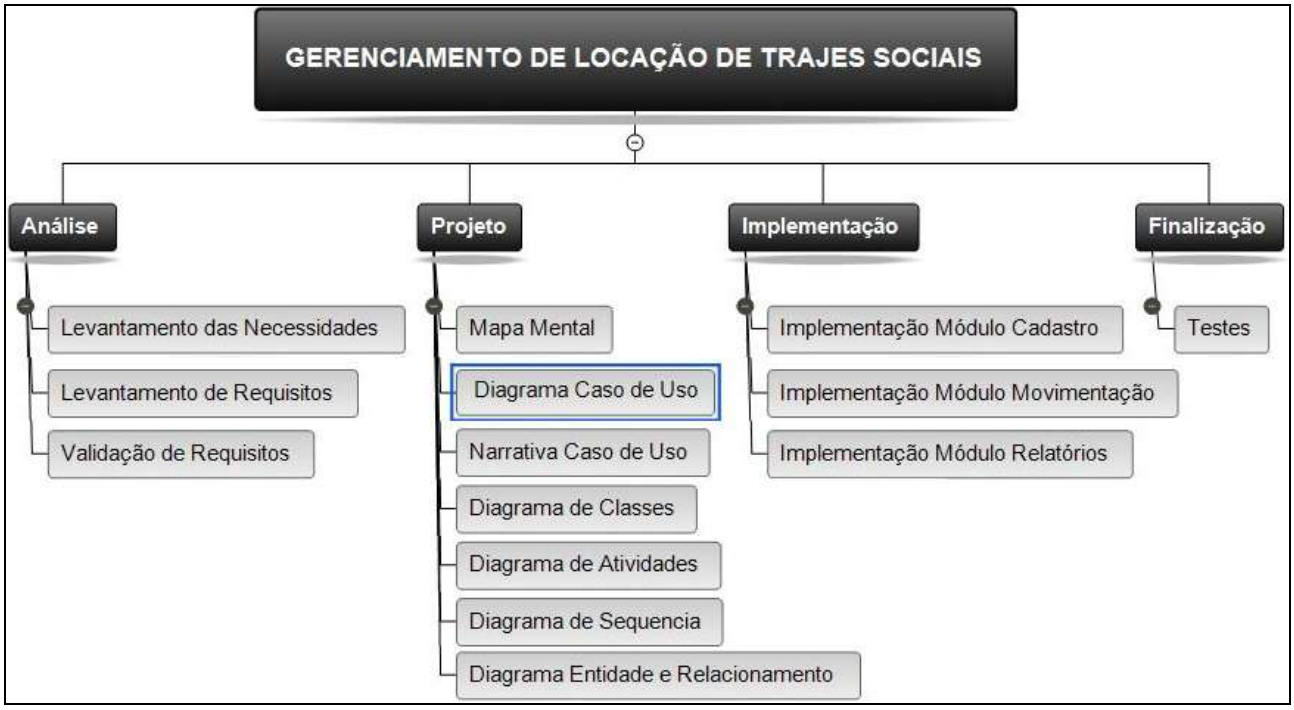


Figura 42: Estrutura Analítica do Projeto.

4.2 – SEQUÊNCIAMENTO DAS ATIVIDADES

O diagrama de sequênciamento de atividades ilustra o tempo de duração para a realização de cada atividade do projeto em desenvolvimento, tendo como objetivo definir de forma lógica de execução das tarefas.

A figura 43 ilustra a duração de cada atividade que compõe o projeto desenvolvido.

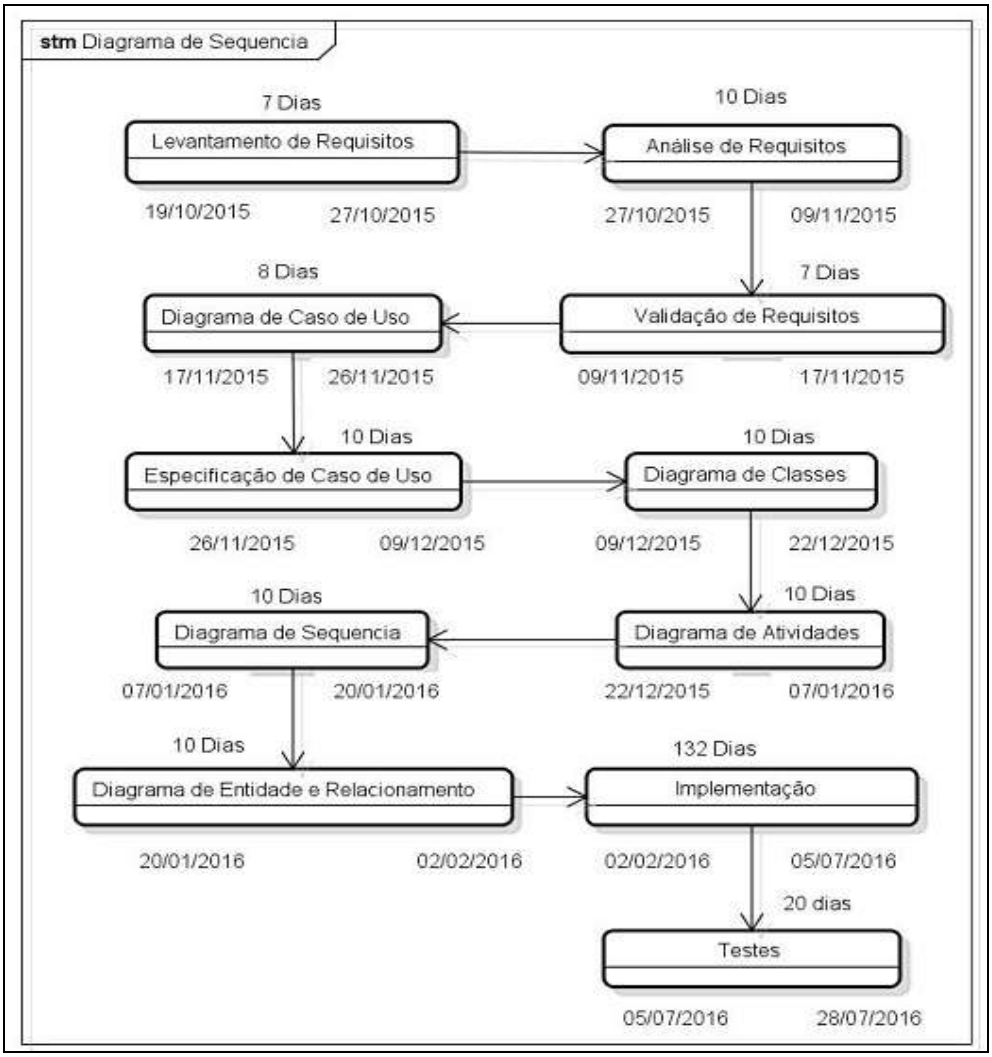


Figura 43: Sequênciamento das Atividades.

O diagrama apresentado ilustra o tempo necessário para conclusão de todas as atividades do trabalho de conclusão de curso.

4.3 – ORÇAMENTO

Os recursos necessários para análise e desenvolvimento do *software* LTrajes são:

Recursos Físicos:

- 01 Analista de Sistema / Programador;

Analista	**Quantidade Dias**	**Custo Dia**	**Total**
Thais Dizero	112	R$ 30,00	R$ 3.360,00
Programador	**Quantidade Dias**	**Custo Dia**	**Total**
Thais Dizero	132	R$ 30,00	R$ 3.960,00
Custo Total			**R$ 7.320,00**

Tabela 23: Orçamento Analista e Programador.

Equipamentos:

- 01 Notebook

  Valor Unitário = R$ 2.200,00

  Depreciação (2 anos) = R$2.200,00 / 24 = R$91,67 /mês

  Custo por dia = R$91,67/ 26 (dias) = R$3,52 (ao dia)

  Custo do computador = R$3,52 * 222 = R$781,44.

- 01 Impressora jato de tinta

  Valor = R$250,00

  Depreciação = R$2500,00 / 24 = R$10,42

  Custo do dia = R$10,42 / 26(dias) = R$0,40

  Custo impressora = R$0,40 *222 = R$88,80.

Custo Total dos equipamentos R$781,44 + R$88,80 = R$870,24

Recurso	**Valor Total**
Analista / Programador	R$7.320,00
Equipamentos	R$870,24
Valor Total do Projeto	**R$ 8.190,24**

Tabela 24: Orçamento Valor Total.

5 – IMPLEMENTAÇÃO DO APLICATIVO

Para a implementação do sistema *Ltrajes* foi utilizado o *IDE NetBeans, pois* fornece ferramentas profissionais para a criação de aplicativos em *desktop*, *web* e *mobile* (GONÇALVES, 2008).

A figura 44 ilustra a tela principal de criação do projeto no netbeans.

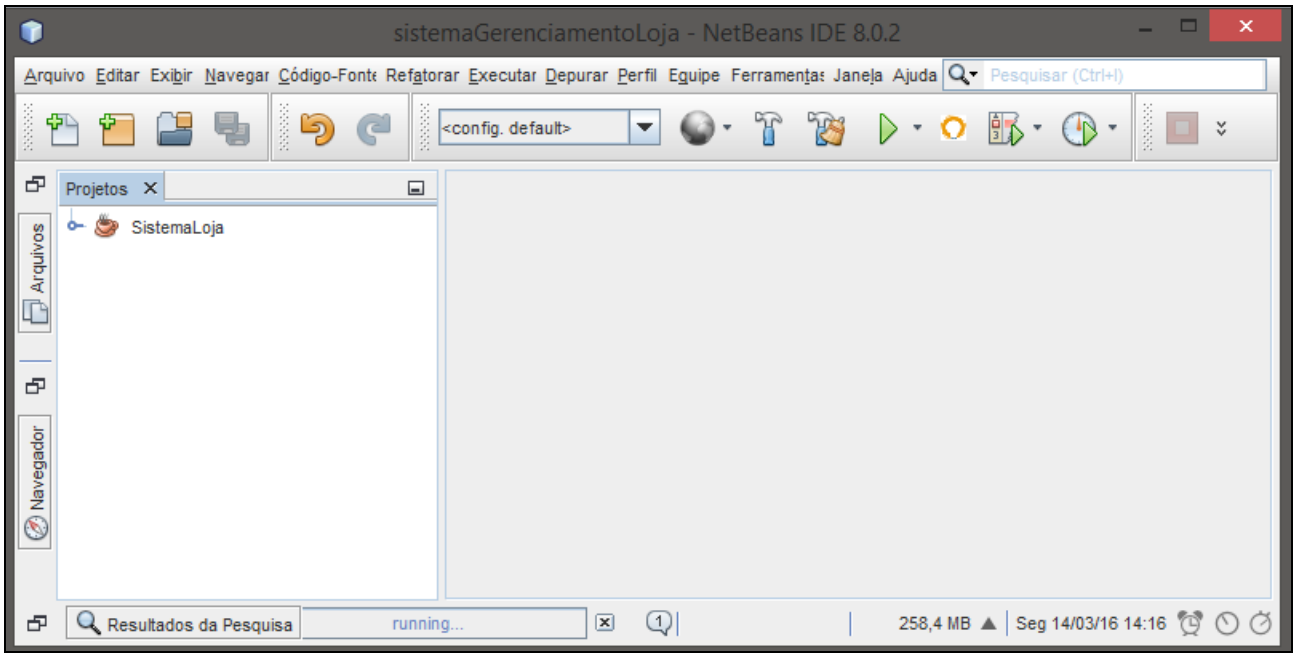


Figura 44: Projeto Sistema Gerenciamento Loja.

SistemaLoja: Projeto responsável por todas as camadas da *Model View Controller* (MVC), utilizando a linguagem de programação Java e o *framework Hibernate*. Além do banco de dados *Postgresql* por ser seguro e versátil.

5.1 – ORGANIZAÇÃO DO PROJETO JAVA

O projeto Java “SistemaLoja” foi dividido em pacote utilizando o padrão MVC: br.com.SistemaLoja.Beans, br.com.SistemaLoja.Conection, br.com.SistemaLoja.Dal,

br.com.SistemaLoja.Dal_filtros, br.com.SistemaLoja.View, br.com.SistemaLoja.imagens e br.com.SistemaLoja.view.relatorios.

A figura 45 ilustra como foi organizado e separado em pacotes cada parte do projeto.

Figura 45: Organização do Projeto.

br.com.SistemaLoja.Beans: Pacote utilizado para encapsular e criar os métodos getter e setter.

br.com.SistemaLoja.Conection: Pacote onde se encontra a classe responsável pela conexão com o banco de dados. O apêndice A exemplifica uma classe Java para conexão com o banco de dados.

br.com.SistemaLoja.Dal: Pacote onde se encontra a parte do CRUD (insert, select, update e delete) de cada classe do sistema como: Classe cliente, produto, fornecedores entre outros. O apêndice B apresenta uma classe Java para persistência de dados.

br.com.SistemaLoja.Dal_filtros: Pacote onde se encontra os controladores para geração de relatórios por filtros.

br.com.SistemaLoja.View: Pacote que é responsável por organizar os formulários de interação com o usuário como: cadastro de clientes, fornecedores, produtos entre outros.

br.com.SistemaLoja.imagens: Pacote onde se encontra todas as imagens utilizadas pelo sistema.

br.com.SistemaLoja.view.relatorios: Pacote que é responsável por organizar os formulários de filtros de relatórios, local onde o usuário tem interação com o sistema.

5.2 – INTERFACES DO SISTEMA

Ao abrir o sistema, a tela de autenticação será apresentada para que assim o usuário possa se identificar. O apêndice C apresenta a classe Java responsável pelo login na aplicação.

A figura 46 ilustra a tela de login, onde para executar qualquer função no sistema o usuário deve se autenticar.

Figura 46: Tela de Login.

Após a autenticação do usuário o sistema exibirá a tela principal onde se encontra os itens: cadastrar, consultar, movimentações, relatórios, contato e opção para sair do sistema.

A figura 47 ilustra a tela principal do sistema, onde se encontra todos os menus.

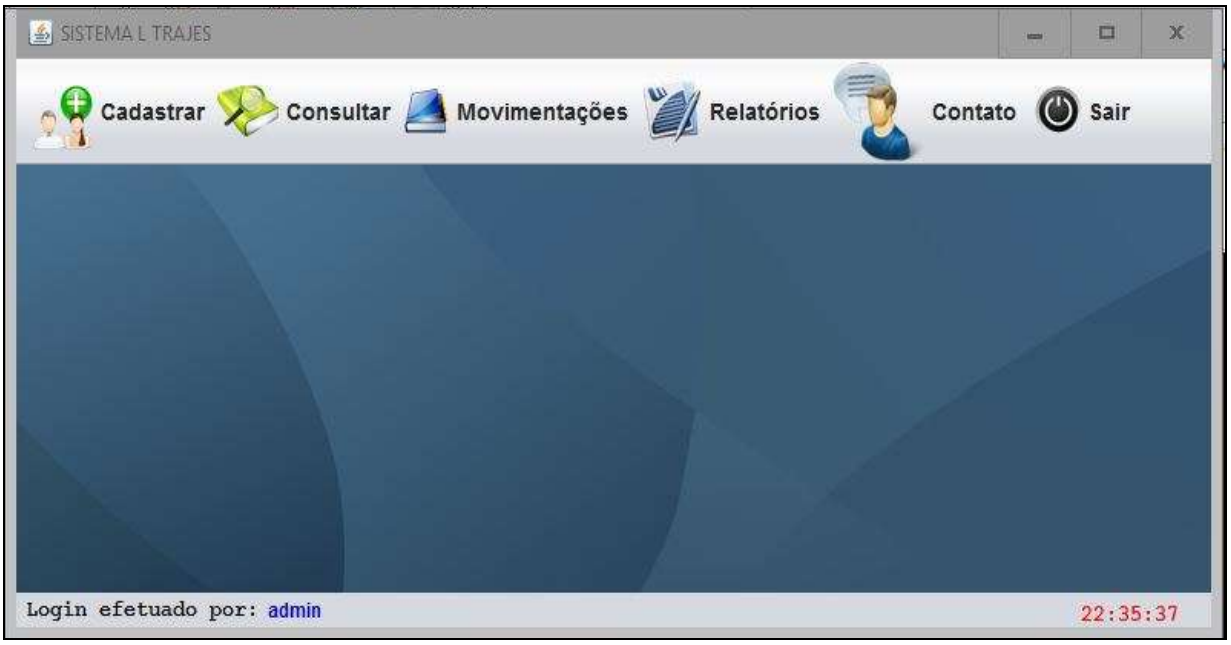


Figura 47: Tela Principal do sistema.

Assim que o usuário efetuar sua autenticação a tela principal será aberta para que possa ser selecionada qual funcionalidade se deseja.

A primeira opção é o menu de cadastro que ao ser selecionado irá exibir um submenu onde se encontra os cadastros dos clientes, funcionários, produtos, fornecedores entre outros.

A segunda opção é o menu consultar que se selecionado irá exibir um submenu onde se encontra os formulários para consulta de cliente, produtos e reservas.

Ao selecionar o menu movimentações irá aparecer um submenu onde se encontra a opção para efetuar a reserva, efetuar devolução, e efetuar compra de produtos.

A figura 48 ilustra o formulário de efetuação de reserva, onde o funcionário faz a busca do produto, a busca do cliente, e escolhe a data para retirada e devolução do produto.

Figura 48: Tela Efetuar Reserva

A figura 49 ilustra o formulário de devolução de produto, onde pode-se pesquisar se há reserva pelo nome do cliente, calcular multa caso o produto seja devolvido com atraso.

Figura 49: Tela Efetuar Devolução

Caso o usuário selecione o menu relatórios irá ser exibido um submenu onde se encontra a opção para emitir relatórios de clientes, produtos, fornecedores, e nos submenus de eventos, produtos, lançamentos reservar, irá aparecer outros menu onde o usuário poderá escolher se quer exibir os relatórios completos ou por filtro.

A figura 50 ilustra cadastro de produto deverá onde o funcionário ou administrador escolhe a opção no menu Cadastros e depois Produto, e em seguida deverá clicar em novo para habilitar os campos que deveram ser preenchidos.

Figura 50: Tela de Cadastro de Produtos.

.

6 – CONCLUSÃO

Empresas de pequeno porte que estão na área de locação de trajes costumam ser simples, sendo assim, não utilizam de um software para fazer o controle da loja. Este software tem como objetivo auxiliar no cadastro de produtos, clientes e na organização das reservas, evitando assim uma possível duplicidade de reservas do mesmo produto.

O sistema Ltrajes apresenta algumas funcionalidades pensadas para facilitar o atendimento aos proprietários das empresas que optarem pelo uso da aplicação, onde irá permitir um melhor controle das informações obtidas pela empresa tais como: dados de clientes, funcionário, fornecedores, além de armazenar um histórico sobre as reservar realizadas.

As funcionalidades principais desenvolvidas neste software como: movimentação de reservas e devoluções agrega valores às empresas, pois a agilidade na efetuação de reservas e uma organização de seus registros valorizam a empresa e ajuda na satisfação dos clientes.

6.1 TRABALHOS FUTUROS

A partir deste projeto é possível desenvolver várias outras implementações adicionando novas funcionalidades. Há vários recursos que podem ser implementados no projeto como: emissão de nota fiscal e sincronização para pagamentos com máquinas de cartão de crédito.

Com o desenvolvimento desse projeto outra oportunidade se originou. Após o término deste sistema será iniciado o desenvolvimento de um software para uma clínica de prótese dentária, solicitado por uma empresa também localizada na cidade de Andirá – PR, sendo assim, através deste software o meu trabalho passará a ser conhecido e consequentemente gerando mais oportunidades de trabalho aumentando gradativamente meus conhecimentos na área em que estou me formando.

REFERÊNCIAS

ABEOC BRASIL. **Pesquisa da Associação Brasileira de Eventos Sociais mostra que o mercado de festas e cerimônias atingiu R$ 16,8 bi no ano passado.** http://www.abeoc.org.br/2015/05/pesquisa-da-associacao-brasileira-de-eventos-sociais-abrafesta-mostra-que-o-mercado-de-festas-e-cerimonias-atingiu-r-168-bi-no-ano-passado/. Acesso em 25 Set. 2015.

CARVALHO, Claudinê Jordão. **Elaboração e Gestão de Projetos.** Florianópolis: Departamento de Ciências da Administração, 2011.

CLARO, Daniela Barreiro; SOBRAL, João Bosco Mangueira. **Programação em Java**. Florianópolis: Copyleft Pearson Education, 2008.

DEITEL, Paul; DEITEL, Harvey. **Java Como Programar**, 8. ed Tradução de Edson Furmankiewicz. São Paulo: Editora Pearson Prentice, 2010.

GABARDO, Ademir Cristiano. **PHP e MVC com CodeIgniter,** 1. ed. São Paulo: Editora Novatec, 2012.

GUEDES, Gilleanes T. A. **UML 2 Uma Abordagem Prática.** 2 ed. São Paulo: Editora Novatec, 2011.

GÓES, Wilson Moraes. **Aprenda UML por meio de estudos de caso.** 1. ed. São Paulo: Editora Novatec, 2014.

GONÇALVES, Edson. **Netbeans IDE 6- Desenvolvendo aplicações web**, 1. ed. Rio de Janeiro: Editora Ciência Moderna, 2008.

GONÇALVES, Edson. **Livro Desenvolvendo Relatórios Profissionais com iReport para Netbeans IDE,** 1. Ed. Rio de Janeiro: Editora Ciência Moderna, 2009.

INVISIONAPP. http://www.invisionapp.com/. Acesso em 10 Out. 2015.

MAYMALA, Jayadevan. **Postgresql for Data Architects**, 1. ed. Birmingham, Reino Unido: Editora Packt Publishing, 2015.

MEDEIROS, Higor. **JUnit - Testes de Integração com Java e Junit. 2009.** Disponível em http://www.devmedia.com.br/testes-de-integracao-com-java-e-junit/25662. Acesso em 10 Out. 2015.

MENDES, Douglas Rocha. **Programação Java com Ênfase em Orientação a Objetos,** 1. ed. São Paulo: Editora Novatec, 2009.

MILANI, André. **Postgresql Guia do Programador,** 1. ed. São Paulo: Editora Novatec, 2008.

MORAIS, Maurício. **iReport Crie relatórios práticos e elegantes,** 1. ed. São Paulo: Editora Casa do Código, 2016.

NUNES, Mauro; O'NEILL, Henrique. **Fundamental de UML,** 7. ed. Portugal: Editora FCA, 2011.

SIERRA, Kathy; BATES, Bert. **Use a Cabeça! Java,** 2. ed. Rio de Janeiro: Editora Alta Books, 2005.

SILBERSCHATZ, Abraham. **Sistema de Banco de dados,** 6. ed. São Paulo: Editora Elsevier, 2012

SILVEIRA, Paulo; SILVEIRA, Guilherme; LOPES, Sérgio; MOREIRA, Guilherme; STEPPAT, Nico; KUNG, Fabio. **Introdução à Arquitetura e Design de Software,** 1. ed
São Paulo: Editora Elsevier - Campus, 2012.

SOUZA, Lucilene. **Redesign da Informação no Processamento de Imagem**, Tese (Doutorado) – Universidade Federal de Santa Catarina.

APÊNDICE - A

Classe responsável pela conexão com o banco de dados onde se encontra toda a parte de identificação e armazenamento de dados.

```
package br.com.SistemaLoja.ModeloConection;
import java.sql.*;
import javax.swing.JOptionPane;
public class ConexaoBD {
    public Statement stm;// responsavel por realizar pesquisa no BD
    public ResultSet rs;// armazena o resultado da pesquisa
    private String driver  = "org.postgresql.Driver";// identifica o serviço
    public static Connection con; //responsavel por realizar a conexao com o BD

    public void conexao(){//metodo responsavel por realizar conexao com a base de dados
        System.setProperty("jdbc.Drivers", driver);// responsavel por setar a propriedade do drive de conexao
        try {
            con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/Ltrajes","postgres","96210298");
            //JOptionPane.showMessageDialog(null, "Conectado com Sucesso");
        } catch (SQLException ex) {
           JOptionPane.showMessageDialog(null, "Erro ao se conectar ao BD:\n"+ex.getMessage());
        }}
    public void executaSql (String sql){
        try {
            stm = con.createStatement(rs.TYPE_SCROLL_INSENSITIVE, rs.CONCUR_READ_ONLY);
            rs = stm.executeQuery(sql);
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Erro ExecutaSql:\n"+ex.getMessage());
        }}
    public void desconecta(){
        try {
            con.close();
            //JOptionPane.showMessageDialog(null, "BD Desconectado com Sucesso");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, "Erro ao Desconectar:\n"+ex.getMessage());
        }}}
```

Figura 51: Método de conexão com o banco de dados.

APÊNDICE - B

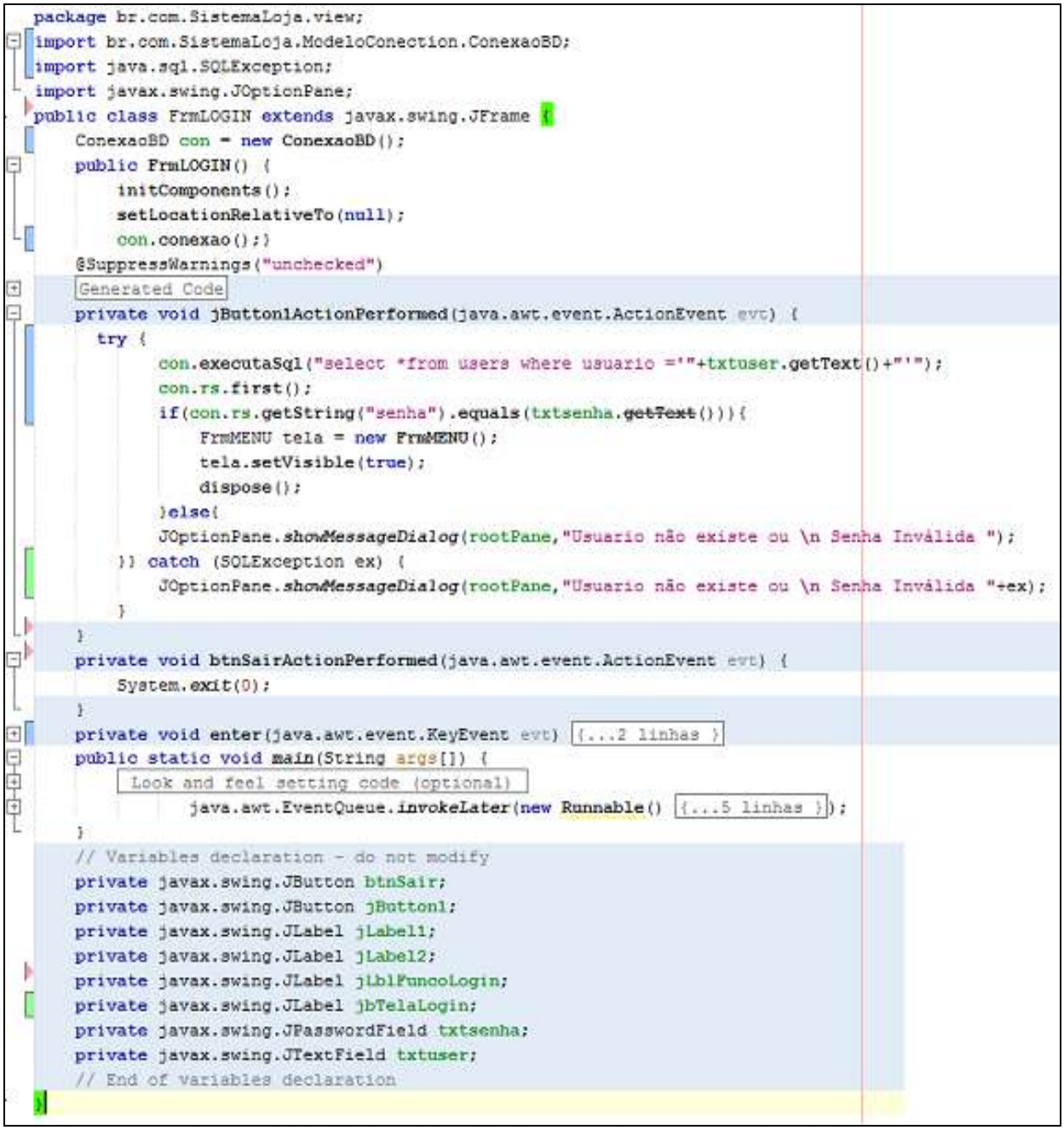
Classe responsável pelos métodos de *insert, select, update e delete* clientes.

```
package br.com.SistemaLoja.ModeloDao;

import br.com.SistemaLoja.ModeloConection.ConexaoBD;
import br.com.SistemaLoja.modeloBeans.BeansCliente;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class DaoCliente {

    ConexaoBD conex = new ConexaoBD();
    BeansCliente mod = new BeansCliente();

    public void Salvar(BeansCliente mod){
        conex.conexao();
        try {
        PreparedStatement pst;
        pst = conex.con.prepareStatement("insert into cliente(nome, endereco, numero, telefone, celular, cpf, "
                + "rg, cidade, estado) values (?,?,?,?,?,?,?,?,?)");
        pst.setString(1, mod.getNome());
        pst.setString(2, mod.getEndereco());
        pst.setInt   (3, mod.getNumero());
        pst.setString(4, mod.getTelefone());
        pst.setString(5, mod.getCelular());
        pst.setString(6, mod.getCpf());
        pst.setString(7, mod.getRg());
        pst.setString(8, mod.getCidade());
        pst.setString(9, mod.getEstado());
        pst.execute();
            JOptionPane.showMessageDialog(null,"Dados Inseridos com Sucesso");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null,"Erro ao  Inserir Dados/nErro:"+ex);
        }
        conex.desconecta();
    }

    public  BeansCliente buscaCliente(BeansCliente mod){...21 linhas }

    public void Excluir(BeansCliente mod){...14 linhas }

    public void Editar(BeansCliente mod){...25 linhas }
}
```

Figura 52: Classe responsável pelos *insert, select, update e delete* de clientes.

APÊNDICE - C

Classe programada onde se encontra o método de login do sistema.



```
package br.com.SistemaLoja.view;
import br.com.SistemaLoja.ModeloConection.ConexaoBD;
import java.sql.SQLException;
import javax.swing.JOptionPane;
public class FrmLOGIN extends javax.swing.JFrame {
    ConexaoBD con = new ConexaoBD();
    public FrmLOGIN() {
        initComponents();
        setLocationRelativeTo(null);
        con.conexao();}
    @SuppressWarnings("unchecked")
    Generated Code
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      try {
            con.executaSql("select *from users where usuario ='"+txtuser.getText()+"'");
            con.rs.first();
            if(con.rs.getString("senha").equals(txtsenha.getText())){
                FrmMENU tela = new FrmMENU();
                tela.setVisible(true);
                dispose();
            }else{
            JOptionPane.showMessageDialog(rootPane,"Usuario não existe ou \n Senha Inválida ");
        }} catch (SQLException ex) {
            JOptionPane.showMessageDialog(rootPane,"Usuario não existe ou \n Senha Inválida "+ex);
        }
    }
    private void btnSairActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }
    private void enter(java.awt.event.KeyEvent evt) {...2 linhas }
    public static void main(String args[]) {
        Look and feel setting code (optional)
            java.awt.EventQueue.invokeLater(new Runnable() {...5 linhas });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton btnSair;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLblFuncoLogin;
    private javax.swing.JLabel jbTelaLogin;
    private javax.swing.JPasswordField txtsenha;
    private javax.swing.JTextField txtuser;
    // End of variables declaration
}
```

Figura 53: Método de consulta dos dados do banco de dados para efetuar o login.