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

TAMIRES ALVES DA SILVA

SISTEMA WEB PARA GESTÃO DE EVENTOS UTILIZANDO TECNOLOGIAS JAVA E GOOGLE MAPS API

Assis
2011

TAMIRES ALVES DA SILVA

SISTEMA WEB PARA GESTÃO DE EVENTOS UTILIZANDO TECNOLOGIAS JAVA E GOOGLE MAPS API

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação.

Orientador: Drº Almir Rogério Camolesi

Área de Concentração: Desenvolvimento de Sistemas

Assis
2011

FICHA CATALOGRÁFICA

SILVA, Tamires Alves da
Sistema web para gestão de eventos utilizando tecnologias Java e Google Maps API / Tamires Alves da Silva. Fundação Educacional do Município de Assis, 2011.
106 p.

Orientador: Dr. Almir Rogério Camolesi
Trabalho de Conclusão de Curso
Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Gestão de Eventos 2. Java 3. Google Maps API

CDD: 001.61
Biblioteca da Fema

SISTEMA WEB PARA GESTÃO DE EVENTOS UTILIZANDO TECNOLOGIAS JAVA E GOOGLE MAPS API

TAMIRES ALVES DA SILVA

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

Orientador: Drº Almir Rogério Camolesi

Analisador : Drº Alex Sandro Romeo de Souza Poletto

Assis
2011

DEDICATÓRIA

Dedico este trabalho à minha família, amigos e todas as pessoas que acreditaram em meus sonhos e anseios, apoiando-me com força necessária para que pudesse realizá-los.

AGRADECIMENTOS

Primeiramente a Deus, é a ele que dirijo minha maior gratidão, pois sem Ele, nada seria possível. Dele que vem tudo que sou, o que tenho e espero.

Aos meus familiares, Solange Maria Alves Cesar, Lais Alves da Silva e Jéssica Alves da Silva, por estarem sempre do meu lado, apoiando-me e conduzindo-me durante minha caminhada.

Ao meu namorado e amigo Guilherme de Cleva Farto pelo seu amor, compreensão, ajuda e companheirismo em todos os momentos, deixando os meus dias mais felizes e belos.

Ao meu orientador Drº Almir Rogério Camolesi, pela orientação, não somente durante este trabalho, mas por toda a minha vida acadêmica.

E a todos que colaboraram direita ou indiretamente na execução deste trabalho.

“Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes.”

Isaac Newton

(1643-1727)

RESUMO

Cada vez mais, a área de tecnologia da informação tem apresentado um grande crescimento no Brasil e no mundo. O uso de *softwares* específicos tem contribuído para esse aumento, devido à necessidade de um maior controle e levantamento de dados de processo de produção.

Este trabalho apresenta a especificação e implementação de um aplicativo *web* destinado a empresas gestoras de eventos, proporcionando soluções inovadoras e criativas, agilizando processos, reduzindo despesas e ampliando a comunicação entre os participantes. Para isso, serão utilizadas duas ferramentas: a rede social *Twitter* e o mecanismo de localização *Google Maps*.

Para o desenvolvimento deste trabalho foi realizado um estudo sobre tecnologias a serem utilizadas para a implementação do sistema, como a linguagem de programação Java, o banco de dados *Oracle Express Edition*, o *framework* de persistência Hibernate, o servidor de aplicação *Apache Tomcat*, a plataforma RIA de *Adobe Flash Builder 4* , a *API Google Maps* e *API Twitter4J*.

ABSTRACT

Increasingly, the area of information technology has experienced tremendous growth in Brazil and around the world. The use of specific software has contributed to this increase due to the need for greater control and data collection process of production.

This paper presents the specification and implementation of a web application designed for event management companies, providing innovative and creative solutions, streamlining processes, reducing costs and increasing communication among participants. To do this, will use two tools: the social network Twitter and Google Maps location engine.

To develop this work was carried out a study on the technologies being used to implement the system, such as the programming language Java, Oracle Database Express Edition, the Hibernate persistence framework, the Apache Tomcat application server, the platform RIA Adobe Flex Builder 4, the Google Maps API and Twitter4J API.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
FTP	File Transfer Protocol
HQL	Hibernate Query Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
JSP	Java Server Pages
JVM	Java Virtual Machine
KML	Keyhole Markup Language
MVC	Model Viewer Controller
MXML	Macromedia Extensible Markup Language
ORM	Object Relational Mapping
SQL	Structured Query Language
SDK	Software Development Kit
SWF	ShockWave Flash
UML	Unified Modeling Language
WBS	Work Breakdown Structure
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A forma de criar, armazenar e compartilhar informações mudou muito nos últimos anos, tudo isso, graças aos avanços tecnológicos.

Atualmente, a informática é um dos campos de trabalho de maior crescimento no Brasil e no mundo. As empresas procuram cada vez mais *softwares* específicos para auxiliar no controle e no levantamento de dados de processos de produção.

Visando essas necessidades, o *software SuperEvent* vem disponibilizar, para o mercado, uma aplicação para a gestão de eventos de pequeno a médio porte, agilizando, simplificando e organizando os dados, iniciando-se na etapa de orçamento até a execução da atividade.

O *SuperEvent* é o sistema *Web* ideal para auxiliar empresas que fornecem, como negócio principal, a prestação de serviços para a organização e realização de eventos de pequeno e médio portes, tais como festas infantis, formaturas, casamentos, reuniões, entre outras cerimônias.

O sistema apresenta como principais funcionalidades interfaces para cadastro de tipos de evento, eventos, clientes, fornecedores, funcionários, produtos e equipamentos, elaboração de orçamentos, controle de estoque, organização de agendas, localização dos eventos utilizando o *GoogleMaps API*, além de informações, em tempo real, dos acontecimentos do evento por meio do *Twitter4J API.*

Os recursos disponibilizados pelo *SuperEvent*proporcionam facilidades que contribuem para o sucesso do evento.

1.1 OBJETIVO

Como objetivo principal, o *software SuperEvent* será desenvolvido para o mercado de gerenciamento de eventos utilizando soluções inovadoras e criativas, agilizando processos, reduzindo despesas, ampliando a comunicação entre os participantes, por meio do uso de uma das mais utilizadas ferramentas da rede social, o *Twitter*, e

a utilização do mecanismo de localização, *Google Maps*. Tudo de forma flexível, facilitando o controle das tarefas antes, durante e após o evento, satisfazendo as necessidades dos clientes.

1.2 JUSTIFICATIVA

A justificativa para que o *SuperEvent* seja um *software* para *Web* se dá pela forma extraordinária de crescimento nos últimos anos do serviço *Web*, o qual proporciona enormes vantagens e facilidades para com seus usuários.

Levantamentos revelam que o Brasil é o quinto maior do mundo, atrás apenas dos Estados Unidos, China, Alemanha e Rússia em relação ao uso de redes sociais (COMSCORE, 2010) e o segundo país que mais cresce em acessos ao *Twitter (*GOMES, 2011) por isso a integração do *SuperEvent* com esta ferramenta.

Além disso, empresas buscam por *softwares* seguros, modernos e precisos para auxiliarem seus negócios, reduzindo custos e elevando a qualidade de seus serviços, para tanto, o *software SuperEvent* fará uso das tecnologias Java e do banco de dados Oracle, por estar presente entre os melhores e mais conceituados bancos de dados, objetivando a segurança de informações. Já usuários comuns buscam por fácil navegação, visual moderno, elegante e por variedade de recursos, presentes em tecnologias e ferramentas como *Google Maps* para controle de localização geográfica, atualizações de informações em tempo real por meio da rede social *Twitter* e desenvolvimento de aplicações ricas para *Web* com *Flash Builder*.

Contudo, o desenvolvimento do *software SuperEvent* atende a essas exigências e acompanha as evoluções tecnológicas com a finalidade de competir com este nicho de mercado. Além de o presente trabalho contribuir para com meu enriquecimento acadêmico e profissional.

1.3 PÚBLICO ALVO

O sistema desenvolvido atende as necessidades de empresas relacionadas à organização de eventos de pequeno a médio porte e empresários interessados nessa tecnologia.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido em capítulos que serão explicitados a seguir.

O primeiro capítulo apresenta a contextualização e a justificativa para o desenvolvimento da proposta do trabalho.

A seguir, no segundo capítulo serão abordados os conceitos de fundamentação teórica das tecnologias utilizadas para o desenvolvimento do sistema.

O terceiro capítulo apresenta as etapas de análise e especificações do sistema contemplando o levantamento de requisitos, lista de eventos, o caso de uso e suas especificações e os principais diagramas da UML (classe, sequência e atividade).

No quarto capítulo descreve a *Work Breakdown Structure* (WBS), o sequenciamento das atividades e orçamento do sistema.

A implementação do sistema, exibindo um detalhamento sobre a aplicação desenvolvida assim como a distribuição das camadas, organização de pacotes e a interface criada para interagir com o usuário final será apresentado no Capítulo 5. Por fim, no sexto e último capítulo serão apresentados a conclusão e trabalhos futuros.

2 TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

Neste capítulo serão apresentadas as tecnologias e ferramentas utilizadas no desenvolvimento do sistema *SuperEvent.*

O *SuperEvent* é um sistema *Web* desenvolvido com a tecnologia Java, que possui uma linguagem de alto nível orientada a objetos. O ambiente de desenvolvimento é baseado na IDE Eclipse, permitindo o acoplamento de novos componentes, facilitando a integração de novas bibliotecas e APIs. A arquitetura do projeto faz uso do padrão *Model Viewer Controller* (MVC), responsável por separar as regras de negócio das camadas de visualização e de controle, ocasionando uma alta reusabilidade de código e uma maior facilidade nos momentos de manutenção do projeto.

O servidor de aplicação *Tomcat* da *Apache* é robusto, leve e apresenta uma interface amigável para publicar e gerenciar aplicações *Web*. Para servir de repositório de informações, será utilizado o banco de dados *Oracle 10g Express Edition*, o qual responderá a instruções de seleção e manipulação de dados por meio do *framework Hibernate*, utilizado para realizar o mapeamento objeto/relacional, facilitando as operações na camada de persistência da aplicação.

O *Plug-in* da *Adobe Flash Builder 4*, tecnologia muito utilizada para criar a camada de visualização, suportando o desenvolvimento de aplicações ricas baseadas na Internet, juntamente com a *API Google Maps*, responsável pela localização dos eventos cadastrados no sistema e pelas informações geográficas relacionadas e a *API Twitter4J*, responsável por integrar o sistema com uma das maiores redes sociais, o *Twitter*.

E finalmente para a avaliação gerencial dos dados, será utilizada a ferramenta *iReport*, visando facilitar a construção de relatórios por meio da biblioteca *JasperReport*, além de definir *designs* modernos através de uma interface gráfica intuitiva, sem a necessidade de se escrever linhas de código no formato de XML (*eXtensible Markup Language*) (JAVAFREE, 2011).

2.1 JAVA

O Java não é apenas uma linguagem de programação orientada a objetos, é também uma plataforma completa de desenvolvimento e ambiente de execução criada pela Sun, sendo lançada publicamente em 1995. Essa plataforma teve seu início em 1991 sob o nome *Oak* por um time de desenvolvedores da *Sun Microsystems* liderados por James Gosling, considerado o pai do Java. O foco inicial do projeto Oak era dispositivos eletrônicos, portanto o desenvolvimento da plataforma foi bastante concentrado na segurança, pois caso a aplicação parasse, não deveria comprometer o restante do dispositivo (JAVA, 2011).

O Java é uma solução que facilita o desenvolvimento de aplicativos portáveis fazendo uso de uma linguagem de programação simplificada, segura, orientada a objetos, com uma extensa API e de um ambiente de execução portável(JAVA, 2011).

Desde seu lançamento, a plataforma Java é considerada, na história da computação, a linguagem de programação que foi adotada mais rapidamente. Até hoje, ela já atraiu mais de 6,5 milhões de desenvolvedores de software em todo o mundo e é executado em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos, inclusive em telefones celulares e dispositivos de televisão. Java tornou-se referência no mercado de desenvolvimento de *software* e continua crescendo (JAVAFREE, 2011) e (JAVA, 2011).

Segundo o índice TIOBE, de maio de 2011, Java é a linguagem mais popular para o desenvolvimento de sistemas. É importante observar que este índice não se refere a melhor linguagem de programação ou a linguagem com a qual se escreveu a maior quantidade de linhas de código até o momento. Este é um índice que leva em consideração, entre outras coisas, a quantidade de profissionais capacitados ao redor do mundo, cursos disponíveis e produtos no mercado (TIOBE, 2011). A Figura 1 ilustra o gráfico TIOBI de outubro de 2011:

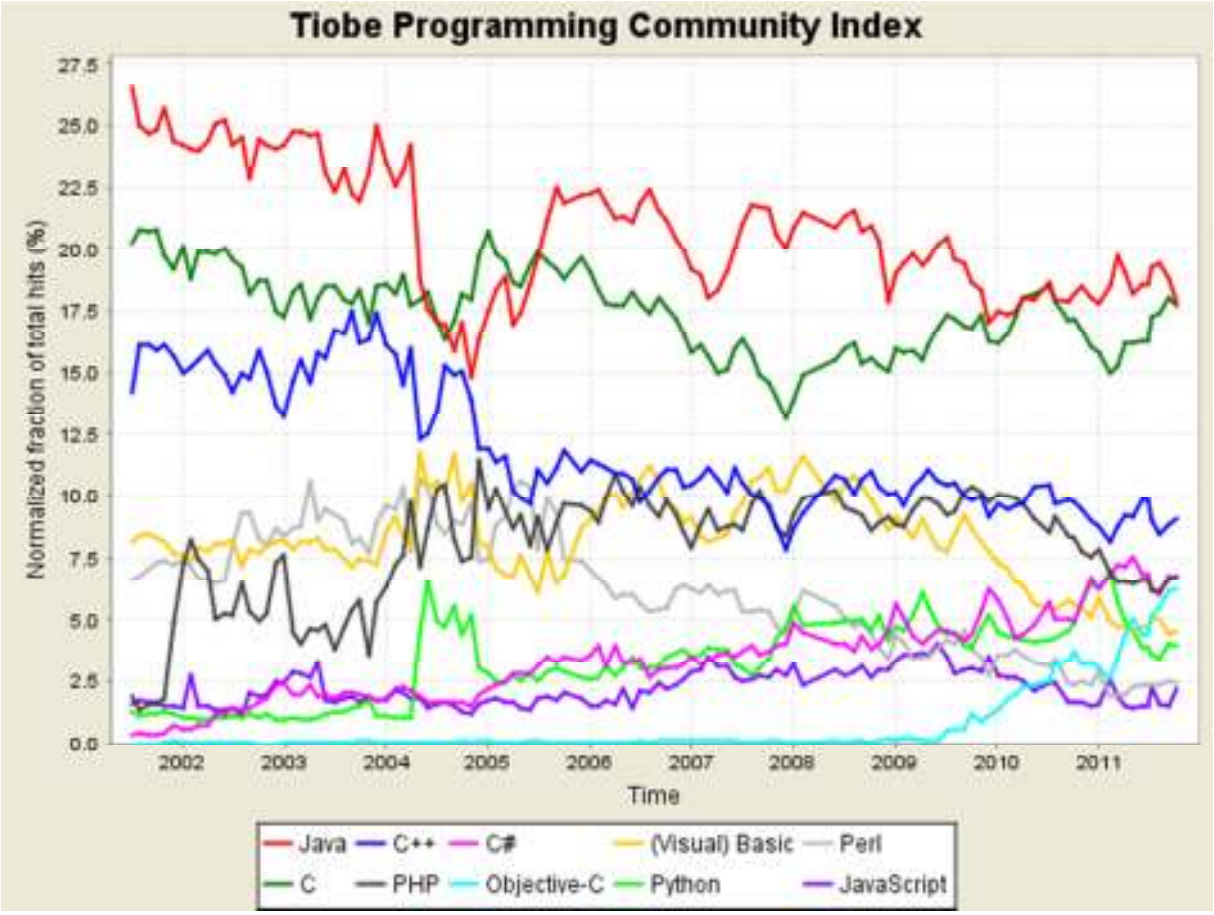

Figura 1 Java é a linguagem mais popular. Fonte: (TIOBE, 2011)

2.1.1 Principais características do Java

A linguagem Java é considerada de alto nível, com uma sintaxe extremamente similar à do C++ e com diversas características herdadas de outras linguagens, como *Smalltalk* e *Modula-3*, além de apresentar outras características, conforme apresentado nos tópicos abaixo:

- SIMPLICIDADE: É muito parecida com C++, porém mais simples, por não possuir sobrecarga de operadores, *structs*, *unions*, aritmética de ponteiros, herança múltipla, arquivos .h, diretivas de pré-processamento, além de que a memória é alocada dinamicamente, sendo gerenciada pela própria linguagem, fazendo uso de um coletor de lixos, ou *garbage collection*, para liberar regiões de memória que não estão mais em uso.

- ORIENTAÇÃO A OBJETOS: É uma linguagem orientada a objetos que segue a linha purista iniciada por *Smalltalk*. Com a exceção dos tipos básicos da linguagem, como *int*, *float*, *double*, *long*, entre outros, a maior parte dos elementos de um programa Java são objetos. O código é organizado em classes, podendo estabelecer relacionamentos de herança simples entre si.

- PROCESSAMENTO DISTRIBUÍDO: Chamadas a funções de acesso remoto por meio de *sockets* e os protocolos Internet mais comuns, como HTTP, FTP e *Telnet*, são suportadas em Java, de forma que a elaboração de aplicativos baseados em arquiteturas cliente-servidor é facilmente obtida.

- *MULTITHREADING*: A maior parte dos sistemas operacionais hoje no mercado dá suporte à multitarefa, como o *Windows*, *OS/2* e *Unix*, ou seja, o computador é capaz de executar diversas tarefas ao mesmo tempo.

- EXCEÇÕES: A máquina virtual Java faz uma verificação em tempo de execução quanto aos acessos de memória, abertura de arquivos e uma série de eventos que podem comprometer o desempenho do computador, gerando exceções que podem ser tratadas em programas Java.

- *GARBAGE COLLECTOR*: Em Java, os programadores não necessitam preocupar-se com o gerenciamento de memória, pois o recurso do *garbage collector* exerce a funcionalidade de varrer a memória de tempos em tempos, liberando automaticamente os blocos que não estão sendo utilizados.

- *MACHINE INDEPENDENT*: Uma das características de Java que a tornou ideal para seu uso na elaboração de aplicativos distribuídos é a sua independência de plataforma. A tecnologia Java consegue essa independência devido ao fato de que o compilador não gera instruções específicas a uma plataforma, bastando possuir uma máquina virtual para cada sistema operacional (DEITEL; DEITEL, 2005).

2.1.2 Máquina Virtual Java

Para atingir a questão da portabilidade, o Java utiliza-se a ideia de uma máquina virtual que executa um código de máquina próprio. Essa máquina virtual traduz os comandos para a plataforma específica e, com isso, se ganha independência de plataforma. Logo após seu lançamento, o Java fez um enorme sucesso com seus *Applets*, pequenas aplicações embutidas em páginas *Web* e executadas no navegador do cliente. Nessa época, nasceu o famoso *slogan* "*Write once, run anywhere*", pois qualquer navegador em qualquer sistema operacional rodaria a mesma aplicação sem necessidade de alterações com a ajuda da máquina virtual.

A Java Virtual Machine ou JVM é uma *Application Virtual Machine*, que abstrai não só a camada de hardware como a comunicação com o Sistema Operacional. A JVM ao invés de gerar um executável específico, como *Linux* ou *Windows*, o compilador Java gera um executável para uma máquina genérica, uma máquina virtual, não física, a JVM.

A JVM é uma máquina completa que é executada em uma camada superior a camadade hardware (máquina real). Ela possui suas próprias instruções de máquina, ou seja, em *Assembly*, e suas APIs próprias. O papel da máquina virtual é executar as instruções genéricas de máquina, conhecidas como *bytecodes*, no Sistema Operacional e no *hardware* específico sob o qual está operando.

A Figura 2 ilustra o processo de compilação e interpretação de um aplicativo desenvolvido na linguagem Java:

Figura 2 Processo de compilação e interpretação do Java

É importante perceber que, caso o usuário final deseja apenas executar o programa, e não desenvolver, deverá também fazer uso dos conceitos da JVM, ou seja, tanto os desenvolvedores quanto os usuários necessitam de uma máquina virtual. Para o ambiente de desenvolvimento, além da JVM, é preciso o compilador e outras ferramentas. Por isso que o Java possui duas distribuições diferentes: o JDK ou *Java Development Kit*, um *kit* focado no desenvolvedor e traz, além da VM, compilador e outras ferramentas úteis; e o JRE ou *Java Runtime Environment*, trazendo apenas o necessário para se executar um aplicativo Java, incluindo a VM e APIs, sendo voltada ao usuário final (DEITEL; DEITEL, 2005).

2.1.3 Plataforma Java

A plataforma Java é dividida em três segmentos ou edições principais: Java SE, Java EE e Java ME. Cada edição engloba e licencia um conjunto de APIs e um ambiente de execução da plataforma Java para atender às necessidades específicas dos desenvolvedores de aplicações.

- Java SE ou *Java Platform, Standard Edition*: É o segmento base da plataforma Java por incluir o ambiente de execução, a JVM, o compilador

Java e as bibliotecas comuns (ORACLE, 2011). A Figura 3 ilustra a plataforma Java SE:

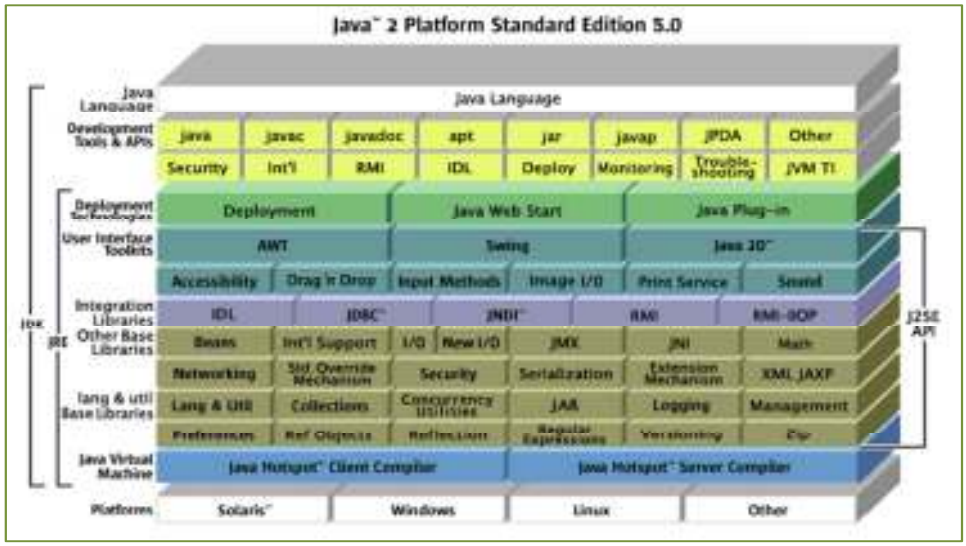

Figura 3 Plataforma Java SE. Fonte: (ORACLE, 2011)

- Java EE ou *Java Enterprise Edition*: É todo o *core* contido em JSE expandindo para aplicações mais robustas, multicamadas e que geralmente executam em um servidor de aplicação. A plataforma JEE contém uma série de especificações, cada uma com funcionalidades distintas e complementares. Entre elas, destacam-se:
    - JDBC ou *Java Database Connectivity* é utilizado no acesso a diferentes servidores de banco de dados;
    - *Servlets* são utilizados para o desenvolvimento de aplicações *Web* com conteúdo dinâmico. Contém uma API que disponibiliza os recursos do servidor *Web* de maneira simplificada para o desenvolvedor.
    - *Enterprise JavaBeans* são utilizados no desenvolvimento de componentes de *software*, permitindo que o programador se concentre nas necessidades do negócio do cliente, enquanto questões de infra estrutura, segurança e outros aspectos são de responsabilidade do servidor de aplicação (SUN, 2011).

A Figura 4 ilustra a plataforma Java EE:

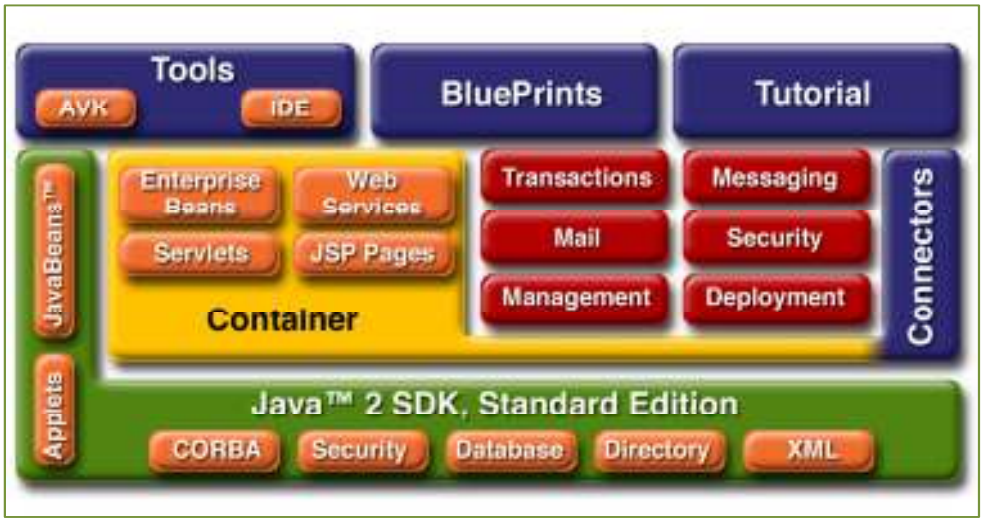

Figura 4 Plataforma Java EE. Fonte: (SUN, 2011)

- Java ME ou Java Micro Edition: É um segmento da plataforma Java destinado para o desenvolvimento de aplicações para dispositivos móveis, PDAs e outros dispositivos embarcados (SUN, 2011).

A Figura 5 ilustra a plataforma Java ME:

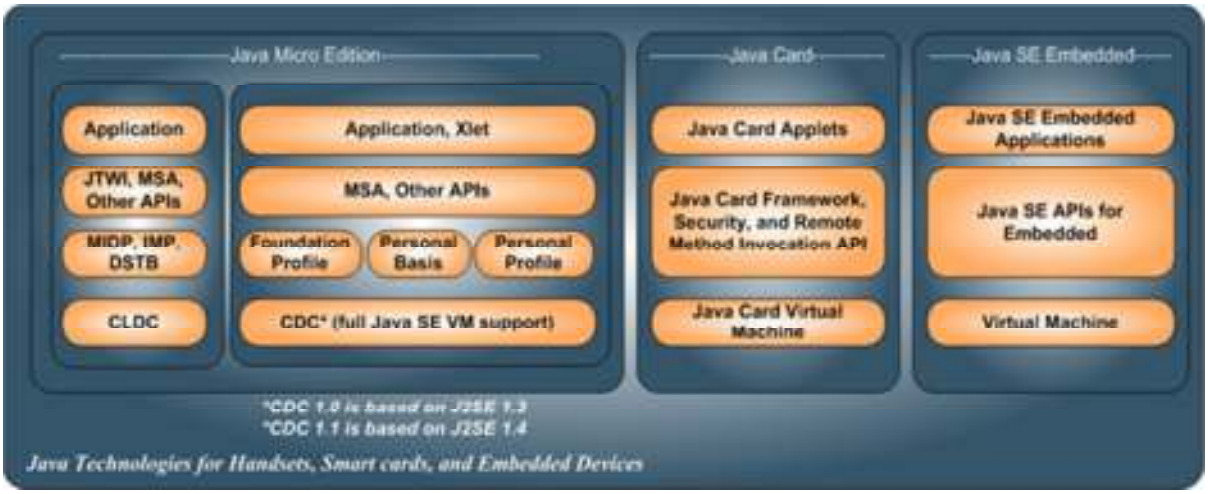

Figura 5 Plataforma Java ME. Fonte: (SUN, 2011)

Além disso, podem-se destacar outras duas plataformas Java mais específicas:

- Java *Card*: especifica para dispositivos embarcados com limitações de processamento e armazenamento, como *smart cards* e o Java *Ring*.

- JavaFX**:** plataforma para desenvolvimento de aplicações multimídia em *desktop/web*, como JavaFX Script, e dispositivos móveis, JavaFX Mobile (SUN, 2011).

2.2 ORACLE

Em agosto de 1977, foi fundada, por Larry Ellison, Bob Miner e Ed Oates, a empresa de consultoria *Software Development Laboratories* (SDL), a qual mais tarde passou a se chamar Oracle, grupo responsável por inovar o mercado com o primeiro modelo de banco de dados relacional, transformando a face da computação empresarial.

Um modelo de banco de dados relacional consiste em uma coleção de tabelas, cada uma com um nome único atribuído, para representar tanto os dados quanto as relações entre eles. Esse modelo conquistou a posição de destaque devido a sua simplicidade. Um banco de dados é uma coleção de dados persistentes, utilizada pelos sistemas de aplicação (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

Um sistema de gerenciamento de banco de dados ou SGBD é um sistema computadorizado cuja finalidade resume-se em armazenar uma coleção de dados inter-relacionados, permitindo ao usuário acessá-las e atualizá-las quando solicitado.

Os SGBDs estão presentes, atualmente, em toda parte e seu principal objetivo é fornecer um ambiente que seja conveniente e eficiente para com seus usuários, pois, a maioria das pessoas interage, direta ou indiretamente, com banco de dados muitas vezes todos os dias (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

O Oracle é o principal banco de dados, sendo responsável pelo armazenamento de boa parte das informações de grandes organizações ao redor do mundo. Um ponto muito alto do banco de dados Oracle é a segurança, principalmente quando bem gerenciado (SILBERSCHATZ; KORTH; SUDARSHAN, 2006)..

Atualmente, a Oracle continua à frente de seu tempo. Ela é a principal fornecedora de *software* para gerenciamento de informações e a segunda maior empresa independente do mundo. Suas tecnologias podem ser encontradas em quase todos os setores do mundo inteiro (ORACLE, 2011).

2.2.1 Oracle Database 10g Express Edition

O Oracle *Database 10g Express Edition* é uma versão gratuita do mais capaz banco de dados relacional do mundo. Ele é baseado no Oracle *Database 10g Release 2*. É livre e simples de instalar, gerenciar, implementar e distribuir. O Oracle *Database XE* disponibiliza uma interface intuitiva baseada em navegador para administrar o banco de dados, criar tabelas, *views* e outros objetos de banco, além de importar e exportar dados.

É um conceituado banco de dados entre desenvolvedores, administradores de banco de dados, vendedores de *software* independentes, estudantes e instituições educacionais, podendo ser instalado em um computador com qualquer número de CPUs, podendo armazenar até 4GB de dados do usuário, utilizando-se 1GB de memória e uma CPU na máquina *host*.

O Oracle disponibiliza, para sua versão *free*, um suporte através do fórum de discussão *free Oracle*, o qual é monitorado por funcionários da Oracle e especialistas da comunidade (ORACLE, 2011).

2.3 HIBERNATE

O Hibernate é um *framework* de mapeamento objeto/relacional para Java que facilita o desenvolvimento de aplicações e da independência de bancos de dados, fazendo com que o programador se preocupe mais com o seu modelo de objeto e seus comportamentos do que com tabelas.

O mapeamento objeto/relacional ou apenas ORM é a persistência automatizada dos objetos em uma aplicação Java para as tabelas de um banco de dados relacional, utilizando metadados que descrevem o mapeamento entre os objetos e o banco de dados.

Além do mecanismo de mapeamento objeto/relacional, o Hibernate também pode trabalhar com um sistema de *cache* das informações do banco de dados, aumentando ainda mais a desempenho das aplicações. O esquema de *cache* do

Hibernate é complexo e totalmente extensível, existindo diversas implementações possíveis, cada uma com suas próprias características.

O Hibernate possui a sua própria linguagem de *Structured Query Language* (SQL) chamada *Hibernate Query Language* (HQL), que é convertida para SQLs específicas de cada banco de dados, características que faz dele um dos principais *frameworks* para independência de banco de dados.

O arquivo de mapeamento é um arquivo no formato *Extensible Markup Language* (XML) que define as propriedades e os relacionamentos de uma classe para o Hibernate, este arquivo pode conter classes, classes componentes e *queries* em HQL ou em SQL. É uma boa prática usar um arquivo de mapeamento para cada classe e usar como extensão do arquivo ".hbm.xml" para diferenciar de outros arquivos XML utilizados na aplicação.

O Hibernate também evita o trabalho de escrever diversas linhas de código repetido para fazer as mesmas coisas, como *inserts, selects, updates* e *deletes* no banco de dados, além de ter duas opções para se montar buscas complexas em grafos de objetos e ainda uma saída para o caso de nada funcionar, utilizando SQL (BAUER; KING, 2007) e (JAVAFREE, 2011).

2.4 APACHE TOMCAT

O *Tomcat,* desenvolvido pela Fundação Apache e distribuído como *software* livre dentro do conceituado projeto *Apache Jakarta*, é um servidor de aplicações centrado na linguagem de programação Java, mais especificamente nas tecnologias de *Servlets* e de *Java Server Pages* (JSP). Ele tem a habilidade de converter automaticamente qualquer página JSP em um *servlet* equivalente. Em outras palavras, o Tomcat é capaz de criar documento HTML a partir de código fonte Java.

O *Tomcat* é robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção. Ele tem a capacidade de atuar também como servidor *Web*, ou pode funcionar integrado a um servidor *web* dedicado como o *Apache* ou o IIS. Como servidor *web*, ele provê um servidor *web* HTTP puramente em Java(APACHE TOMCAT, 2011).

Do ponto de vista técnico, *Tomcat* é a implementação das tecnologias de *servlets* e JSP. Do ponto de vista operacional, a principal finalidade das tecnologias de s*ervlets* e JSP é permitir a criação dinâmica de conteúdos. A dinâmica, em um cenário típico, funciona do seguinte modo:

- Um usuário, no seu *browser*, solicita algum documento, indicado por um URL, a um servidor *Tomcat*;
- O servidor, ao receber uma solicitação URL do usuário, executa o *servlet* ou JSP correspondente àquele URL (a associação entre URL e servlet ou JSP é especificada no arquivo "*web.xml*"). O conteúdo gerado pelo *servlet* ou JSP, normalmente um documento no formato HTML, é uma combinação de *tags* HTML e o resultado de algum processamento.
- O usuário recebe o conteúdo gerado pelo servidor *Tomcat* e o exibe através do seu *browser* (APACHE TOMCAT, 2011).

2.5 ADOBE FLEX

O *Adobe Flex* é um *framework* de desenvolvimento de aplicações ricas para Internet (RIA), possuindo uma estrutura produtiva, gratuita e de código aberto para a criação de aplicativos expressivos para dispositivos móveis, *web* e *desktop*. Ele permite criar aplicativos móveis e *web* compartilhando a mesma base de código, reduzindo o tempo de custo da criação de aplicativos e manutenção de longo prazo. Enquanto os aplicativos *Flex* podem ser criados utilizando-se apenas o SDK gratuito, a ferramenta de desenvolvimento *Adobe Flash Builder*, baseada no *Eclipse*, acelera o desenvolvimento através de recursos, como edição de código inteligente, depuração interativa por etapas, geradores de perfis de memória e desempenho, e design visual.

O MXML é uma linguagem de marcação, baseada em XML utilizada para construir a interface com o usuário nas aplicações construídas com *Flex*, essas características fazem com que a linguagem MXML se torne mais estruturada e menos ambígua que

o HTML, fornecendo componentes mais ricos que HTML e é renderizada no *Flash Player*. O *ActionScript*, uma linguagem orientada por objetos, é a linguagem usada para criar a lógica do aplicativo. O MXML e o *ActionScript* são compilados juntos em um único arquivo no formato SWF que constitui o aplicativo *Flex*. Como o compilador está disponível como um utilitário independente no SDK do *Flex* 4 e como parte do *software Adobe Flash Builder* 4, antigo *Adobe Flex Builder*, os desenvolvedores podem optar por desenvolver no IDE do *Flash Builder* com base no *Eclipse* ou em um IDE de sua preferência.

O *Flex* também inclui uma biblioteca de classe pré-criada e serviços de aplicativos que ajudam os desenvolvedores a montarem e criarem aplicativos usando mais de 100 componentes avançados e pré-criados, além de novos recursos. Esses serviços incluem o vínculo de dados, o gerenciamento de arrastar e soltar, o sistema de exibição que gerencia o *layout* da interface, o sistema de estilo que gerencia a aparência dos componentes de interface e o sistema de efeitos e animação que gerencia o movimento e as transições (ADOBE, 2011).

2.6 BLAZEDS

O *BlazeDS* é um projeto gratuito que inclui recursos remotos e de mensagem. Como BlazeDS é de código aberto, facilita a conexão de aplicativos *Flex* e *AIR* ao *back-end*, aos dados distribuídos e à infra-estrutura de servidor Java proporcionando aos usuários experiências receptivas, em tempo real, orientadas por dados, implantadas no navegador ou no *desktop* (ADOBE, 2011).

2.7 GOOGLE MAPS

O *Google Maps*, serviço gratuito fornecido pela empresa *Google*, foi lançado, em fevereiro de 2005 tornando-se rapidamente uma referência em serviços de mapas na Internet. Com uma interface rica, intuitiva e interativa, a aplicação permite pesquisar e visualizar mapas e imagem de satélite da Terra, além de traçar rotas. No

Google Maps, com um simples clique o usuário pode dar uma volta virtual pelo mundo, desfrutando de vários recursos.

A *Google*, além de oferecer os serviços do *Google Maps* para usuários comuns disponibiliza o *Google Maps* API para todos aqueles interessados em usar este serviço e integrá-lo em sua própria página, de forma simples e gratuita permitindo incorporar funcionalidade robusta e de grande utilidade do *Google Maps* em sites e aplicativos, e adicionar dados sobre os mapas. A API de dados do *Google Maps* usa o conhecido padrão KML para representar recursos nos mapas, incluindo marcadores, polilinhas e polígonos (GOOGLE DEVELOPER GUIDE, 2011).

2.8 TWITTER

O Brasil é considerado o país mais sociável do mundo. Segundo o Ibope NetRatings, mais de 80% dos internautas têm perfis em redes sociais. A média de amigos virtuais no mundo é de 195 pessoas por usuário, no Brasil, é de 365. Segundo a pesquisadora Raquel Recuero, a participação em redes sociais é igual em todas as classes sociais, além de ser o primeiro contato dos jovens com a internet (REVISTA ÉPOCA, 2010). A Figura 6 ilustra quais redes sociais os brasileiros se encontram.

Figura 6 O Brasil é o país mais sociável da internet. Fonte: Revista Época, 2010

Segundo pesquisa divulgada em agosto de 2010 pela comScore, empresa de análise do mercado digital, o Brasil é o 5º país que mais acessa redes sociais no mundo (COMSCORE, 2010). A Figura 7 ilustra o ranking mundial de acesso a redes sociais.

Figura 7 Ranking mundial de acesso a redes sociais. Fonte: comScore, 2010

E uma das redes sociais mais conhecidas atualmente é o *Twitter*, criativo e inteligentemente, diferente das mais populares redes, oferece um tipo de serviço diferenciado conhecido com *microblog*. No *Twitter* você escreve o que está pensando ou se passando em sua vida ou em sua cabeça, e outras pessoas gostam do que você está escrevendo e passam a te seguir. Lançado em 2006 por Jack Dorsey, o *Twitter* já alcançou 200 milhões de usuários registrados e tem uma média de 100 milhões de *tweets* diários (REVISTA ÉPOCA, 2010). A Figura 1 apresenta o gráfico, liberado pela Pingdom em fevereiro de 2011, o qual ilustra as redes sociais que recebem mais de 1 milhão de visitantes por dia (PINGDOM, 2011).

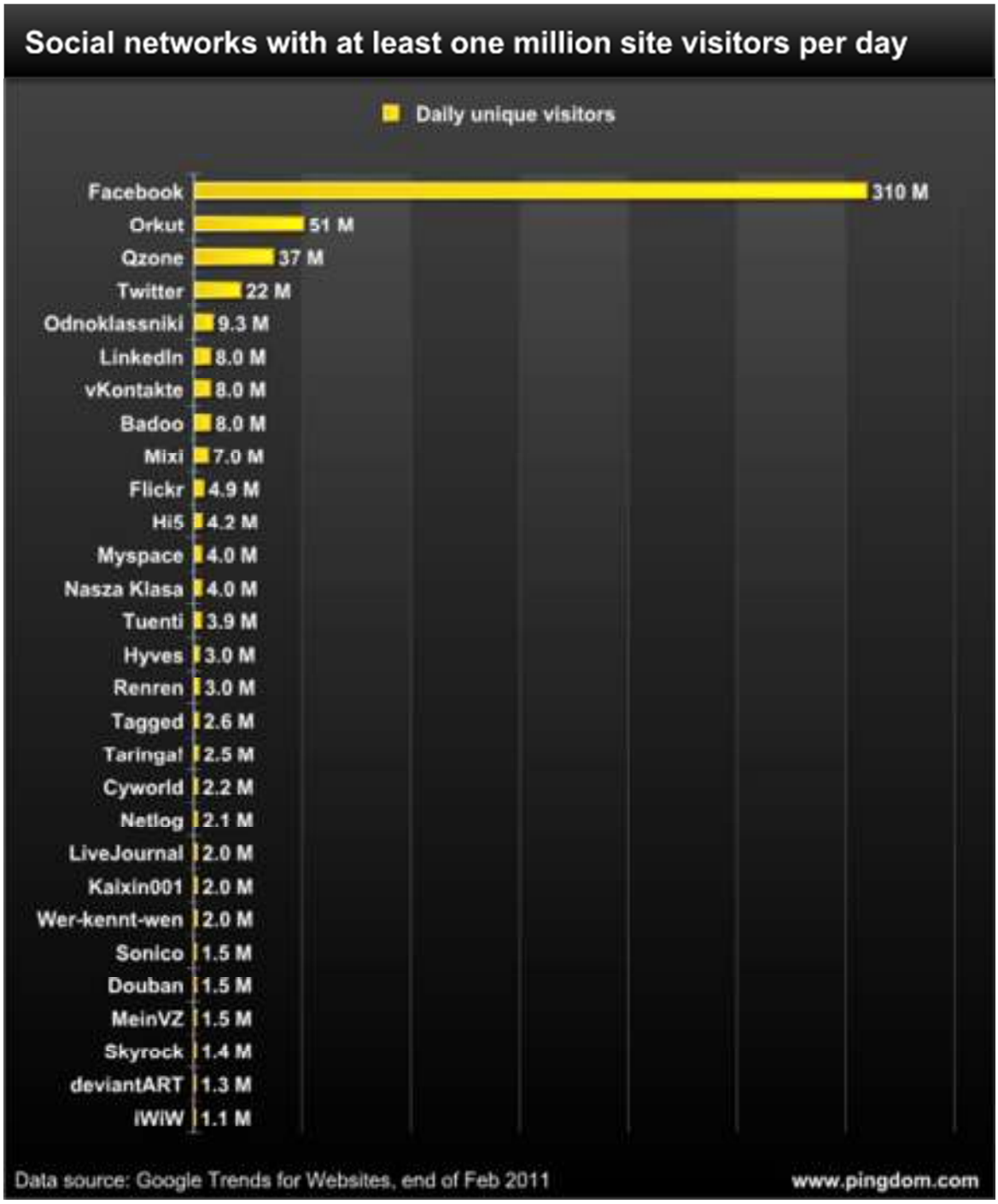

Figura 8 Redes sociais que recebem mais de um milhão de visitantes por dia. Fonte: Pingdom, 2011

2.8.1 Twitter4J API

O Twitter4J é uma biblioteca Java não oficial. Esta API é de código-fonte aberto e gratuito, o qual pode ser usado livremente, em projetos comerciais e não comerciais,

integrando facilmente aplicações Java com o serviço do *Twitter*. O Twitter4J é liberado sob licença Apache 2.0 (TWITTERJ4, 2011).

2.9 PADRÃO DE PROJETO MVC

Um padrão de projeto nomeia, abstrai e identifica os aspectos-chaves de uma estrutura de projeto comum, tornando-a útil para a criação de um projeto orientado a objetos de maneira reutilizável.

O padrão de projeto *Model-View-Controller* (MVC) é composto por três tipos de objetos. O modelo é o objeto de aplicação, a visão é a apresentação na tela e o controle define a maneira como a interface do usuário reage às entradas do mesmo.

- *Model* ou modelo: As classes que trabalham no armazenamento e busca de dados representam as entidades do programa. Por exemplo, um cliente pode ser modelado em uma aplicação, podendo haver vários modos de criar novos clientes ou mudar informações de um relativo cliente.
- *View* ou visão: Maneja a apresentação visual dos dados representados pelo modelo. Em resumo, é a camada responsável por apresentar os dados resultantes do Modelo ao usuário.
- *Controller* ou controlador: Objeto que responde as ordens executadas pelo usuário, atuando sobre os dados apresentados pelo modelo, decidindo como o modelo deverá ser alterado ou revisto e qual apresentação deverá ser exibida.

O padrão de projeto MVC é uma forma de desenvolvimento que ajuda na manutenção do sistema, sendo um padrão muito aceito durante a implementação de aplicações Java, principalmente em aplicações *web* (GONÇALVES, 2008).

2.10 IREPORT

A ferramenta *iReport* visa facilitar a construção de relatórios, desde o mais simples ao mais complexo para aplicações Java, por meio da biblioteca *JasperReport*, além de definir *designs* modernos através de uma interface gráfica intuitiva, sem a necessidade de se escrever linhas de código no formato de XML. O *iReport* pertence ao programa *Open Source*, distribuindo gratuitamente seus códigos fontes, de acordo com o *General Public License* ou GNU. O *JasperReports* é um poderoso *framework open-source* escrito em Java para geração de relatórios, possibilitando a criação de relatórios em diversos formatos; entre eles: PDF, HTML, XLS, CSV e XML (GONÇALVES, 2008).

3 ANÁLISE DE ESPECIFICAÇÃO DO SISTEMA

Este capítulo apresenta informações das ferramentas utilizadas para a modelagem e análise do sistema proposto.

3.1 MAPA MENTAL

Para melhor entendimento e visualização do sistema é apresentado o mapa mental. Mapa mental ou *Mind Map* é um método utilizado para armazenar, organizar e priorizar informações através de diagramas elaborados a partir de uma ideia principal irradiando informações relacionadas. Tony Buzan, o inglês considerado o inventor do mapa mental afirma que os mapas mentais funcionam como o cérebro. A Figura 2 ilustra o mapa mental referente às atividades internas do *software SuperEvent:*

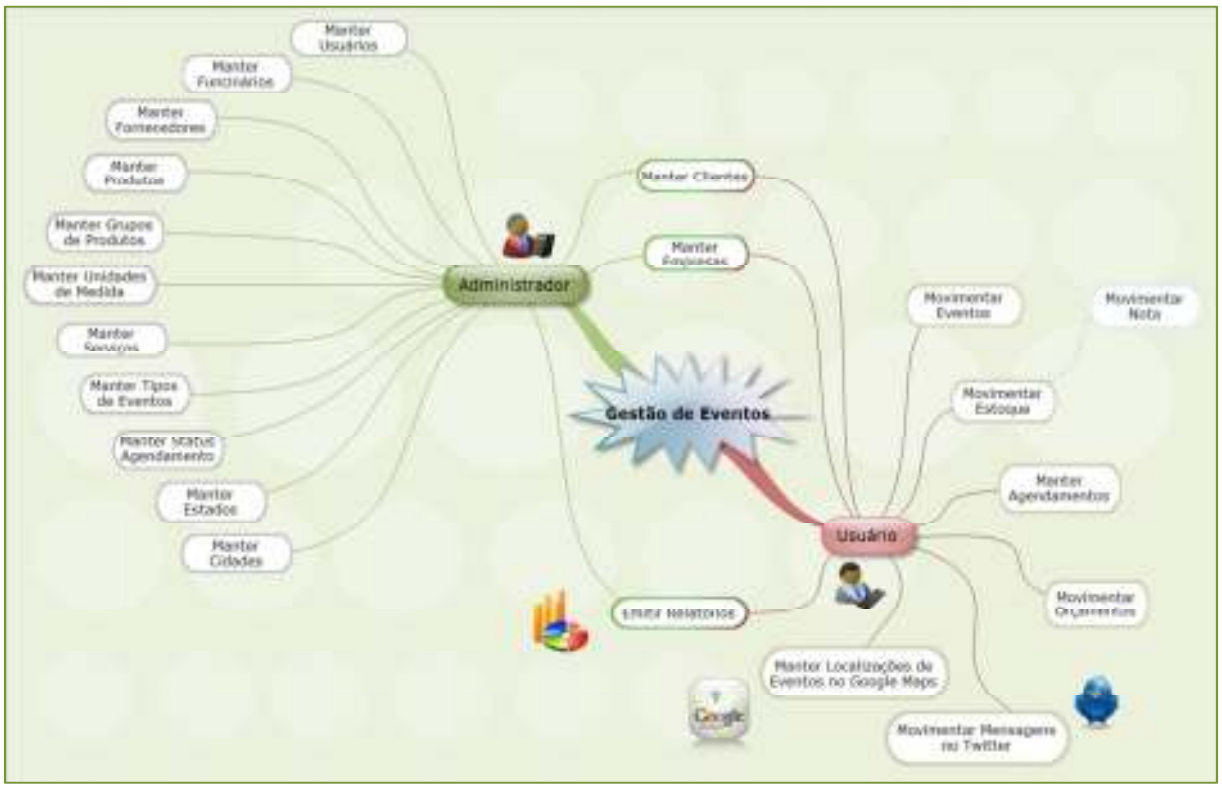

Figura 9 Mapa mental

3.2 LISTA DE EVENTOS

Para visualizar, especificar, construir e documentar artefatos de *software* utiliza-se a *UML*, linguagem padrão para a elaboração de estrutura de projetos de software. Esta é adequada para a modelagem de sistemas, cuja abrangência poderá incluir aplicações simples até sistemas complexos. É uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação de sistemas.

Para modelar o comportamento dos sistemas baseados em objetos, determinam-se quais eventos acontecem. Eventos fazem com que os sistemas tomem várias ações (BOOCH; JACOBSON; RUMBAUGH, 2000). A seguir são descritos os principais eventos:

Lista de Eventos

1. Cadastrar Usuários
2. Cadastrar Funcionários
3. Cadastrar Fornecedores
4. Cadastrar Produtos
5. Cadastrar Grupos de Produto
6. Cadastrar Unidades de Medida
7. Cadastrar Serviços
8. Cadastrar Tipos de Evento
9. Cadastrar Status de Agendamento
10. Cadastrar Estados
11. Cadastrar Cidades
12. Cadastrar Clientes
13. Cadastrar Empresas
14. Cadastrar Endereços
15. Cadastrar Telefones
16. Cadastrar Endereços Eletrônicos
17. Cadastrar Locais de Evento

18. Cadastrar Agendamentos
19. Movimentar Eventos
20. Movimentar Mensagens no Twitter
21. Movimentar Estoque
22. Movimentar Notas
23. Emitir Relatório de Clientes
24. Emitir Relatório de Fornecedores
25. Emitir Relatório de Produtos
26. Emitir Relatório de Estoque
27. Emitir Relatório de Eventos Orçados
28. Emitir Relatório de Eventos Agendados
29. Emitir Relatório de Eventos Realizados

3.3 CASOS DE USO

Como elemento primário da UML utiliza-se caso de uso. Caso de uso é um conjunto de cenários amarrados por um objetivo comum de um usuário, especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de sequencias de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor de um ator (BOOCH; JACOBSON; RUMBAUGH, 2005).

A Figura 10 apresenta o caso de uso geral comum entre os atores.

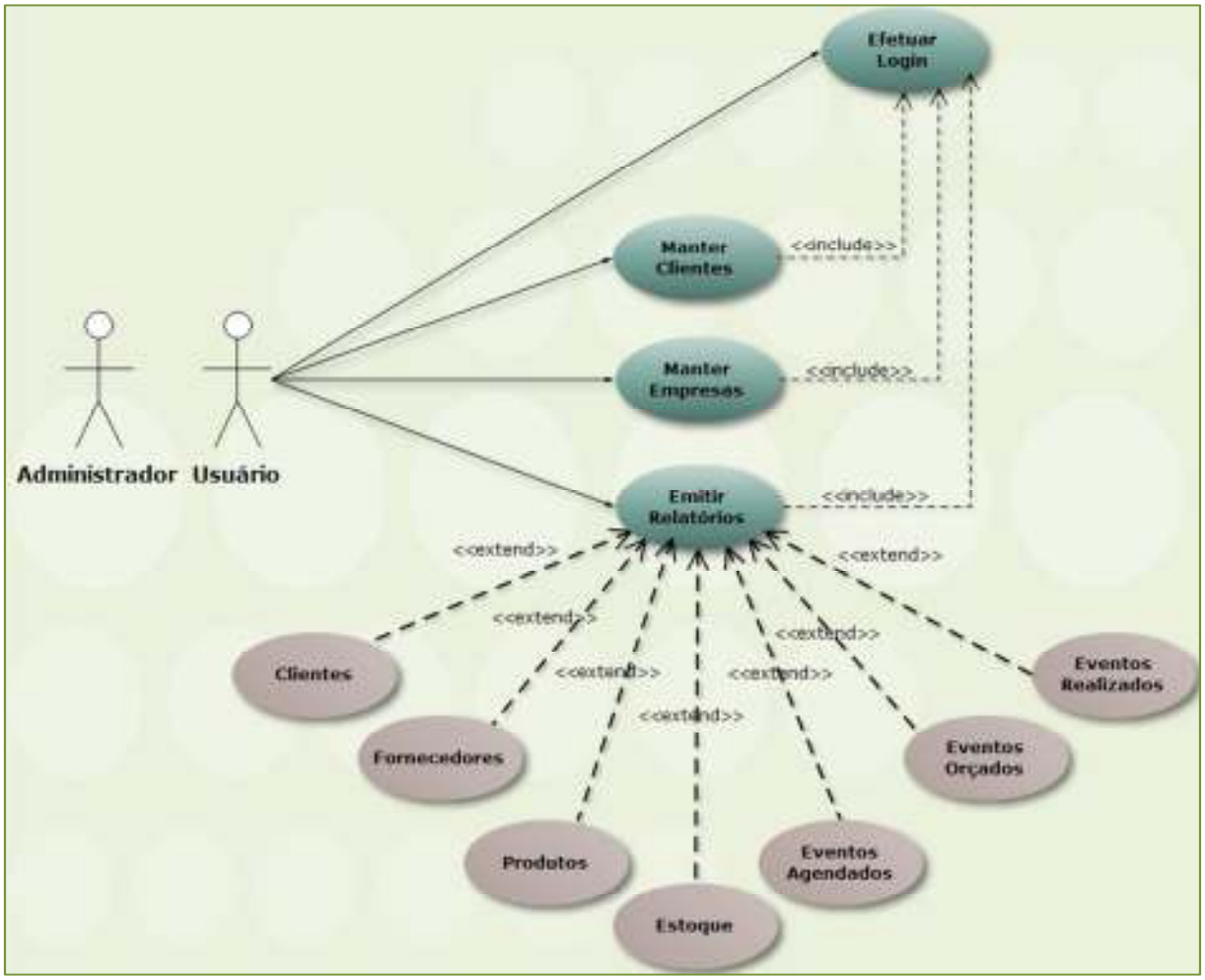

Figura 10 Caso de Uso Geral em comum – Administrador e Usuário

A Figura 11 apresenta o caso de uso geral do administrador.

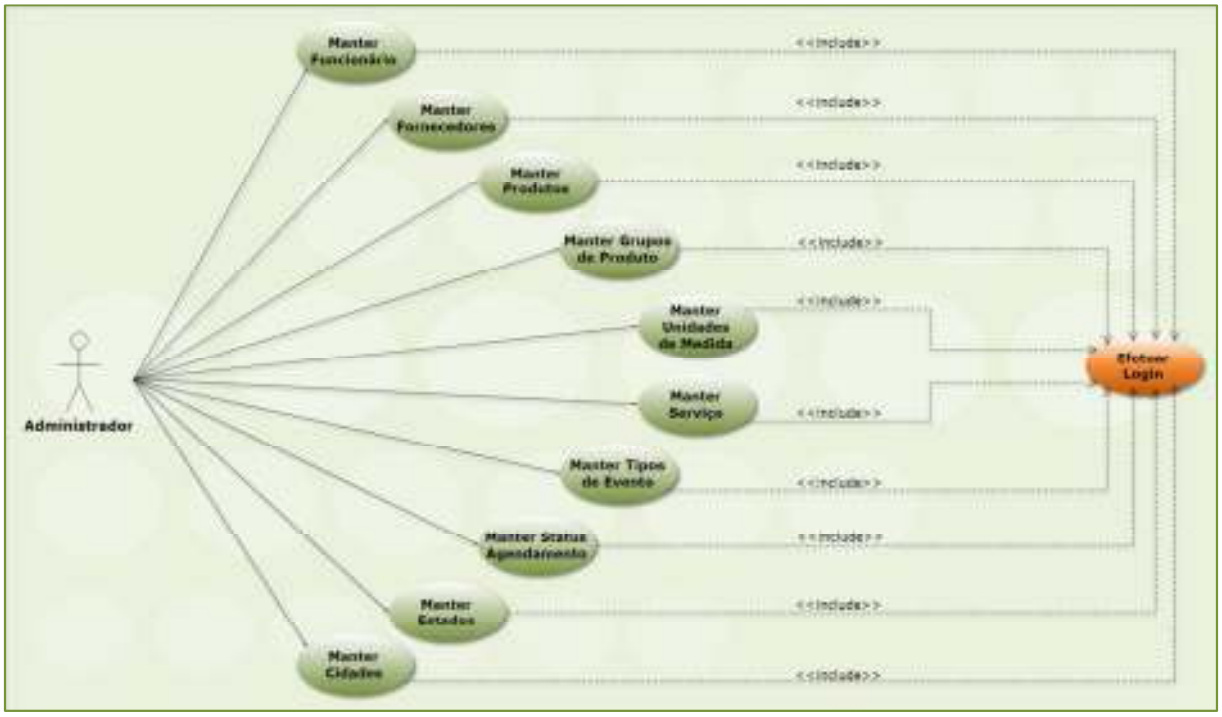

Figura 11 Caso de Uso Geral – Administrador

A Figura 12 apresenta o caso de uso geral do usuário.

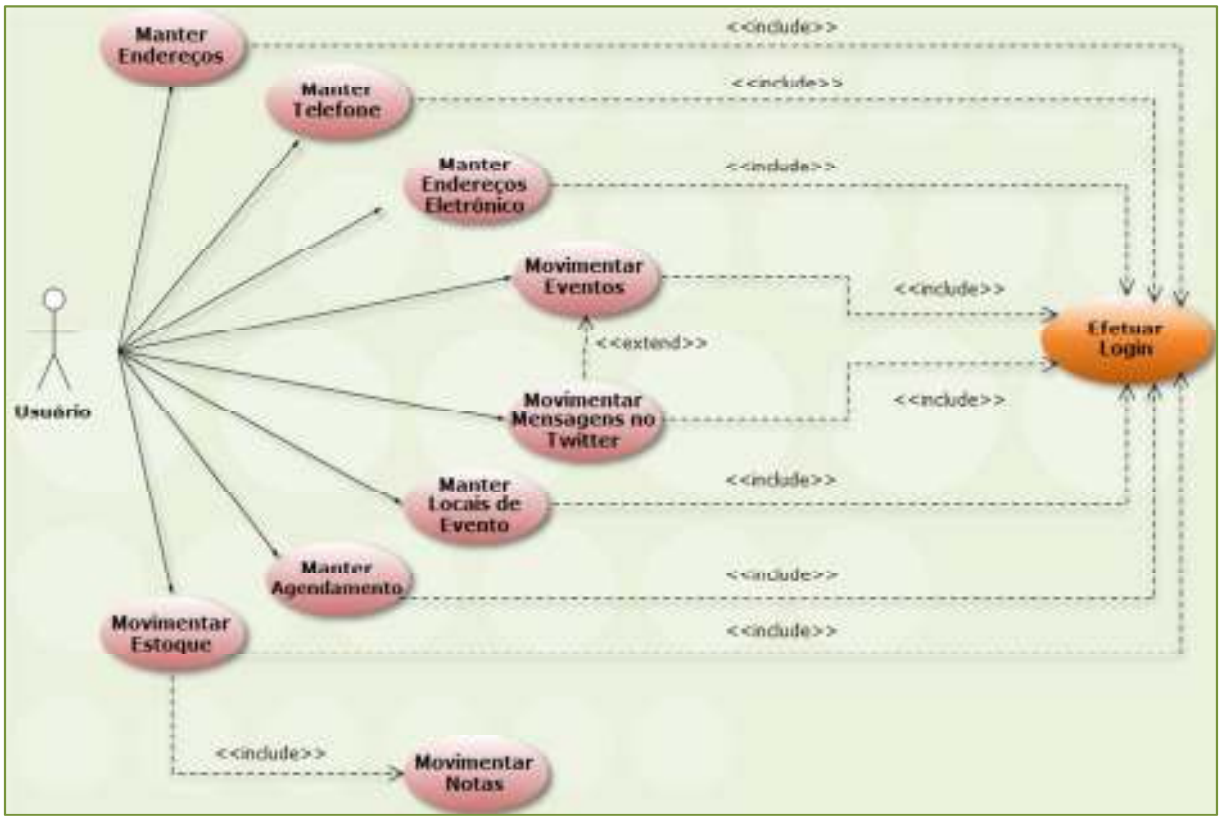

Figura 12 Caso de Uso Geral – Usuário

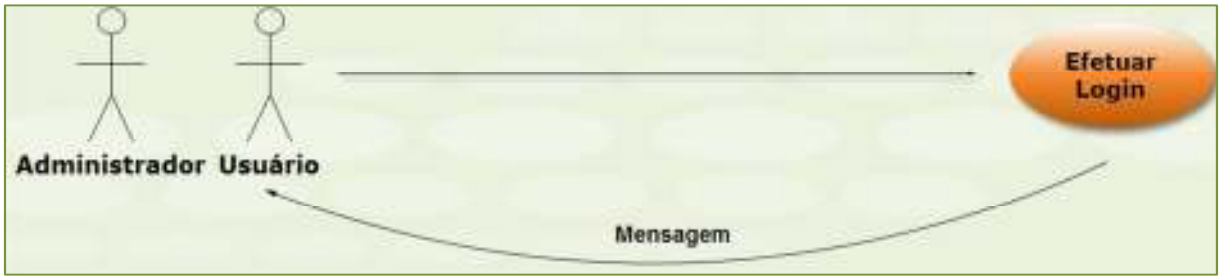

Figura 13 UC1 Diagrama de caso de uso Efetuar Controle de Acesso

Nome do caso de uso 1	Efetuar Controle de Acesso
Atores	Usuário
Pré-condição	Não existe
Cenário principal	1- O sistema solicita os dados para efetuar Controle de Acesso. 2- O usuário informa os dados. 3- O usuário confirma Controle de Acesso. 4- O sistema recupera os dados informados pelo usuário. 5- O sistema valida os dados. 6- O usuário conecta no sistema.
Cenário alternativo	Não existe.
Casos de teste	4.1- Caso os dados estejam corretos executa operação solicitada. 4.2- Caso os dados estejam incorretos cancela operação e exibe mensagem de alerta.

Tabela 1 Efetuar Controle de Acesso

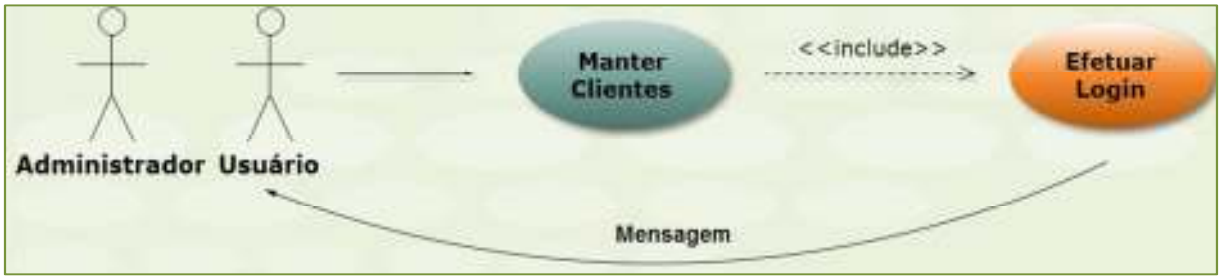

Figura 14 UC2 Diagrama de caso de uso Manter Clientes

Nome do caso de uso 2	Manter clientes
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador ou usuário informa os dados do cliente. 2- O administrador informa um endereço (Caso de uso manter endereço) 3- O administrador informa zero ou mais telefone (Caso de uso manter telefone) 4- O administrador informa zero ou mais endereços eletrônicos (Caso de uso manter endereços eletrônicos). 5- O sistema valida os dados informados. 6- O administrador ou usuário seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra o usuário.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O administrador ou usuário poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 O sistema verifica se o administrador informou um endereço.

Tabela 2 Manter Clientes

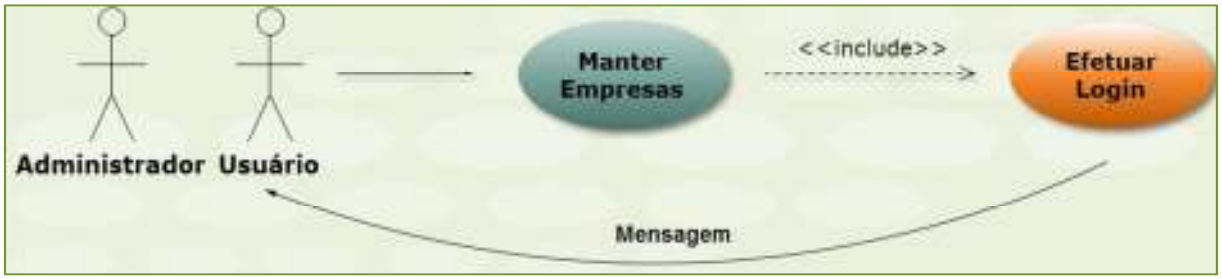

Figura 15 UC3 Diagrama de caso de uso Manter Empresas

Nome do caso de uso 3	Manter empresas
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador ou usuário informa os dados da empresa. 2- O administrador informa um endereço (Caso de uso manter endereço) 3- O administrador informa zero ou mais telefone (Caso de uso manter telefone) 4- O administrador informa zero ou mais endereços eletrônicos (Caso de uso manter endereços eletrônicos). 5- O sistema valida os dados informados. 6- O administrador ou usuário seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra a empresa.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador ou usuário, exibirá mensagem de alerta. O administrador ou usuário poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 O sistema verifica se o administrador informou um endereço.

Tabela 3 Manter Empresas

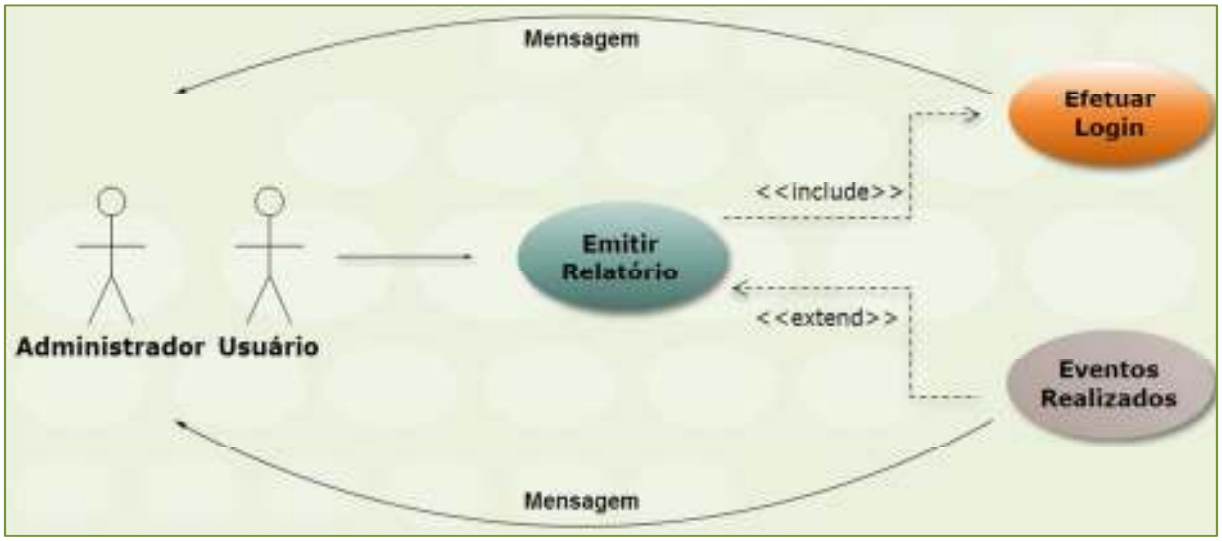

Figura 16 UC4 Diagrama de caso de uso Emitir Relatório de Eventos Realizados

Nome do caso de uso 4	Emitir relatório de eventos realizados
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 4 Emitir Relatório de Eventos Realizados

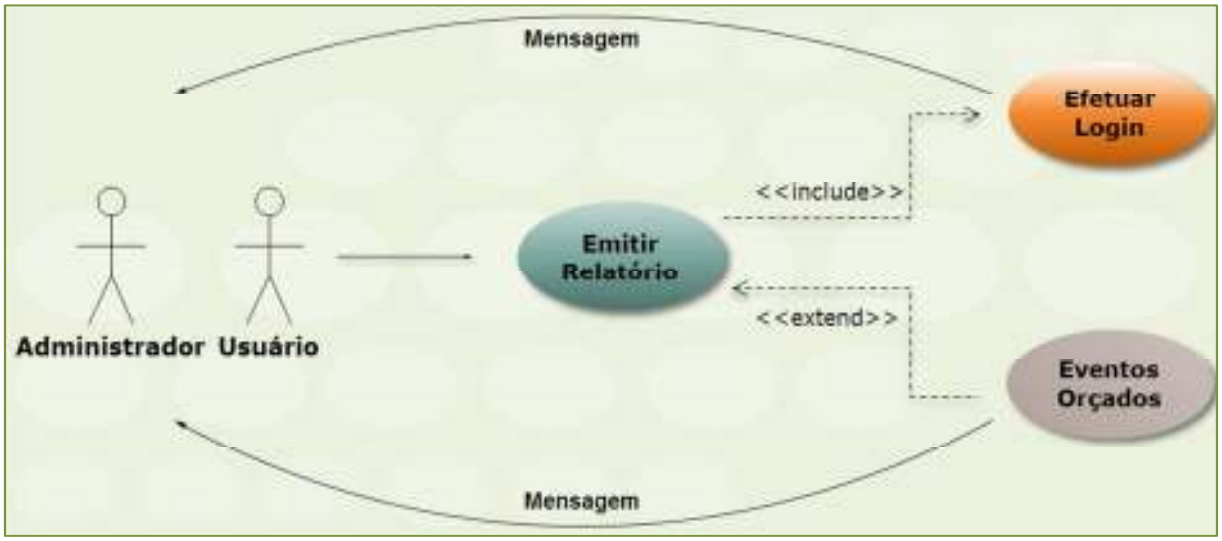

Figura 17 UC5 Diagrama de caso de uso Emitir Relatório de Eventos Orçados

Nome do caso de uso 5	Emitir relatório de eventos orçados
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 5 Emitir Relatório de Eventos Orçados

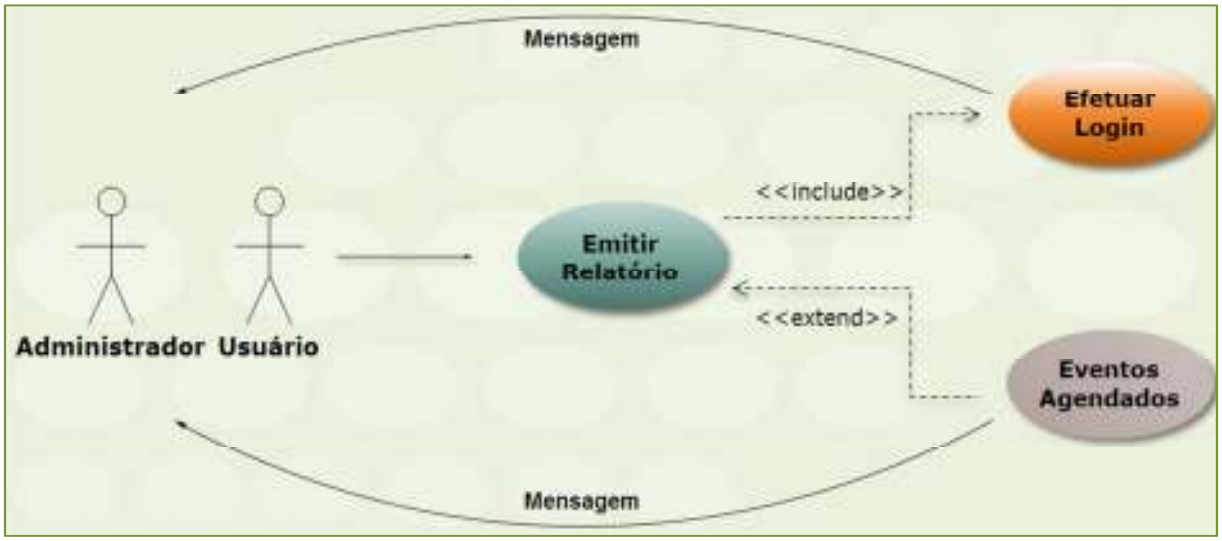

Figura 18 UC6 Diagrama de caso de uso Emitir Relatório de Eventos Agendados

Nome do caso de uso 6	Emitir relatório de eventos agendados
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 6 Emitir Relatório de Eventos Agendados

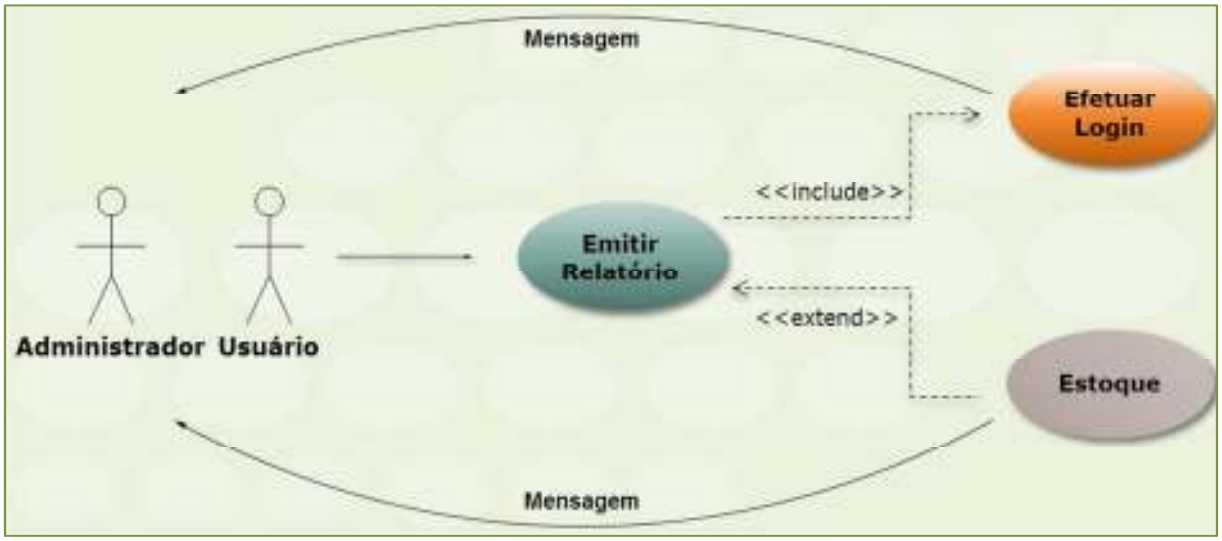

Figura 19 UC7 Diagrama de caso de uso Emitir Relatório de Estoque

Nome do caso de uso 7	Emitir relatório do estoque
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 7 Emitir Relatório de Estoque

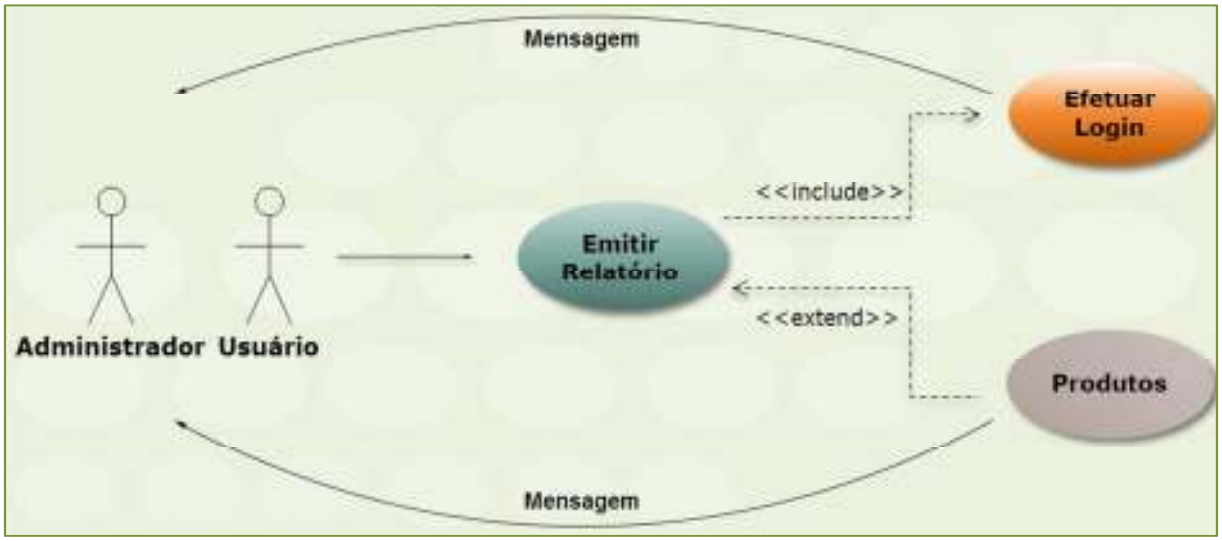

Figura 20 UC8 Diagrama de caso de uso Emitir Relatório de Produtos

Nome do caso de uso 8	Emitir relatório de produtos
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 8 Emitir Relatório de Produtos

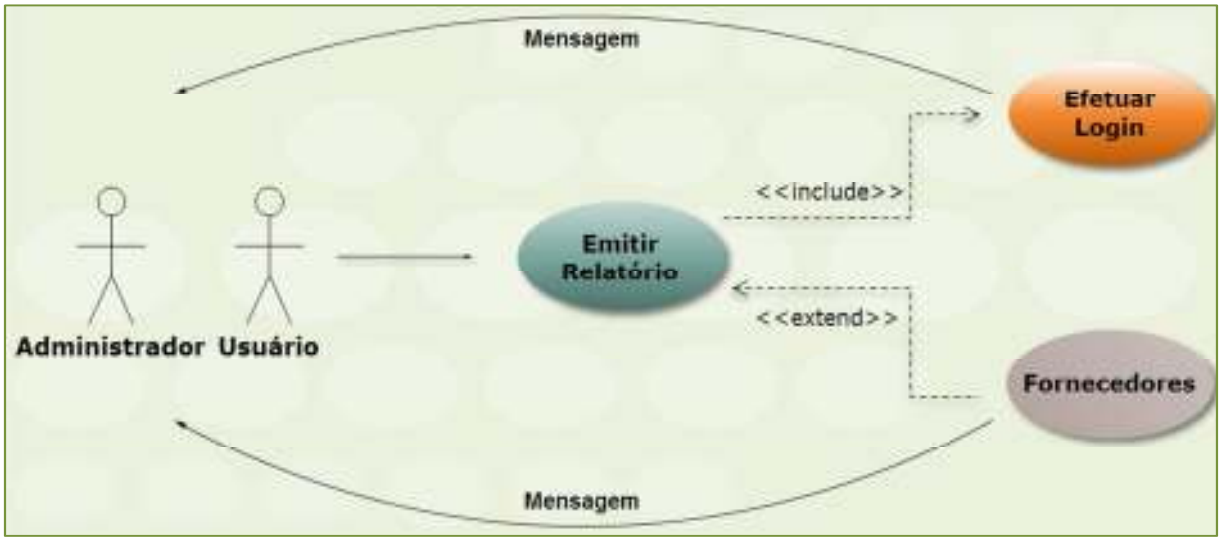

Figura 21 UC9 Diagrama de caso de uso Emitir Relatório de Fornecedores

Nome do caso de uso 9	Emitir relatório de fornecedores
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 9 Emitir Relatório de Fornecedores

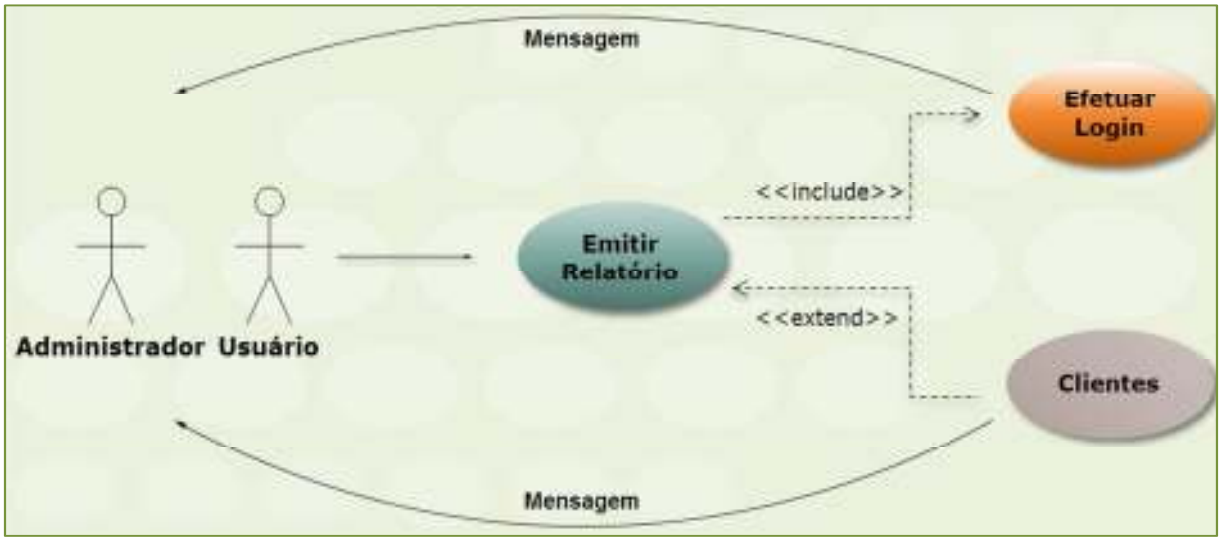

Figura 22 UC10 Diagrama de caso de uso Emitir Relatório de Clientes

Nome do caso de uso 10	Emitir relatório de clientes
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório 2- O administrador e/ou usuário seleciona o botão visualizar relatório 3- O administrador e/ou usuário seleciona o botão imprimir 4- O sistema imprime o relatório com sucesso
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não imprimir
Casos de teste	4.1 O administrador e/ou usuário cancela a operação

Tabela 10 Emitir Relatório de Clientes

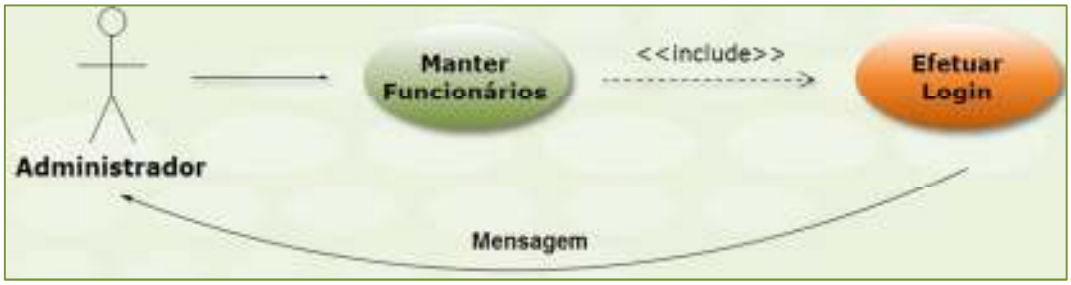

Figura 23 UC11 Diagrama de caso de uso Manter Funcionários

Nome do caso de uso 11	Manter funcionário
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do funcionário**. 2- O administrador informa um endereço (Caso de uso manter endereço). 3- O administrador informa zero ou mais telefone (Caso de uso manter telefone). 4- O administrador informa zero ou mais endereços eletrônicos (Caso de uso manter endereços eletrônicos). 5- O sistema valida os dados informados. 6- O administrador seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra **o funcionário**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 O sistema verifica se o administrador informou um endereço.

Tabela 11 Manter Funcionários

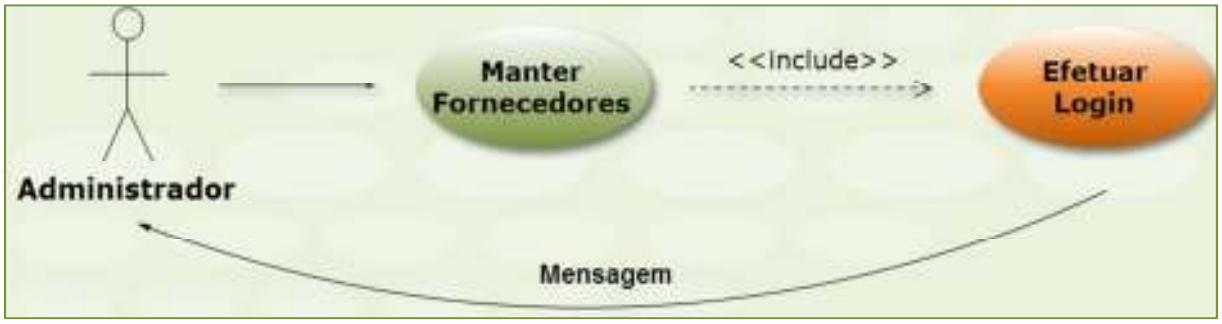

Figura 24 UC12 Diagrama de caso de uso Manter Fornecedores

Nome do caso de uso 12	Manter fornecedores
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do fornecedor**. 2- O administrador informa um endereço (Caso de uso manter endereço). 3- O administrador informa zero ou mais telefone (Caso de uso manter telefone). 4- O administrador informa zero ou mais endereços eletrônicos (Caso de uso manter endereços eletrônicos). 5- O sistema valida os dados informados. 6- O administrador seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra **o fornecedor**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 O sistema verifica se o administrador informou um endereço.

Tabela 12 Manter Fornecedores

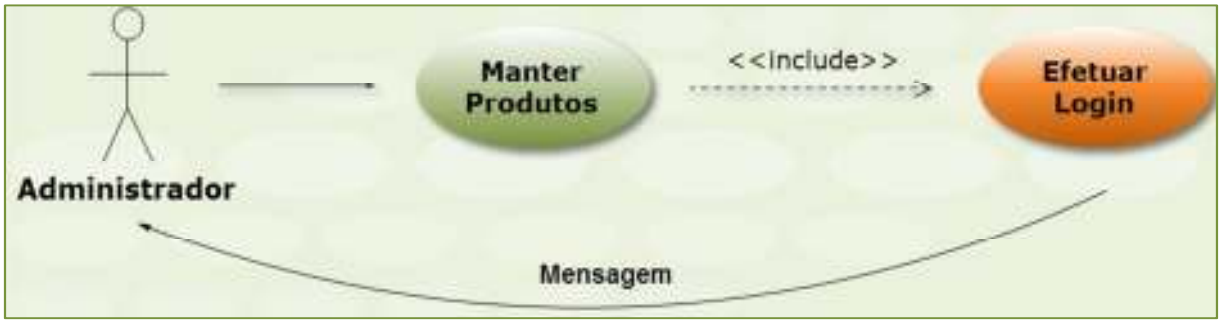

Figura 25 UC13 Diagrama de caso de uso Manter Produtos

Nome do caso de uso 13	Manter produtos
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do produto**. 2- O administrador seleciona uma unidade de medida. 3- O administrador seleciona um grupo de produto. 4- O administrador seleciona zero ou mais fornecedores. 5- O sistema valida os dados informados. 6- O administrador seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra **o produto**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 O sistema verifica se o administrador selecionou uma unidade de medida. 5.2 O sistema verifica se o administrador selecionou um grupo de produto. 5.3 O sistema não confirma o cadastro e emite uma mensagem de erro. 5.4 O sistema cancela a operação.

Tabela 13 Manter Produtos

Figura 26 UC14 Diagrama de caso de uso Manter Grupos de Produto

Nome do caso de uso 14	Manter grupos de produto
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do grupo do produto**. 2- O sistema valida os dados informados. 3- O administrador seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o grupo do produto**. .
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 14 Manter Grupos de Produto

Figura 27 UC15 Diagrama de caso de uso Manter Unidades de Medida

Nome do caso de uso 15	Manter unidades de medida
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **da unidade de medida**. 2- O sistema valida os dados informados. 3- O administrador seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **a unidade de medida**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 15 Manter Unidades de Medida

Figura 28 UC16 Diagrama de caso de uso Manter Serviços

Nome do caso de uso 16	Manter serviços
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do serviço**. 2- O sistema valida os dados informados. 3- O administrador seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o serviço**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 16 Manter Serviços

Figura 29 UC17 Diagrama de caso de uso Manter Tipos de Evento

Nome do caso de uso 17	Manter tipos de evento
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do tipo do evento.** 2- O sistema valida os dados informados. 3- O administrador seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o tipo de evento**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 17 Manter Tipos de Evento

Figura 30 UC18 Diagrama de caso de uso Manter Status Agendamentos

Nome do caso de uso 18	Manter status do agendamento
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados **do status do agendamento**. 2- O sistema valida os dados informados. 3- O administrador seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o status do agendamento**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro 1.3 O sistema cancela a operação.

Tabela 18 Manter Status do Agendamento

Figura 31 UC19 Diagrama de caso de uso Manter Estados

Nome do caso de uso 19	Manter estados
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados do estado. 2- O sistema valida os dados informados. 3- O administrador seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra o estado.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 19 Manter Estados

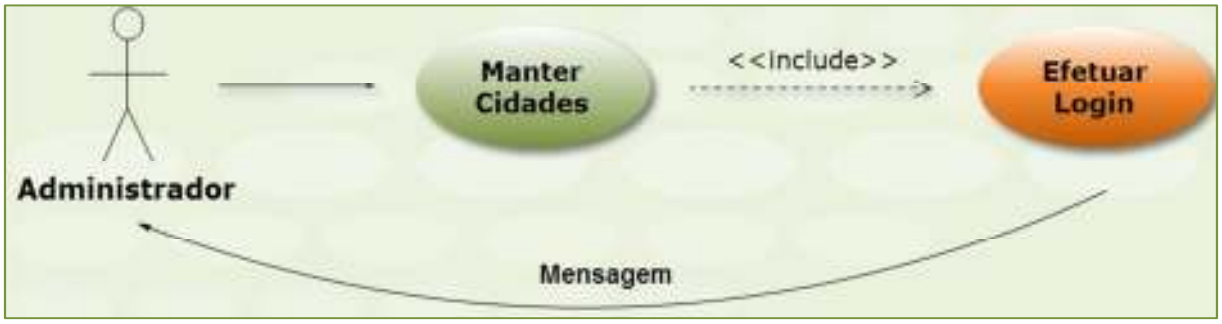

Figura 32 UC20 Diagrama de caso de uso Manter Cidades

Nome do caso de uso 20	Manter cidades
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados da cidade. 2- O administrador seleciona um estado. 3- O sistema valida os dados informados. 4- O administrador seleciona a opção “cadastrar”. 5- O sistema emite mensagem de sucesso. 6- O sistema cadastra a cidade.
Cenário alternativo	Caso o sistema não valide os dados informados pelo administrador, exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 3.1 O sistema verifica se o administrador selecionou um estado.

Tabela 20 Manter Cidades

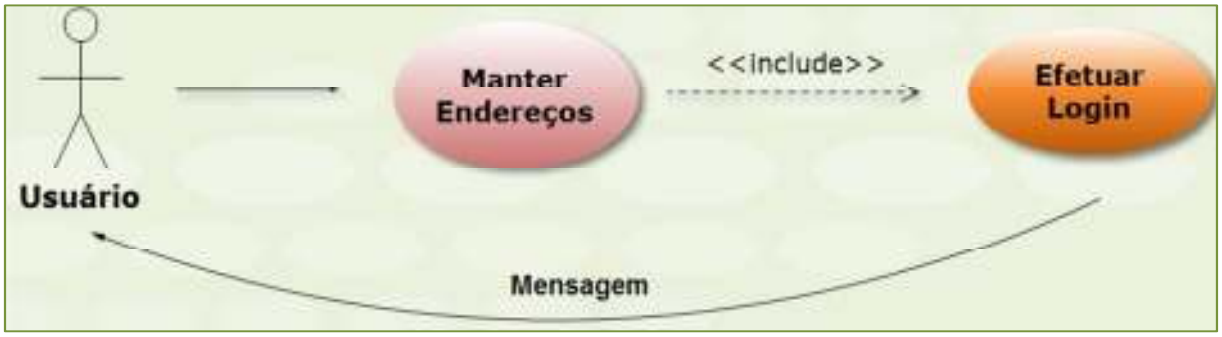

Figura 33 UC21 Diagrama de caso de uso Manter Endereços

Nome do caso de uso 21	Manter endereços
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados **do endereço** 2- O sistema valida os dados informados. 3- O usuário seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o endereço**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 21 Manter Endereços

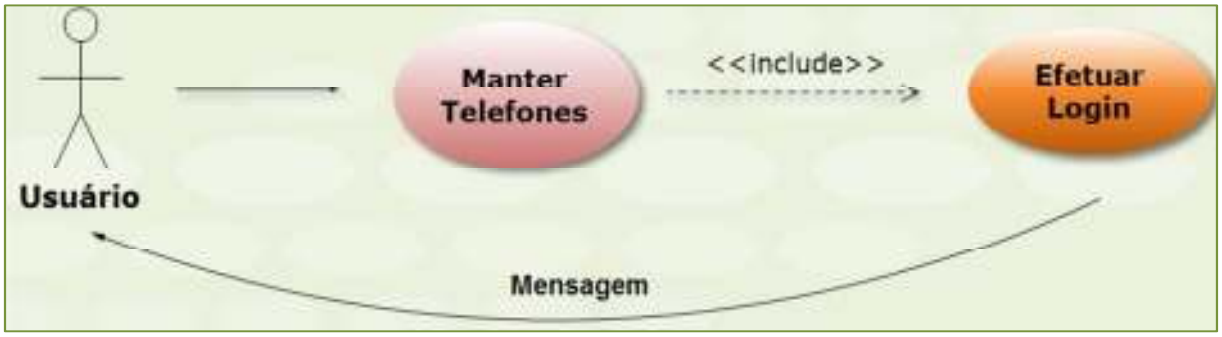

Figura 34 UC22 Diagrama de caso de uso Manter Telefones

Nome do caso de uso 22	Manter telefones
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados do telefone. 2- O sistema valida os dados informados. 3- O usuário seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o telefone**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 22 Manter Telefones

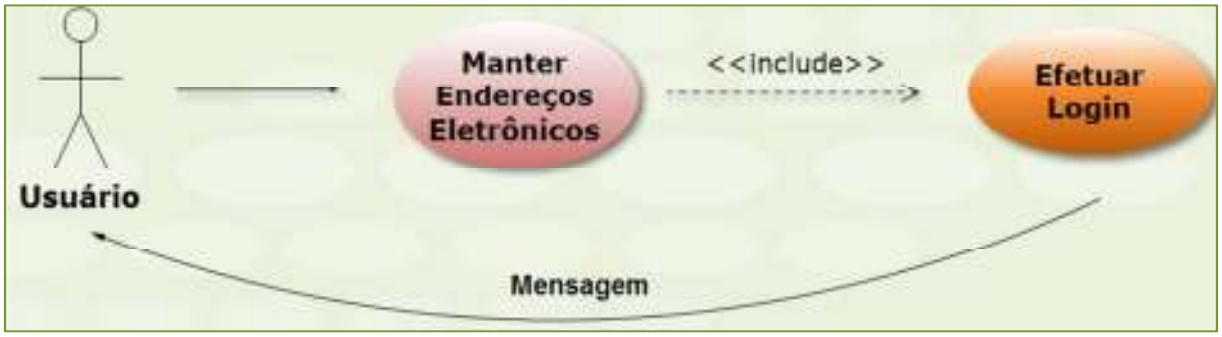

Figura 35 UC23 Diagrama de caso de uso Manter Endereços Eletrônicos

Nome do caso de uso 23	Manter endereços eletrônicos
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados do endereço eletrônico 2- O sistema valida os dados informados. 3- O usuário seleciona a opção “cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra **o endereço eletrônico**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 23 Manter Endereços Eletrônicos

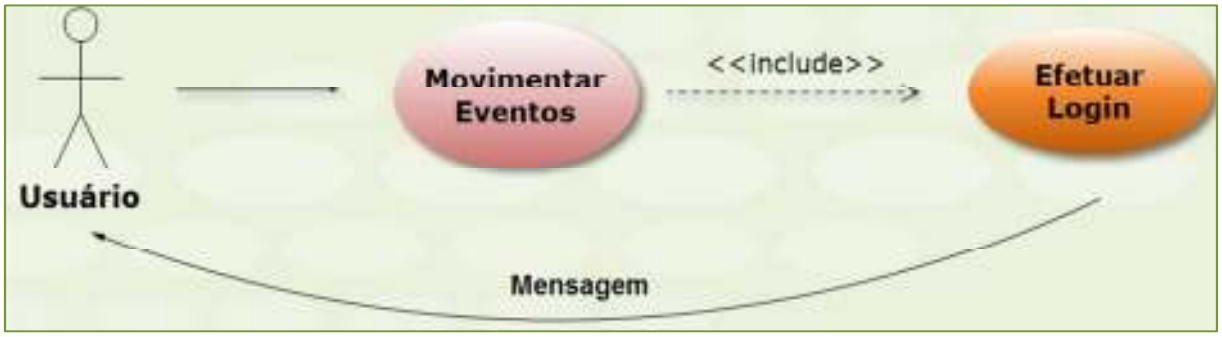

Figura 36 UC24 Diagrama de caso de uso Movimentar Eventos

Nome do caso de uso 24	Movimentar evento
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados **do evento.** 2- O usuário informa um ou mais clientes. 3- O usuário seleciona um tipo de evento. 4- O usuário informa zero ou mais serviços para serem executados por funcionários. 5- O usuário informa um local para o evento (Caso de uso movimentar local de evento). 6- O sistema valida os dados informados. 7- O usuário seleciona a opção “cadastrar”. 8- O sistema emite mensagem de sucesso. 9- O sistema cadastra **o evento**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante a movimentação.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 7.1 O sistema verifica se o usuário informou um ou mais clientes. 7.2 O sistema verifica se o usuário informou um tipo de evento 7.2 O sistema verifica se o usuário informou um local para o evento.

Tabela 24 Movimentar Evento

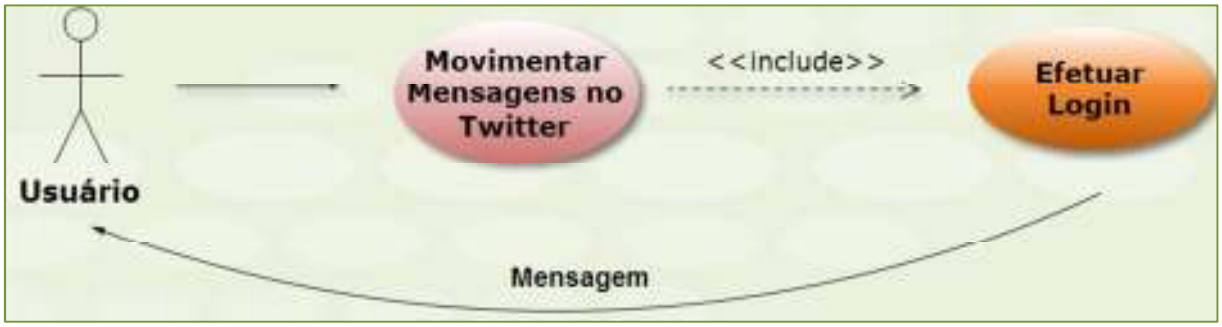

Figura 37 UC25 Diagrama de caso de uso Movimentar Mensagens no Twitter

Nome do caso de uso 25	Movimentar mensagens no *twitter*
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O sistema recupera informações para autenticação no *twitter*. 2- O sistema aguarda validação de dados. 3- O usuário informa mensagem para ser enviada ao *twitter*. 4- O usuário seleciona a opção “enviar”. 5- O sistema exibe mensagem de confirmação. 6- O sistema publica mensagem no *twitter*. 7- O sistema emite mensagem de sucesso.
Cenário alternativo	Caso o sistema não valide os dados de autenticação do usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante a movimentação.
Casos de teste	2.1 Caso as informações de autenticação estejam incorretas, o sistema exibe mensagem. 2.2 Caso as informações de autenticação estejam corretas, o sistema exibe tela de envio de mensagem.

Tabela 25 Movimentar Mensagens no Twitter

Figura 38 UC26 Diagrama de caso de uso Movimentar Locais de Evento

Nome do caso de uso 26	Movimentar local de evento
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados **do local de evento.** 2- O usuário informa um endereço. 3- O usuário informa zero ou mais telefones. 4- O usuário informa zero ou mais endereços eletrônicos. 5- O sistema valida os dados informados. 6- O usuário seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra **o local de evento**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante a movimentação.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 O sistema verifica se o usuário informou um endereço

Tabela 26 Movimentar Locais de Evento

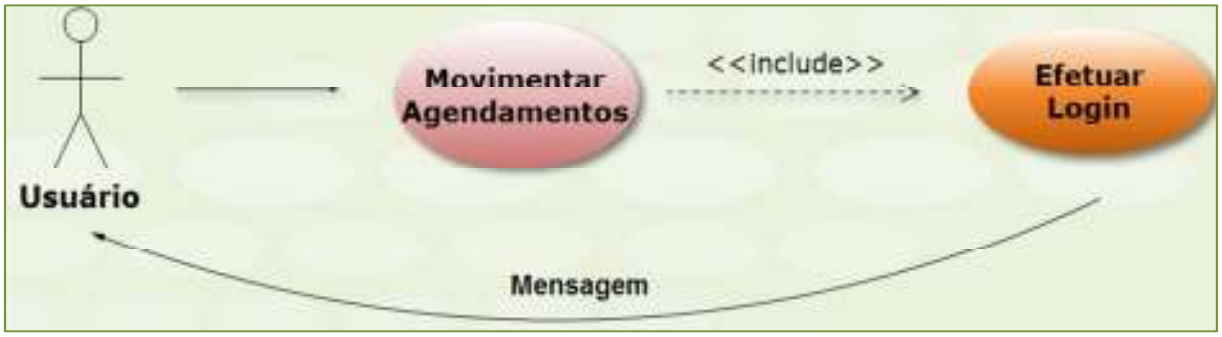

Figura 39 UC27 Diagrama de caso de uso Movimentar Agendamentos

Nome do caso de uso 27	Movimentar agendamentos
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados **do agendamento.** 2- O usuário seleciona um funcionário. 3- O usuário seleciona um evento. 4- O usuário seleciona um status do agendamento. 5- O sistema valida os dados informados. 6- O usuário seleciona a opção “cadastrar”. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra **o agendamento**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante a movimentação.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 5.1 Verificar se já existe um evento na mesma data e no mesmo local. 5.2 O sistema verifica se há conflito na agenda dos funcionários. 5.2 O sistema verifica se o administrador selecionou um funcionário. 5.3 O sistema verifica se o administrador selecionou um evento. 5.4 O sistema verifica se o administrador selecionou um status do agendamento.

Tabela 27 Movimentar Agendamentos

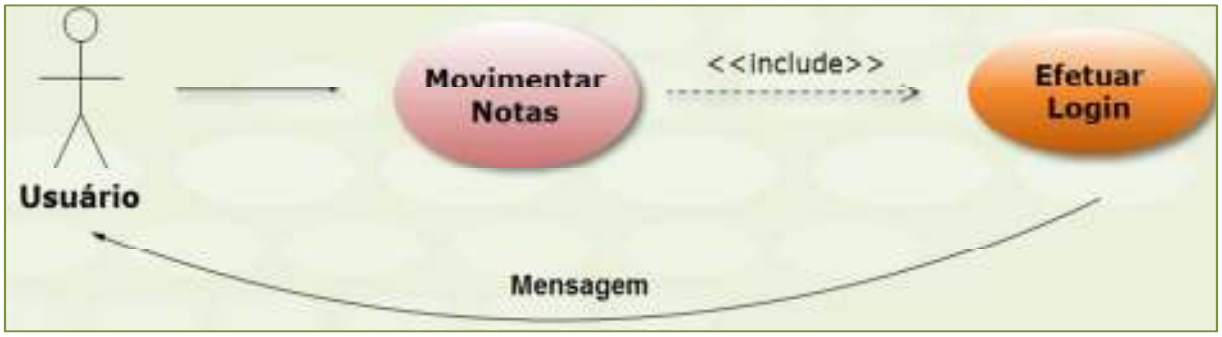

Figura 40 UC28 Diagrama de caso de uso Movimentar Notas

Nome do caso de uso 28	Movimentar notas
Atores	Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados **da nota**. 2- O usuário insere um ou mais itens de nota. 3- O sistema valida os dados informados. 4- O usuário seleciona a opção “cadastrar”. 5- O sistema verifica o tipo de movimentação 6- O sistema atualiza a quantidade do produto. 7- O sistema emite mensagem de sucesso. 8- O sistema cadastra **a nota**.
Cenário alternativo	Caso o sistema não valide os dados informados pelo usuário, exibirá mensagem de alerta. O usuário poderá cancelar o processo durante o cadastro.
Casos de teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação. 4.1 O sistema verifica se o usuário inseriu um ou mais itens de nota. 5.1 Caso o tipo de movimentação seja de saída o sistema verifica a quantidade do produto esta disponível no estoque. 5.1.1 Caso a quantidade do produto não esteja disponível no estoque o sistema exibe uma mensagem de alerta. 5.1.2 Caso a quantidade do produto esteja disponível no estoque o sistema subtrai da quantidade do produto. 5.2 Caso o tipo de movimentação seja de entrada o sistema adiciona a quantidade do produto no estoque.

Tabela 28 Movimentar Notas

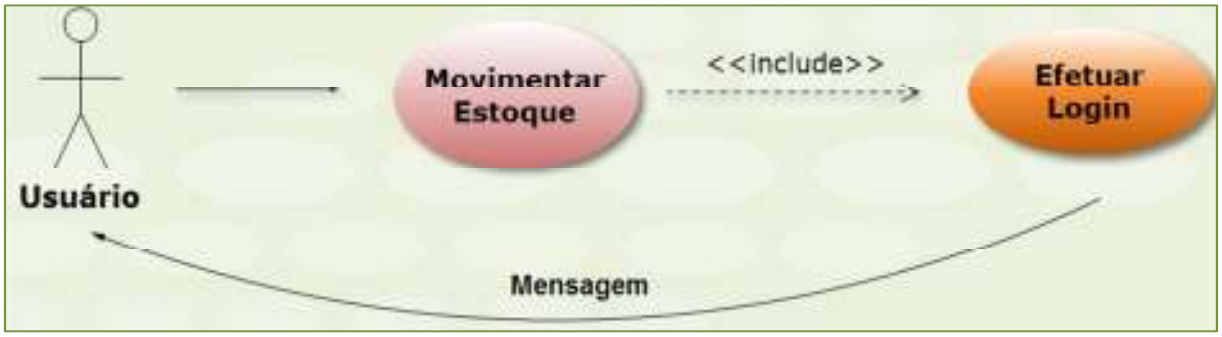

Figura 41 UC29 Diagrama de caso de uso Movimentar Estoque

Nome do caso de uso 29	Movimentar estoque
Atores	Usuário
Pré-condição	Efetuar login
Cenário principal	1- O usuário cadastra uma nova nota. 2- Executar caso de uso “Movimentar Notas” 3- O usuário confirma nota.
Cenário alternativo	Não existe
Casos de teste	Não Existe

Tabela 29 Movimentar Estoque

3.4 DIAGRAMAS DE ATIVIDADES

Os diagramas de atividades são um dos principais diagramas disponíveis na UML para a modelagem de aspectos dinâmicos de sistema. Ele é essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra (BOOCH; JACOBSON; RUMBAUGH, 2000).

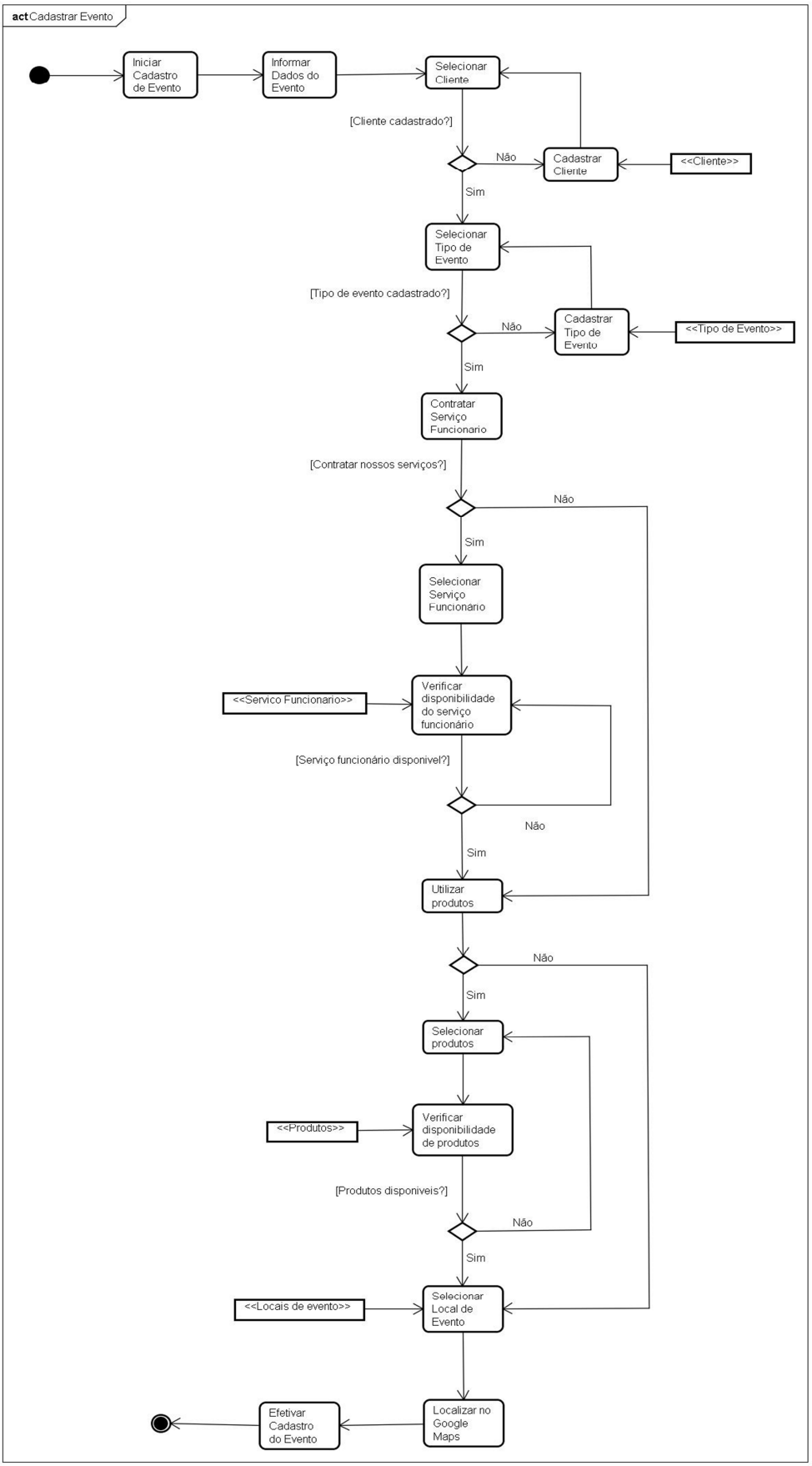

Figura 42 Diagrama de Atividade - Cadastrar Evento

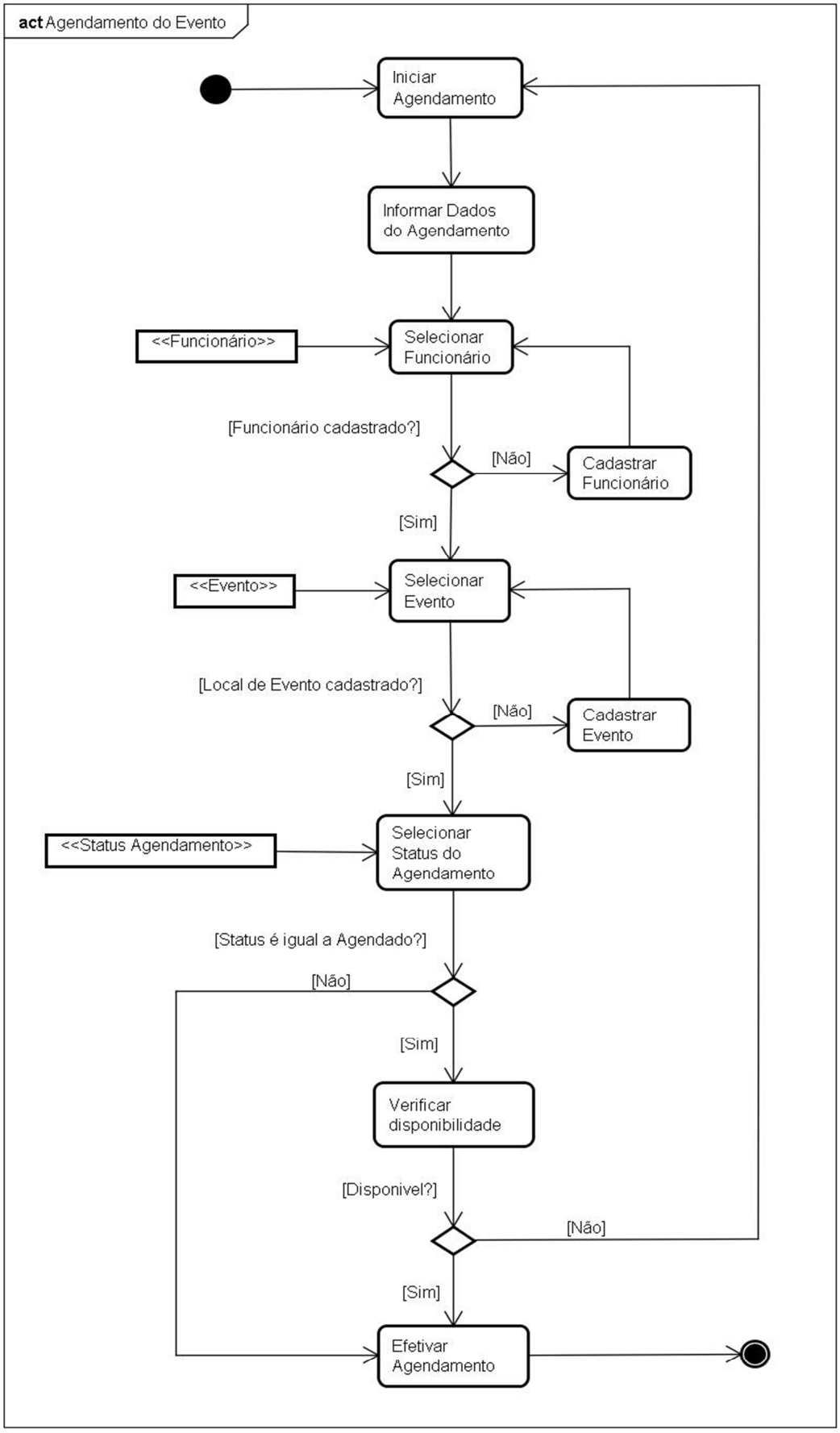

Figura 43 Diagrama de Atividade - Agendamento

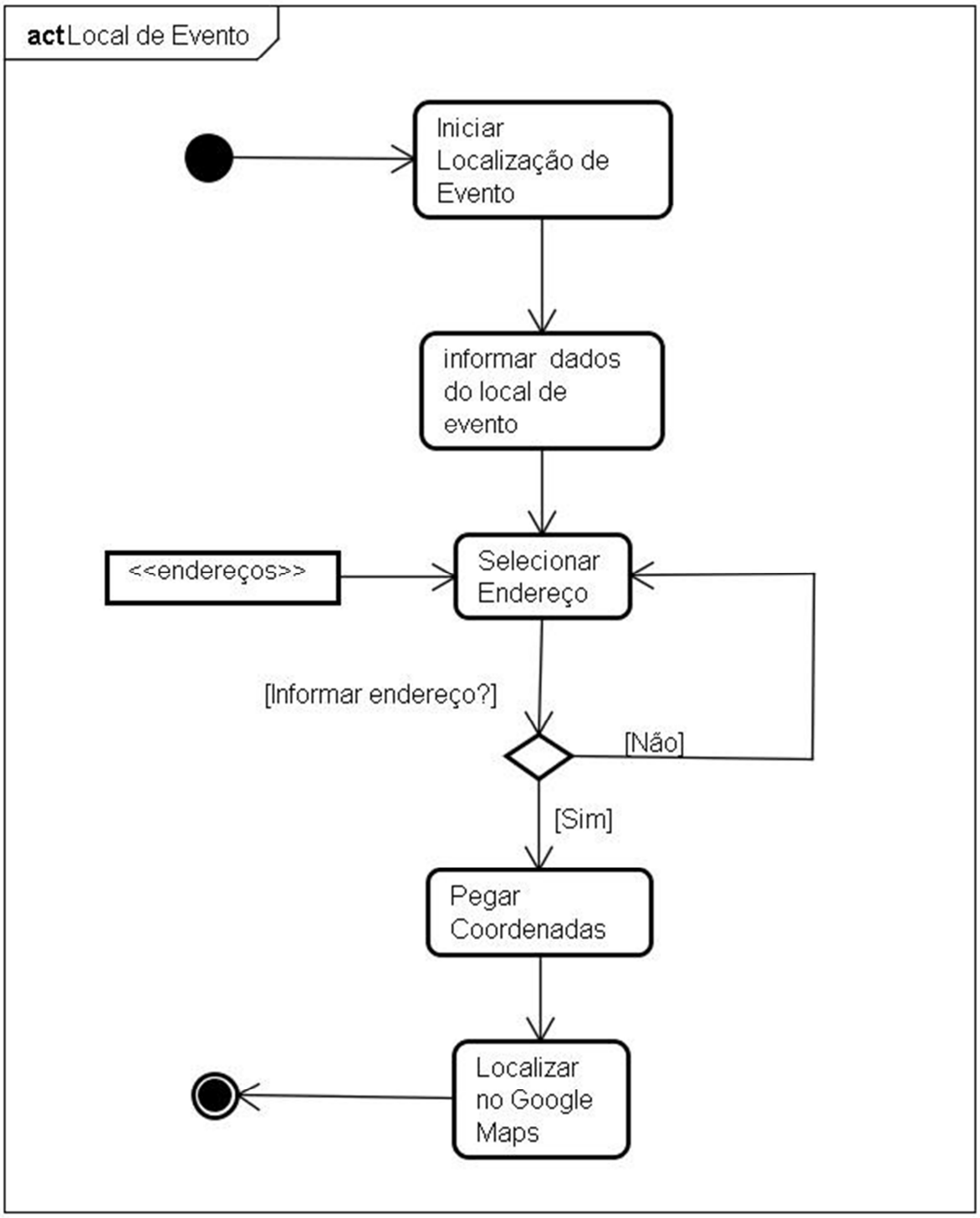

Figura 44 Diagrama de Atividade - Localizar Evento

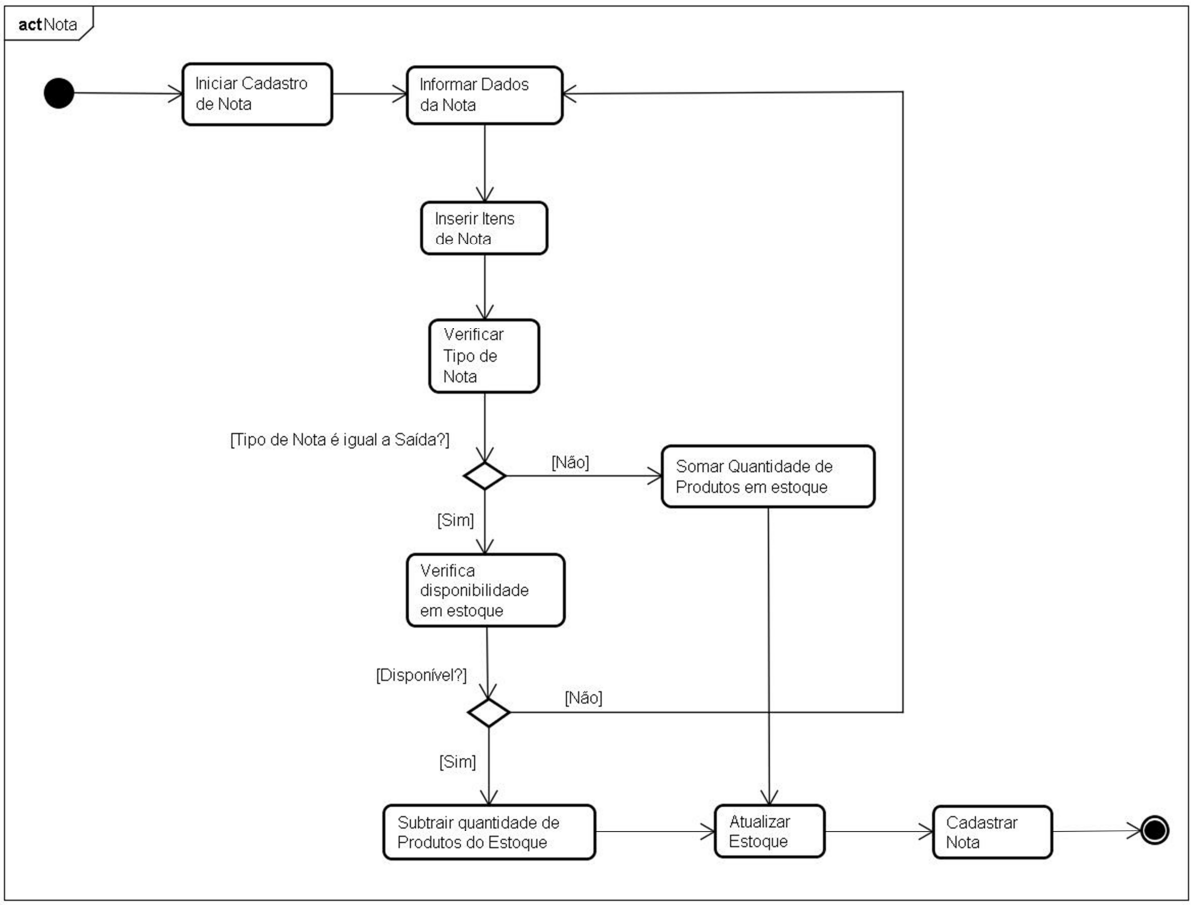

Figura 45 Diagrama de Atividade – Nota

3.5 DIAGRAMAS DE SEQUENCIAS

Um diagrama de sequência mostra o fluxo de controle por ordenamento de tempo e enfatiza a passagem de mensagens à medida que elas vão se desenrolando durante o tempo. Ele revela a sequência de mensagens que implementam um serviço ou transação (LEE, TEPFENHART 2001).

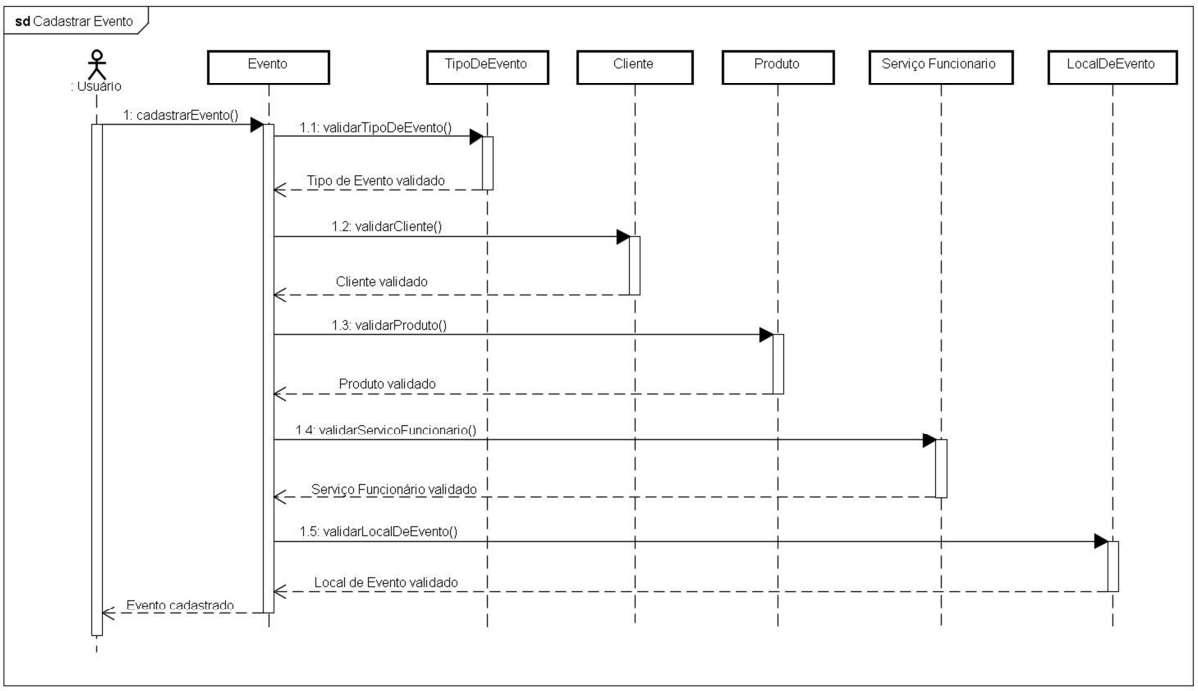

Figura 46 Diagrama de Sequencia - Cadastrar Evento

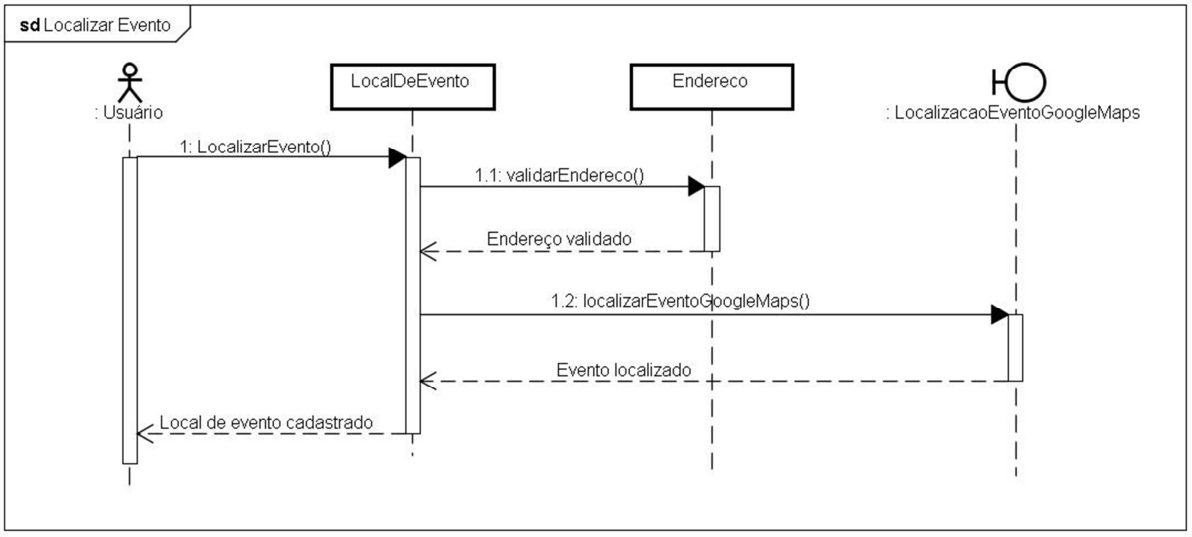

Figura 47 Diagrama de Sequencia - Localizar Evento

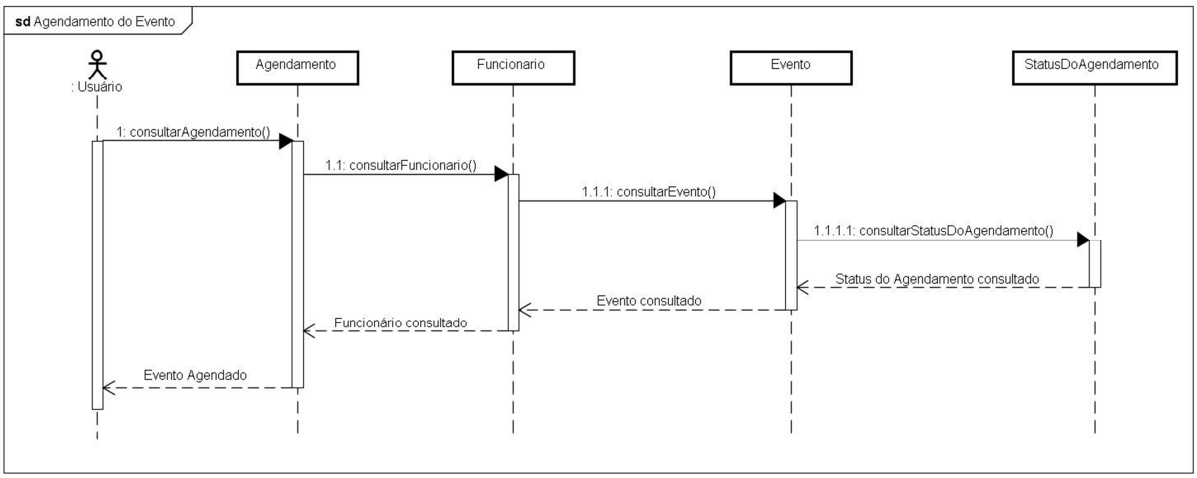

Figura 48 Diagrama de Sequencia - Agendar Evento

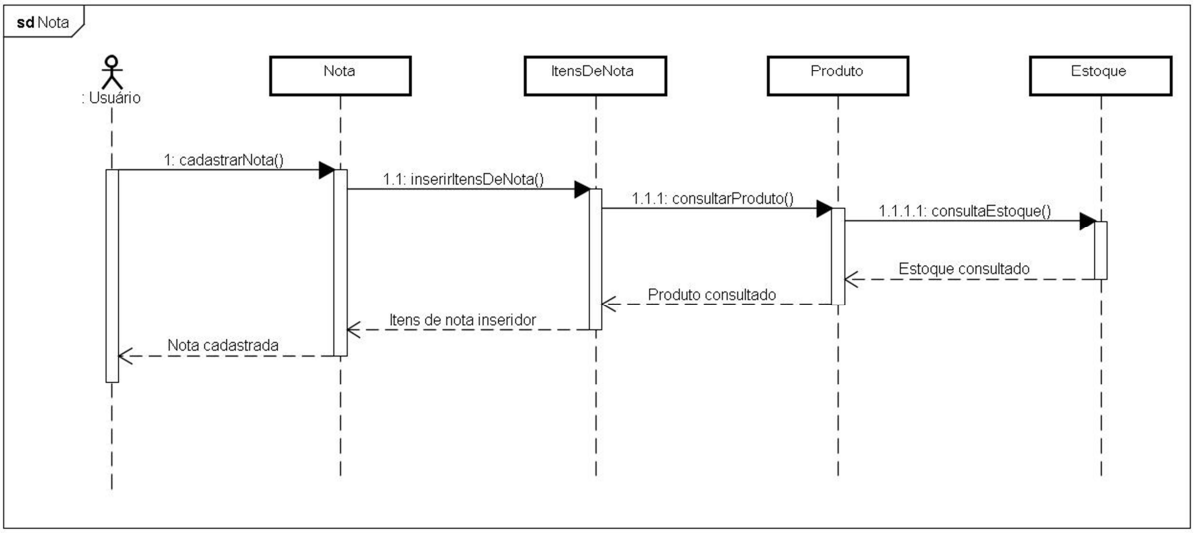

Figura 49 Diagrama de Sequencia - Nota

3.6 DIAGRAMA DE CLASSE

O diagrama de classe apresenta uma visão estática do projeto de um sistema. Eles são importantes não só para a visualização, a especificação e a documentação de modelos estruturais, mas também para a construção de sistema executáveis por intermédio de engenharia de produção e reversa (BOOCH; JACOBSON; RUMBAUGH, 2000).

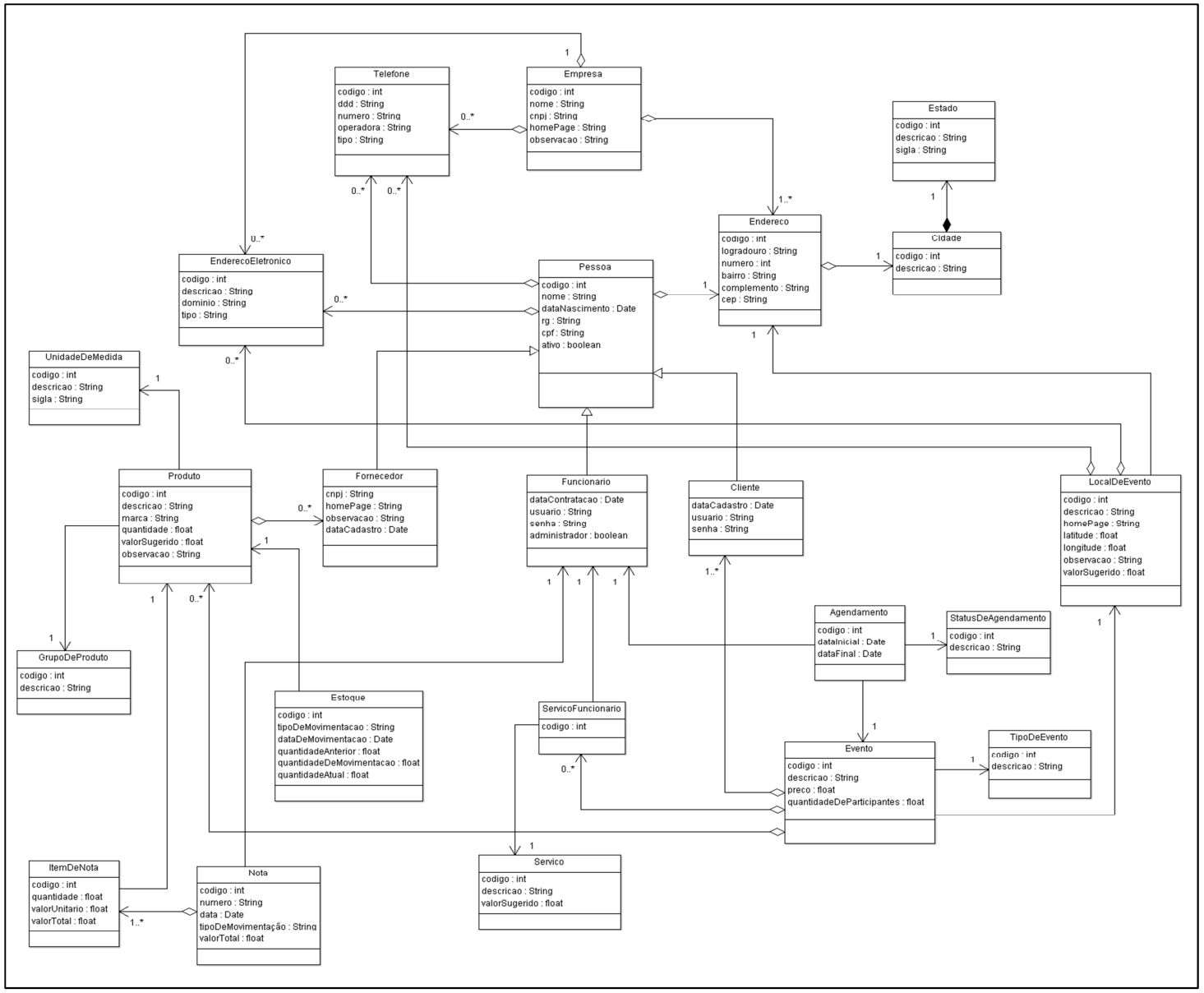

Figura 50 Diagrama de Classes

3.7 MODELAGEM DE ENTIDADE E RELACIONAMENTO

O diagrama entidade relacionamento é baseado na percepção do mundo real que consiste em um conjunto de objetos básicos chamados entidades e nos relacionamentos entre estes objetos.

Ele foi desenvolvido para facilitar o projeto de banco de dados, permitindo a especificação de um esquema de negócio, onde tal esquema representa a estrutura lógica geral do banco de dados (REZENDE, 2005).

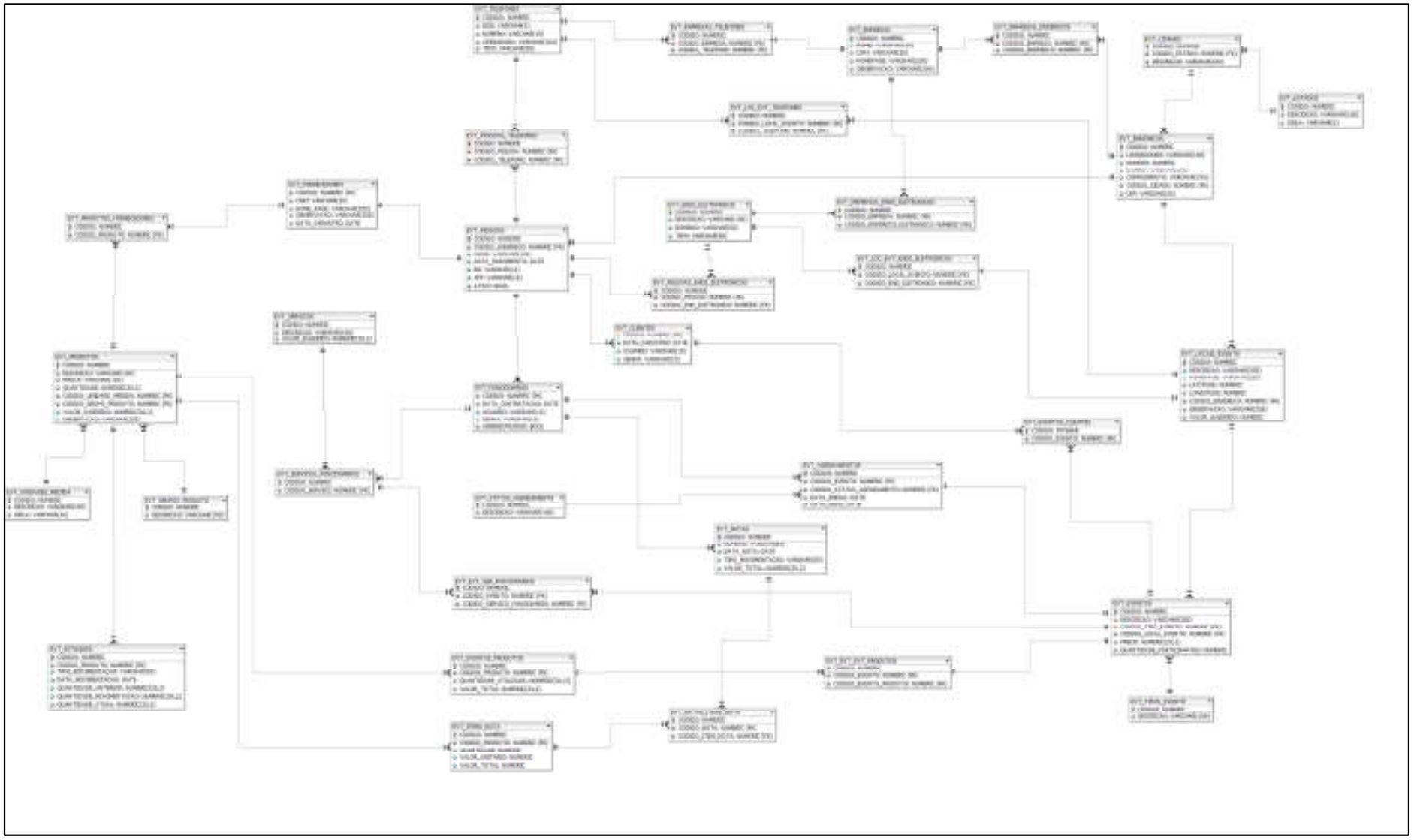

Figura 51 Modelo Entidade-Relacionamento

4 ESTRUTURA DO PROJETO

O neste capitulo será apresentado a metodologia de desenvolvimento adotada para este trabalho o qual consiste em fases e etapas. Para ilustrá-las, será apresentado o diagrama da *Work Breakdown Structure* (WBS) também conhecida como estrutura analítica de trabalho, o diagrama de sequência de atividades e orçamento do sistema.

4.1 ESTRUTURA ANALÍTICA DE TRABALHO

A WBS é um *checklist* que divide todas as partes de um projeto e as tarefas associadas, ou seja, subdivide o trabalho de um projeto em partes menores que podem ser gerenciadas com maior facilidade, fornece uma ilustração detalhada do escopo do projeto, auxilia na montagem da equipe e distribuição do trabalho, facilita a identificação de riscos. Ela é a peça central no planejamento de um projeto, uma vez que ela permite definir o conjunto de atividades que precisa ser executado. É com base na WBS que todos os elementos do projeto são planejados (MARTINS, 2010).

Portanto para a realização do presente projeto serão desenvolvidas as seguintes tarefas, ilustradas na Figura 1, visando à organização do trabalho e a orientação dos resultados desejados.

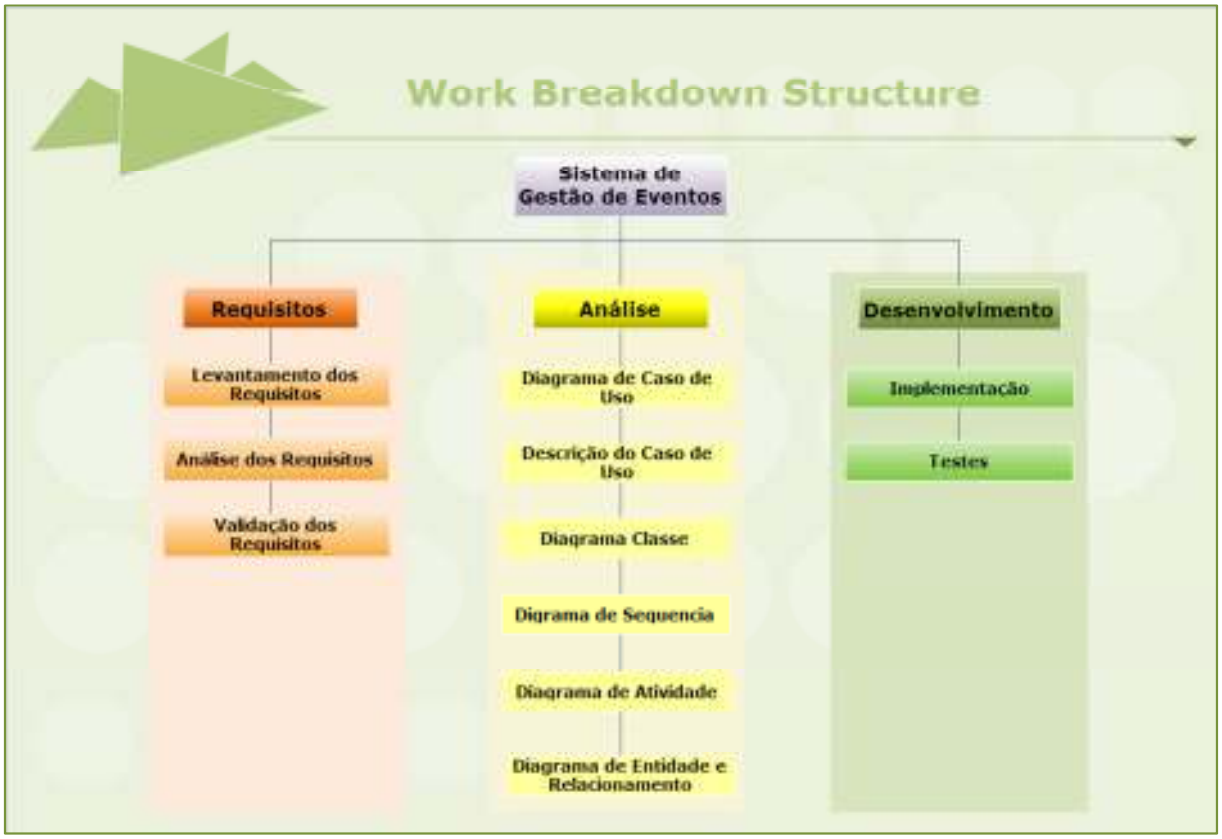

Figura 52 Work Breakdown Structure

4.2 SEQUENCIAMENTO DE ATIVIDADES

O diagrama de sequenciamento de atividades apresenta o tempo de duração para realização de cada atividade desenvolvida no decorrer do projeto.

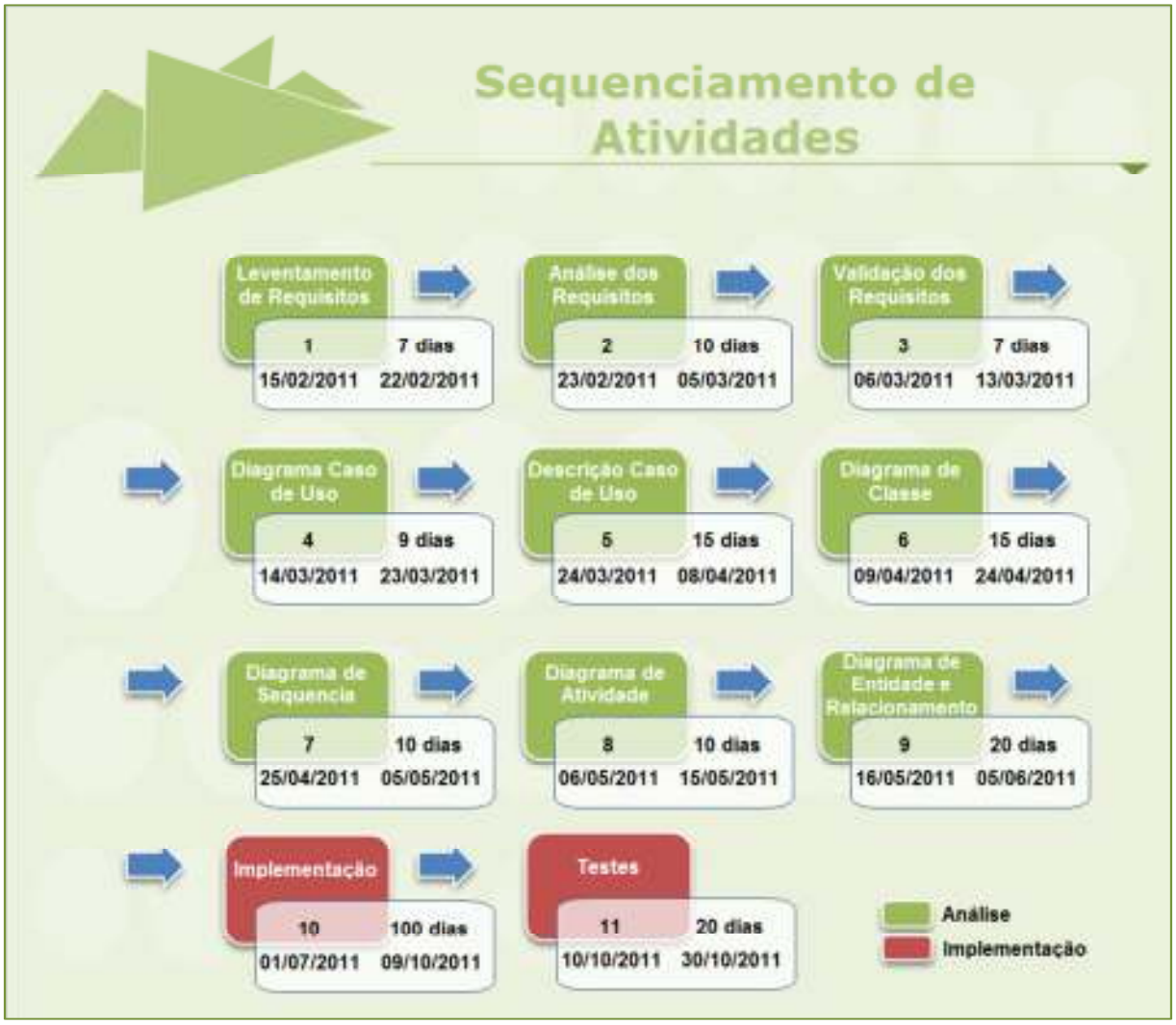

Figura 53 Sequenciamento de Atividades

4.3 ORÇAMENTO

Os recursos necessário para a análise e desenvolvimento do software SuperEvent são:

- 01 Analista de Sistema
- 01 Programador
- 01 Impressora Jato de Tinta
- 01 Notebook

ANALISTA DE SISTEMAS			
Analista	**Quantidade de Horas**	**Valor Hora**	**Total**
Tamires Alves da Silva	103	R$ 30,00	R$ 3.090,00

PROGRAMADOR			
Programador	**Quantidade de Horas**	**Valor Hora**	**Total**
Tamires Alves da Silva	120	R$ 20,00	R$ 2.400,00

TOTAL	R$ 5.490,00

Figura 54 Orçamento - Analista

EQUIPAMENTOS			
Equipamento	**Valor**	**Depreciação Diária**	**Total**
Notebook	R$ 2.600,00	R$ 3,61	R$ 805,30
Impressora	R$ 450,00	R$ 0,62	R$ 139,37

TOTAL	R$ 944,67

Figura 55 Orçamento - Equipamento

VALOR TOTAL DO PROJETO	
Mão de obra	R$ 5.490,00
Equipamentos	R$ 944,67
TOTAL	**R$ 6.434,67**

Figura 56 Orçamento - Valor Total

5 IMPLEMENTAÇÃO DO APLICATIVO

Para a implementação do sistema *SuperEvent* foi utilizado o ambiente de desenvolvimento *Eclipse* com a linguagem de programação Java e o *framework Abobe Flash Builder* com a linguagem *action script*, no qual foi criado e configurado dois projetos, com o objetivo de ganhar independência, reutilizar recursos e diminuir custos com manutenção. A figura 57 ilustra o projeto Java EE e o projeto da camada de visualização desenvolvido em Flex.

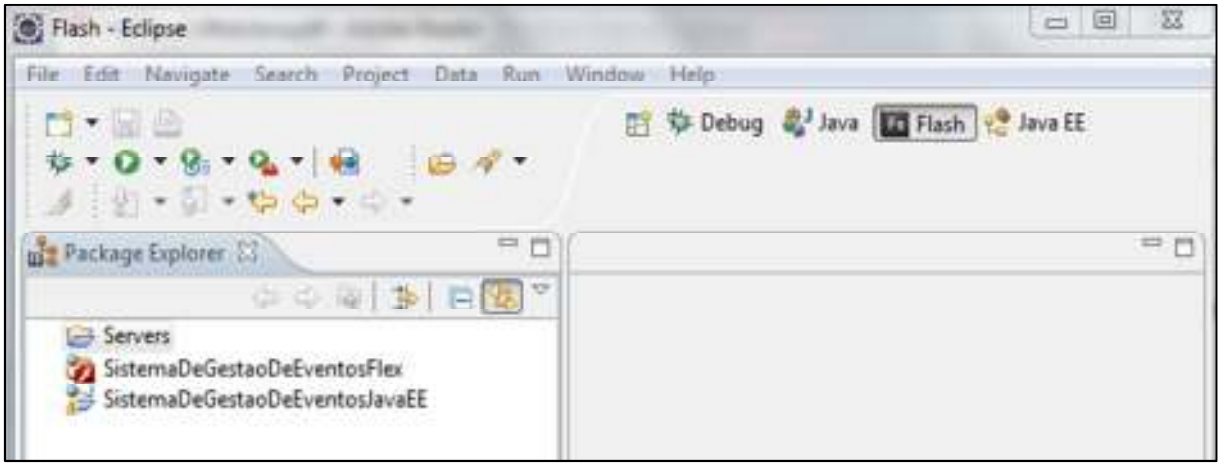

Figura 57 Projeto Java EE e projeto de camada de visualização Flex

- **SistemaDeGestaoDeEventosJavaEE**: Projeto Java responsável por todas as regras de negócio e acesso a dados, utilizando a linguagem de programação Java e o *framework* de persistência *Hibernate.*
- **SistemaDeGestaoDeEventosFlex:** Projeto responsável pela camada de visualização, utilizando a linguagem de programação *Action Script* do *Flash Builder.*

Além do banco de dados *Oracle XE*, responsável por armazenar os dados de forma integra e segura.

5.1 ORGANIZAÇÃO DO PROJETO JAVA

Para uma melhor organização do sistema, os projetos foram divididos em pacotes.

O projeto Java “SistemaDeGestaoDeEventosJavaEE” apresenta sua organização dividida em três pacotes: br.com.sistemadegestaodeeventos.beans, br.com.sistemadegestaodeeventos.dao e br.com.sistemadegestaodeeventos.util, conforme apresentado na figura 58, apoiando-se no padrão MVC:

Figura 58 Organização do Projeto Java

- **br.com.sistemadegestaodeeventos.beans**: Pacote onde se localiza as classes de modelo, as quais possuem os métodos construtor e de acesso a valores de atributos da classe, do tipo *get* e *set*, além dos respectivos mapeamentos da camada de persistência por meio de anotações Java. A Figura 59 ilustra o mapeamento da classe Status de Agendamento.

```java
package br.com.sistemadegestaodeeventos.beans;

import java.io.Serializable;

//Informa que a classe StatusDeAgendamento é persistente
@Entity
//Informa o nome da tabela para aqual está mapeada a classe StatusDeAgendamento
@Table(name="EVT_STATUS_AGENDAMENTO")
public class StatusDeAgendamento implements Serializable {

    //Definição da chave primária
    @Id
    //Nome da coluna mapeada para o atributo codigo
    @Column(name="CODIGO")
    //Tipo da coluna mapeada
    @Type(type="java.lang.Long")
    private Long codigo;

    //Nome da coluna mapeada para o atributo descricao
    //Comprimento máximo de 100 caracteres
    @Column(name="DESCRICAO", length=100)
    private String descricao;

    //Método construtor vazio
    public StatusDeAgendamento() {
    }

    //Método construtor com parâmetros
    public StatusDeAgendamento(Long codigo, String descricao) {
        this.codigo = codigo;
        this.descricao = descricao;
    }

    //Métodos get e set
    public Long getCodigo() {
        return codigo;
    }

    public void setCodigo(Long codigo) {
        this.codigo = codigo;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
}
```

Figura 59 Mapeamento da Entidade “StatusDeAgendamento”

- **br.com.sistemadegestaodeeventos.dao**: Pacote onde se localiza as classes cuja instância é um objeto responsável por acessar os dados pela camada de persistência.

```
package br.com.sistemadegestaodeeventos.dao;

import br.com.sistemadegestaodeeventos.util.HibernateUtil;

public abstract class Dao<E> {

    private Class clazz;

    public Dao(Class c) {
        this.clazz = c;
    }

    public void inserir(E obj) throws Exception {
        Session s = HibernateUtil.getSession();
        try {
            s.beginTransaction().begin();
            s.save(obj);
            s.beginTransaction().commit();
            s.flush();
        } catch (Exception e) {
            s.beginTransaction().rollback();
            throw e;
        } finally {
            s.close();
        }
    }
```

Figura 60 Método “inserir()” da classe genérica DAO

- **br.com.sistemadegestaodeeventos.util**: Pacote onde se localiza a classe *HibernateUtil*, responsável por obter uma única instância de um objeto *SessionFactory , ou seja,* classe Java responsável pelo controle de sessões ou objetos que controlam as sessões de banco de dados e por disponibilizar métodos encarregados pelo controle transacional de informações ou simplesmente, *Session*. A figura 61 ilustra os principais métodos da classe “HibernateUtil”:

```
package br.com.sistemadegestaodeeventos.util;

import org.hibernate.Session;


public class HibernateUtil {

    public static SessionFactory sf = null;

    public HibernateUtil() {
    }

    public static SessionFactory buildSessionFactory() {
        try {
            AnnotationConfiguration cfg = new AnnotationConfiguration();

            cfg.configure("/hibernate.cfg.xml");

            return cfg.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        sf = buildSessionFactory();
        return sf;
    }

    public static Session getSession() {
        return getSessionFactory().openSession();
    }
}
```

Figura 61 Classe utilitária “HibernateUtil”

5.2 ORGANIZAÇÃO DO PROJETO FLEX

O projeto Java “SistemaDeGestaoDeEventosFlex” apresenta sua organização dividida em pacotes, conforme ilustra a figura 62.

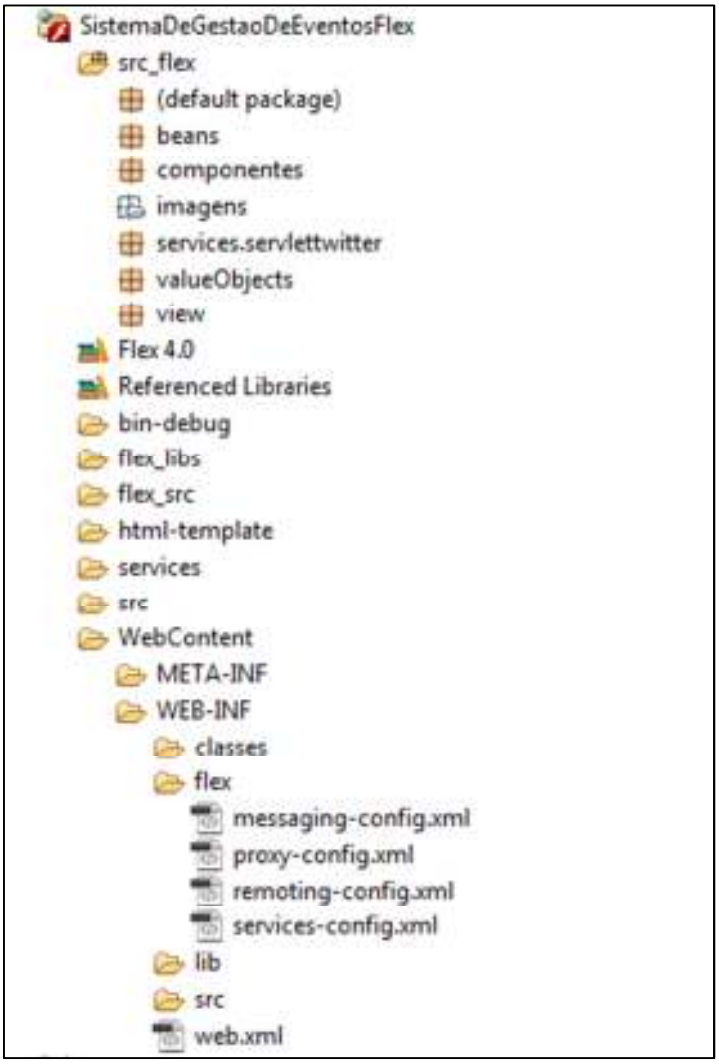

Figura 62 Organização do Projeto Flex

- ***default package***: Neste pacote localiza-se a aplicação ou componente principal de uma aplicação *Flex*, já que os demais componentes e classes serão chamados ou instanciados a partir desse.

- ***beans***: Assim como no projeto Java EE, o pacote *beans* é responsável por armazenar as classes de modelo da aplicação. O principal objetivo de um projeto Flex é realizar a implementação da interface com o usuário, portanto, para cada classe Java deverá haver uma classe *bean* no projeto *Flex*, possibilitando a comunicação e troca de informações entre a camada de visualização, desenvolvida utilizando-se *Adobe Flex,* e a de negócios, utilizando-se a linguagem de programação e tecnologias Java.

- **componentes**: A tecnologia *Adobe Flash Builder* faz uso dos conceitos de componentização e reuso de componentes, possibilitando um maior reaproveitamento de código, além de uma entrega de aplicações de maneira mais fácil e com qualidade superior. No pacote “componentes” estão organizados todos os componentes visuais e não-visuais para uma aplicação Flex, tornando possível a reutilização por outros elementos do projeto. Por exemplo: componentes de calendário, telas de pesquisas, botões configuráveis, entre outros tipos de componentes que podem ser facilmente reutilizados em diferentes pontos do sistema *Flex*.

- **imagens**: Pacote responsável pela organização de elementos gráficos a serem utilizados por componentes visuais de uma aplicação Flex.

- ***servlets.servlettwitter*:** Pacote responsável pela organização das classes de serviços utilizados pela aplicação *Flex* para recuperar dados utilizando-se a API do *Twitter.* Neste pacote localizam-se os métodos de chamada para buscar e atualizar informações de uma conta no *Twitter* por meio de um *Servlet* desenvolvido no projeto Java.

- ***valueObjects*:** Neste pacote também estão localizados as classes de entidade ou negócio utilizados pela aplicação Flex para manipular informações recuperadas de uma conta no Twitter. Enquanto os serviços estão no pacote “servlets.servlettwitter”, o pacote “valueObjects” se encarrega de recepcionar os dados fornecidos pela API do Twitter, assim como disponibilizar essas informações em forma de objetos para que as interfaces visuais possam manipulá-los.

- ***view***: Este pacote é responsável por organizar os formulários visuais desenvolvidos em Flex. Geralmente, todas as telas da camada de visualização podem estar localizadas em apenas um pacote, por exemplo, no pacote *view*. Alguns exemplos são telas de cadastros de clientes, fornecedores, produtos, eventos, entre outras.

Além dos pacotes de codificação de classes, componentes e formulários visuais apresentados anteriormente, também há três diretórios importantes para a configuração de um projeto Flex: *WebContent/WEB-INF/classes, WebContent/WEB-INF/flex* e *WebContent/WEB-INF/lib.*

- Diretório ***WebContent/WEB-INF/classes***: Este diretório é responsável por armazenar as classes Java compiladas que podem ser utilizadas em conjunto com classes e componentes de um projeto Flex.

- Diretório ***WebContent/WEB-INF/flex***: Neste diretório localizam-se os arquivos de configuração de um projeto Flex para que esse possa ser integrado a um projeto Java. Por meio desses arquivos, é possível fazer com que uma classe ou componente da camada de visualização Flex visualize e execute métodos de uma classe Java de outro projeto.

- Diretórios ***WebContent/WEB-INF/lib***: Neste diretório localizam-se todos os arquivos de bibliotecas e APIs necessárias para o projeto Flex. Por exemplo, um arquivo na extensão SWC responsável por fornecer métodos e recursos da API Google Maps.

5.3 INTERFACES DO SISTEMA

Ao acessar o sistema, abrirá a página inicial contento um menu com três itens: Cadastros, Movimentações e Relatórios. A Figura 63 ilustra a página inicial do sistema SuperEvent.

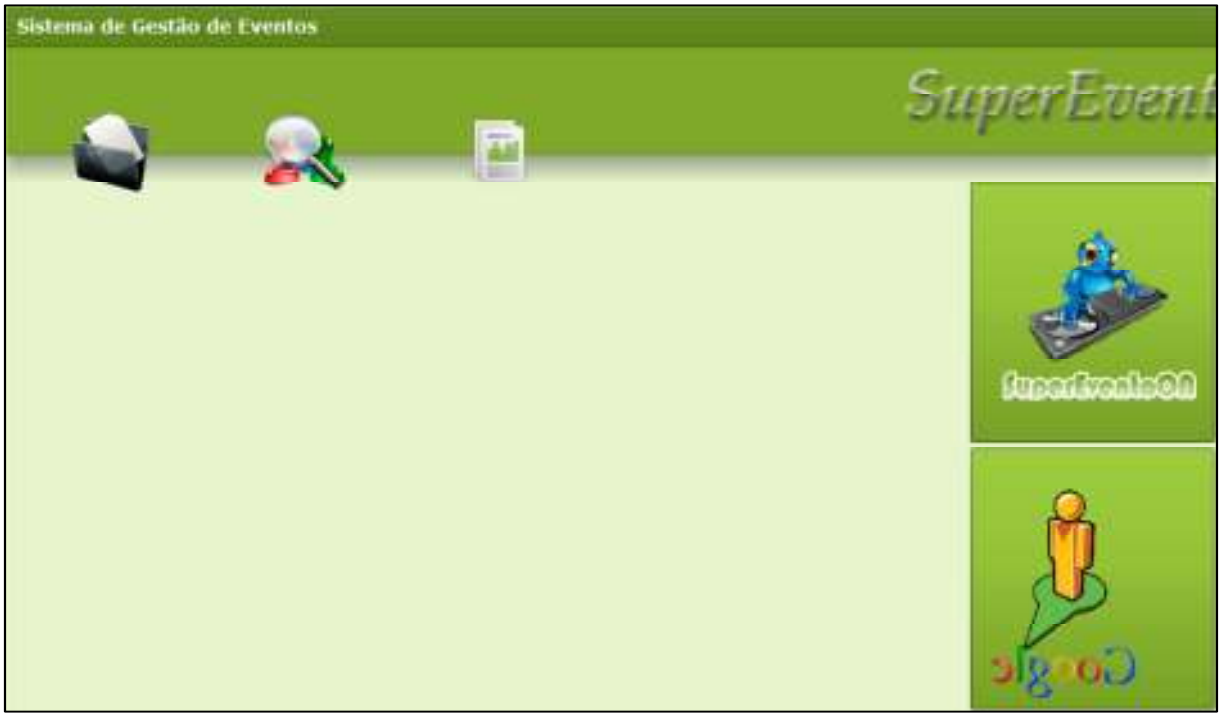

Figura 63 Página inicial do sistema SuperEvent

Ao selecionar o menu Cadastros, abrirá um submenu onde se encontra os cadastros de locais de eventos, clientes, funcionários, produtos, fornecedores, entre outros.

Ao selecionar o menu Movimentações, abrirá um submenu onde se encontra as movimentações de eventos, agenda, entre outros.

Ao selecionar o menu Relatórios, abrirá um submenu onde se encontra os relatórios de eventos agendados, eventos realizados, entre outros.

O usuário ao desejar cadastrar um local de evento deverá escolher a opção local de evento no menu Cadastros, o qual será apresentado um formulário de cadastro de local de evento com quatro abas: dados gerais, locais de evento, telefones e endereços eletrônicos, como ilustra a Figura 64, com as opção de inserir, alterar, remover e limpar o local de evento e uma tabela listando todos os locais de eventos cadastrados no sistema *SuperEvent*.

Na aba dados gerais o usuário deverá preencher apenas os campos código, descrição e *home page*, pois os campos latitude e longitude serão preenchidos

automaticamente ao cadastrar o endereço do local de evento na aba local de evento.

A Figura 64 ilustra o processo de comunicação do *SuperEvent* com o *Google Maps.*

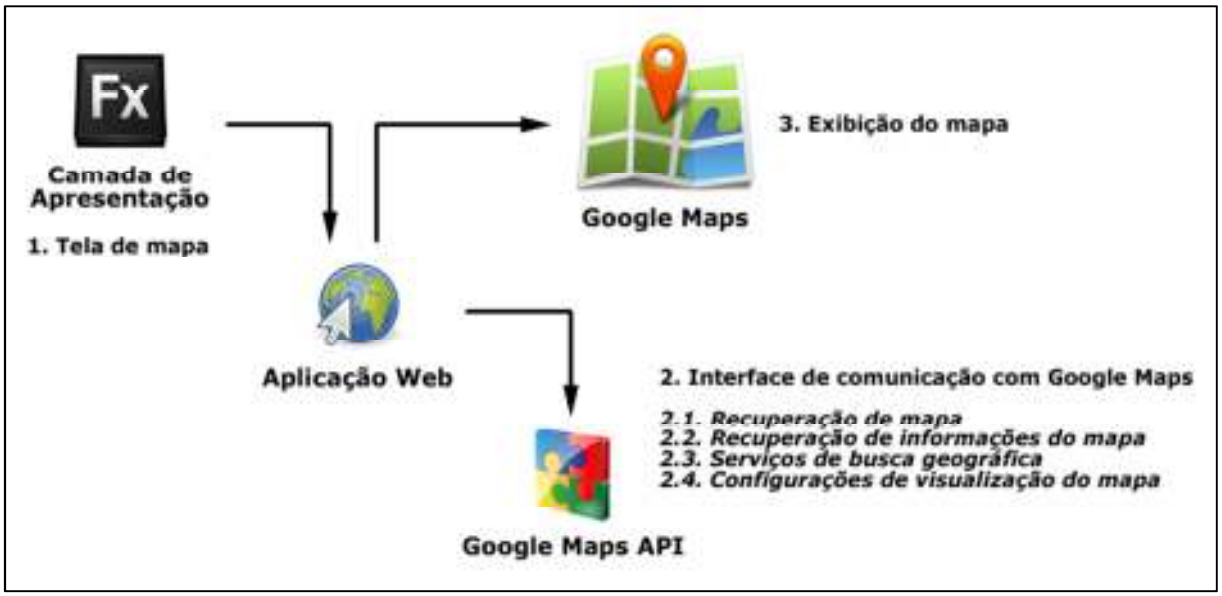

Figura 64 SuperEvent X Google Maps

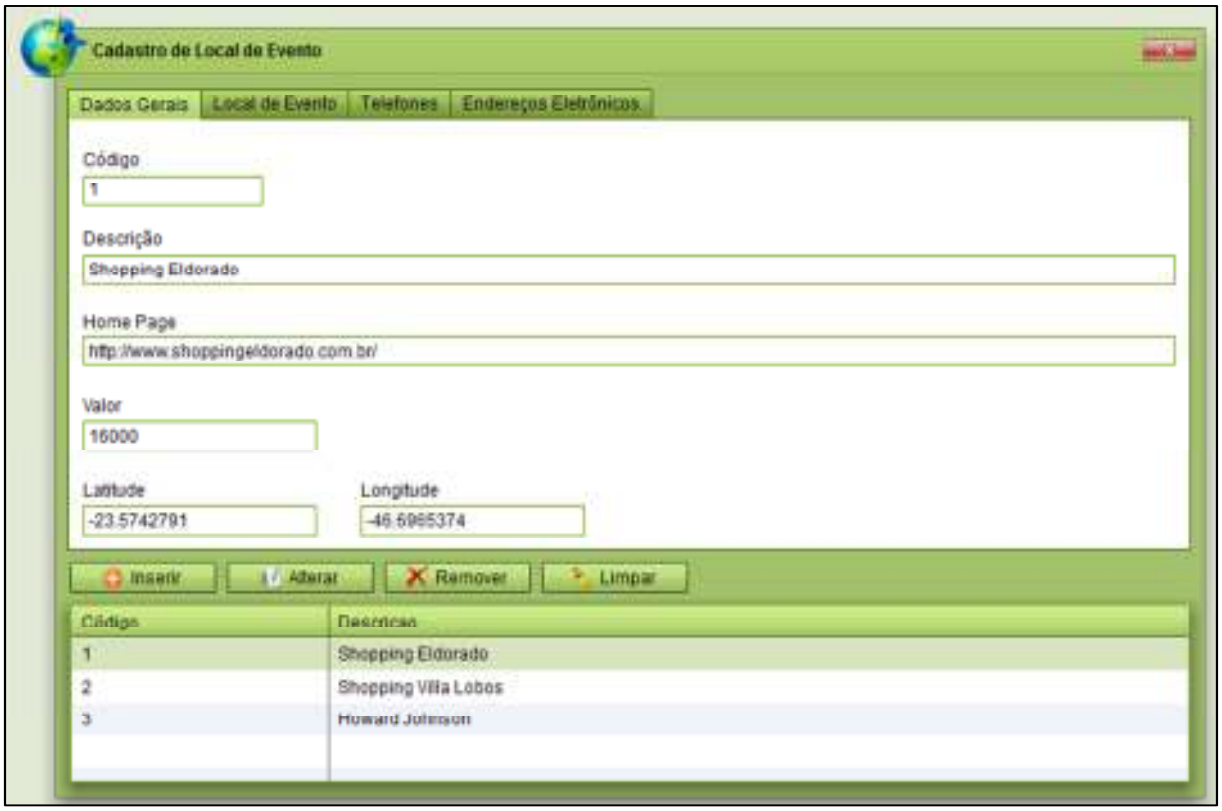

Figura 65 Tela de cadastro de local de evento - Dados gerais

Na aba local de evento o usuário deverá inserir um endereço e em seguida clicar no botão pesquisar para ser realizada a busca do local do evento no *Google Maps.* A Figura 65 ilustra a localização do local do evento no *Google Maps.*

Figura 66 Tela de cadastro de local de evento - Local de evento

```
private function pesquisar(event:Event):void
{
    var geocoder:ClientGeocoder = new ClientGeocoder();

    geocoder.addEventListener(
        GeocodingEvent.GEOCODING_SUCCESS,
        function(event:GeocodingEvent):void
        {
            var placemarks:Array = event.response.placemarks;
            if (placemarks.length > 0)
            {
                var marker:Marker = new Marker(placemarks[0].point);
                marker.openInfoWindow(new InfoWindowOptions({width: 250, height: 70, content: "Latitude: "
                        + marker.getLatLng().lat() + "\n" + "Longitude: " + marker.getLatLng().lng()}));
                map.clearOverlays();
                map.addOverlay(marker);
                map.FlyTo(marker.getLatLng(), 16, new Attitude(20,30,0), 3);
                txtLatitude.text = String(marker.getLatLng().lat());
                txtLongitude.text = String(marker.getLatLng().lng());
            }
        });
    geocoder.addEventListener(
        GeocodingEvent.GEOCODING_FAILURE,
        function(event:GeocodingEvent):void
        {
            Alert.show("Falha ao carregar mapa");
        }
    );
    geocoder.geocode(txtEndereco.text);
}
```

Figura 67 Método pesquisar local de evento

Figura 68 Tela de localização dos eventos agendados utilizando o Google Maps

```
public function criarMarker(latlng:LatLng, name:String, address:String, type:String):void {
    var marker:Marker = new Marker(latlng, new MarkerOptions({icon: new categorias[type].icon,
                                iconOffset: new Point(-16, -32)}));

    var html:String = "<b>" + name + "</b> <br/>" + address;
    marker.addEventListener(MapMouseEvent.CLICK, function(e:MapMouseEvent):void {
        marker.openInfoWindow(new InfoWindowOptions({contentHTML:html}));
    });
    categorias[type].markers.push(marker);
    map.addOverlay(marker);
}

private function visualizarCategoria(type:String):void {
    for (var i:Number = 0; i < categorias[type].markers.length; i++) {
        var marker:Marker = categorias[type].markers[i];
        if (!marker.visible) {
            marker.visible = true;
        } else {
            marker.visible = false;
        }
    }
}
```

Figura 69 Métodos para criar e visualizar marcadores por tipo de evento

Figura 70 Localização do local de evento utilizando o Google Maps e o Street View

```
private function clicarMapa(event:MapMouseEvent):void
{
    marker.setLatLng( event.latLng );
    ExternalInterface.call( "setPanorama", event.latLng.lat(), event.latLng.lng() );
}


private function handleResultListaLocalDeEvento(event:ResultEvent):void {
    locaisDeEvento = event.result as ArrayCollection;

    cmbLocaisDeEvento.selectedItem = locaisDeEvento.getItemAt(0);
}

private function handleFault(event:FaultEvent):void {
    Alert.show(event.message.toString());
}

private function navegarLocalDeEvento():void {
    var localDeEvento:LocalDeEvento = cmbLocaisDeEvento.selectedItem as LocalDeEvento;

    marker.setLatLng( new LatLng(localDeEvento.latitude, localDeEvento.longitude) );

    map.setCenter( new LatLng(localDeEvento.latitude, localDeEvento.longitude) );

    ExternalInterface.call( "setPanorama", marker.getLatLng().lat(), marker.getLatLng().lng() );
}
```

Figura 71 Métodos para localizar o local de evento no Google Maps e no Street View

O usuário ao desejar enviar um ou mais *tweets* para o *Twitter* através do sistema *SuperEvent* deverá selecionar o menu Movimentações da tela principal e escolher a opção Mensagens - Twitter, o qual será apresentado o formulário, como ilustra a Figura 67, com as opção de enviar, limpar e carregar os tweets.

A Figura 72 ilustra o processo de comunicação do *SuperEvent* com o *Twitter*.

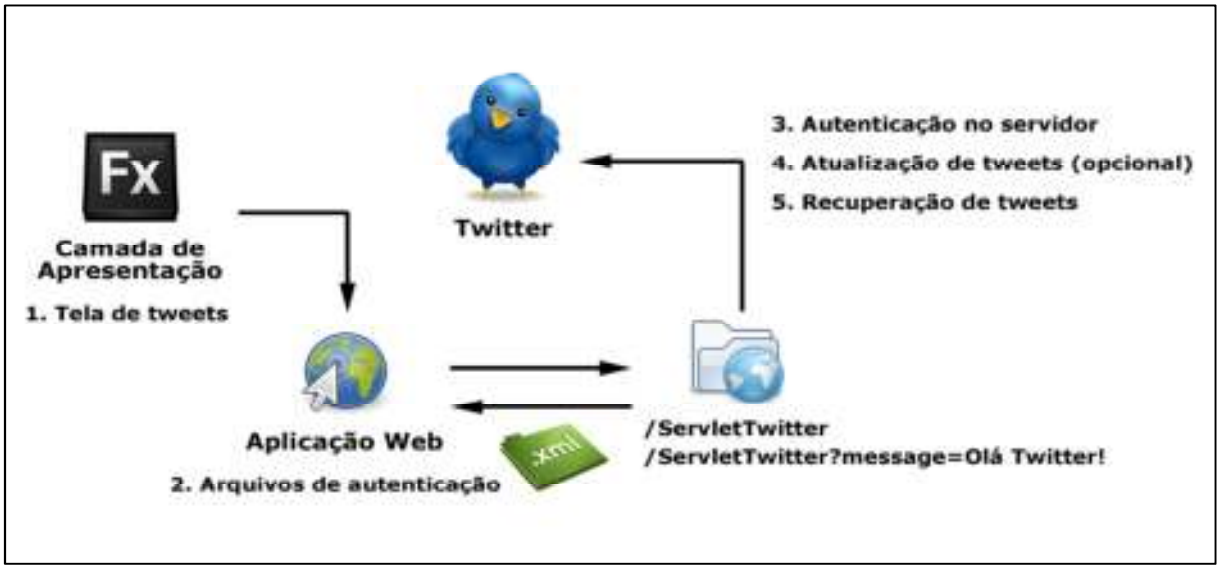

Figura 72 SuperEvent X Twitter

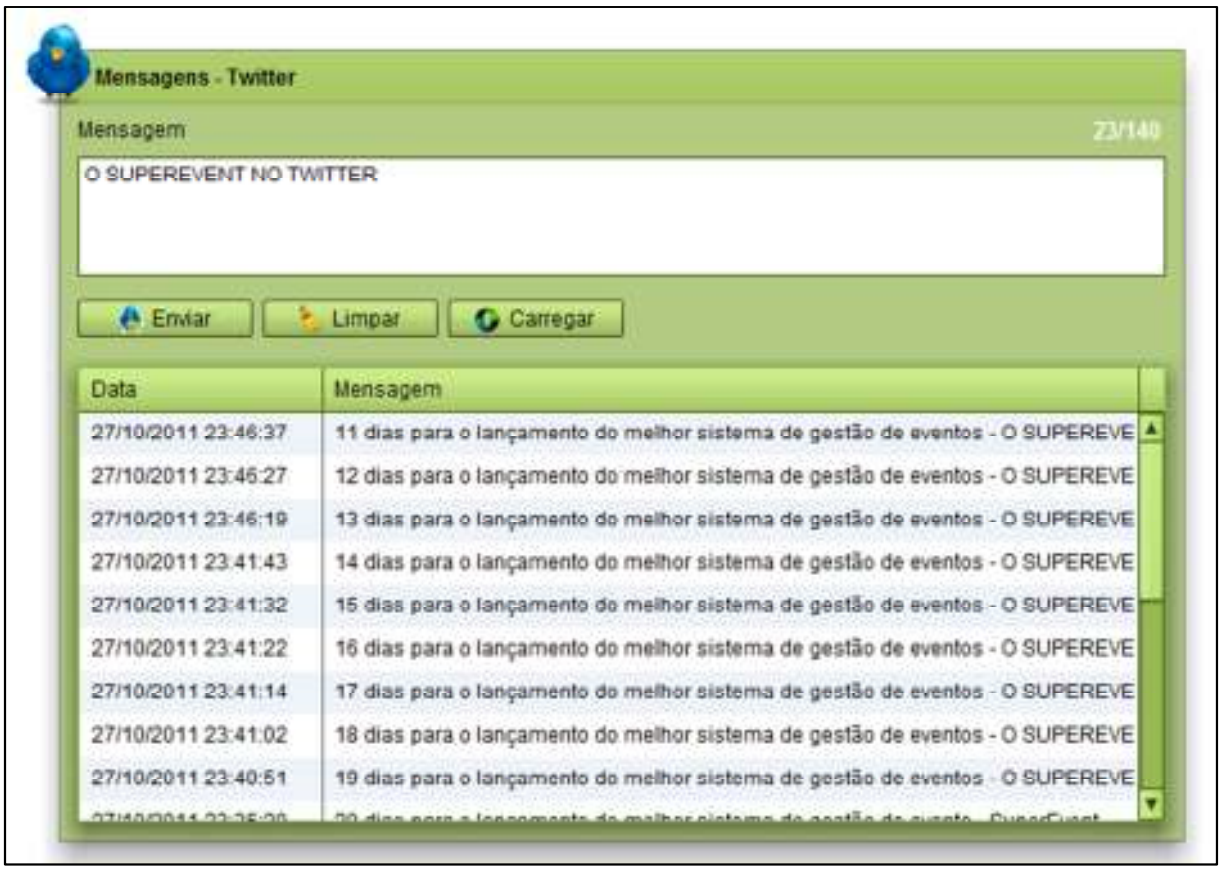

Figura 73 Tela de enviar tweets

```
ConfigurationBuilder cb = new ConfigurationBuilder();
cb.setDebugEnabled(true);
cb.setOAuthConsumerKey(bundle.getString("OAuthConsumerKey"));
cb.setOAuthConsumerSecret(bundle.getString("OAuthConsumerSecret"));
cb.setOAuthAccessToken(bundle.getString("OAuthAccessToken"));
cb.setOAuthAccessTokenSecret(bundle.getString("OAuthAccessTokenSecret"));
TwitterFactory factory = new TwitterFactory(cb.build());
Twitter twitter = factory.getInstance();
if (update) {
    twitter.updateStatus(message);
}
List<Status> statuses = twitter.getUserTimeline();
DateFormat formato = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
StringBuilder xmlMensagens = new StringBuilder("");
xmlMensagens.append("<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>");
xmlMensagens.append("<root>");
for (Status status : statuses) {
    xmlMensagens.append("<tweet>");
    {
        Date data = status.getCreatedAt();
        xmlMensagens.append("<data>").append(formato.format(data)).append("</data>");
        xmlMensagens.append("<mensagem>").append(status.getText()).append("</mensagem>");
    }
    xmlMensagens.append("</tweet>");
}
xmlMensagens.append("</root>");
out.print(xmlMensagens.toString());
```

Figura 74 Servlet para envio de tweets

Figura 75 Tweets enviados do Sistema SuperEvent

6 CONCLUSÃO

Cada vez mais, a área de tecnologia da informação tem apresentado um grande crescimento no Brasil e no mundo. O uso de *softwares* específicos tem contribuído para esse aumento, devido à necessidade de um maior controle e levantamento de dados de processo de produção.

O *software* SuperEvent foi desenvolvido para o mercado de gerenciamento de eventos utilizando soluções inovadoras e criativas, agilizando processos, reduzindo despesas, ampliando a comunicação entre os participantes, por meio do uso de uma das mais utilizadas ferramentas da rede social, o *Twitter*, e a utilização do mecanismo de localização, *Google Maps*. Tudo de forma flexível, facilitando o controle das tarefas antes, durante e após o evento, satisfazendo as necessidades dos clientes.

Por meio da elaboração de diagramas da UML, pode-se obter uma visão completa e detalhada sobre o funcionamento do sistema, contribuindo para um maior entendimento e simplificação dos processos do *software* a ser desenvolvido.

A implementação da aplicação utilizando-se o conceito de MVC ou construção em camadas possibilitou a separação das regras de negócio das regras de persistência, isolando a camada de visualização para ser responsável apenas pela interação homem-máquina, facilitando a manutenção e evolução do *software*.

Durante o desenvolvimento do sistema, houve dificuldades, especificamente com a plataforma RIA do *Adobe Flex* e sua integração com o código Java, porém essas dúvidas foram sanadas por meio da leitura da documentação do *Adobe Flash Builder*.

As tecnologias pesquisadas e utilizadas no trabalho mostraram-se eficientes no desenvolvimento de aplicações *web* dinâmicas, provando-se a capacidade da linguagem de programação Java. Tratando-se de um sistema desenvolvido para a *web*, percebeu-se uma grande vantagem sobre a portabilidade, tal que, a aplicação poderá oferecer todas as informações sobre a gestão de eventos de uma maneira mais rápida e eficiente.

Para o desenvolvimento de trabalhos futuros, pretende-se aplicar técnicas e algoritmos para permitir que o mapa seja mais dinâmico, desejando-se implementar funções geográficas para cálculo e controle de rotas entre um participante e o evento, assim como a obtenção de demais informações sobre a localização.

REFERÊNCIAS

ADOBE. **Website oficial – Adobe**. Disponível em: <http://www.adobe.com/br/>. Acesso em: 01 de junho de 2011.

APACHE TOMCAT. **Website oficial – Apache Tomcat**. Disponível em: <http://tomcat.apache.org/>. Acesso em: 01 de junho de 2011.

GOMES, Helton Simões. **Brasil é 2º país em que Twitter mais cresce, mas 68% dos jovens não acessam**. Disponível em: <http://www.band.com.br/jornalismo/tecnologia/conteudo.asp?ID=100000339972>. Acesso em: 6 de junho de 2011.

BAUER, Christian; KING, Gavin. **Java Persistence com Hibernate**. 2ª Edição. Tradução Gustavo Vinocur. Rio de Janeiro: Editora Ciência Moderna Ltda., 2007.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML Guia do Usuário**. 2ª Edição. Tradução Fábio Freitas da Silva e Cristiana de Amorim Machado. Rio de Janeiro: Elsevier, 2005.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML Essencial – Um breve guia para a linguagem-padrão de modelagem de objetos**. 2ª Edição. Tradução de Vera Pezerico e Christian Thomas. Porto Alegre: Bookman, 2000.

CAMARGO, Dalton. **Tutorial de IReport** . Disponível em <http://javafree.uol.com.br/artigo/3154/Tutorial-de-IREPORT.html>. Acesso em: 30 de março de 2011.

COMSCORE. **Facebook Captures Top Spot among Social Networking Sites in India**. Disponível em: <http://www.comscore.com/Press_Events/Press_Releases/2010/8/Facebook_Captures_Top_Spot_among_Social_Networking_Sites_in_India>. Acesso em: 6 de junho de 2011.

DEITEL, H.M; DEITEL, P.J. **Java - Como programar**. 6ª Edição. Tradução de Carlos Arthur Lang Lisboa. Pearson, Prentice Hall, 2005.

GONÇALVES, Edson. **Desenvolvendo aplicações web com Jsp, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax**. 1ª Edição. Rio de Janeiro: Ciência Moderna, 2007.

GOOGLE DEVELOPER GUIDE. **Google Maps Java API Data**. Disponível em: <http://code.google.com/intl/ptR/apis/maps/documentation/mapsdata/developers_guide_java.html>. Acesso em: 05 de junho de 2011.

JAVAFREE. Fórum. Disponível em: <http://javafree.uol.com.br/wiki/Java>. Acesso em: 30 de maio de 2011.

LEE, Richard C; TEPFENHART, William M; **Uml e C++ Guia Prático de Desenvolvimento Orientado a Objeto**. 1ª Edição. Tradução de Celso Roberto Paschoa. São Paulo: MAKRON Books, 2001.

MARTINS, José Carlos Cordeiro; **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML**. 5ª Edição. Rio de Janeiro: Brasport, 2010.

ORACLE. **JDK 5.0 Documentation**. Disponível em: <http://download.oracle.com/javase/1.5.0/docs/>. Acesso em: 30 de maio de 2011.

ORACLE. **Oracle Database 10*g* Express Edition**. Disponível em: < http://www.oracle.com/technetwork/database/express-edition/overview/index.html> Acesso em: 31 de maio de 2011.

PINGDOM. **Vinte e Nove social networks that have at least one milion visitors per day. Disponível em:** <http://royal.pingdom.com/2011/03/25/social-networks-one-million-visitors-per-day/>. Acesso em: 6 de junho de 2011.

REVISTA ÉPOCA. **Onde os brasileiros se encontram**. Disponível em: <http://revistaepoca.globo.com/Revista/Epoca/1,,EMI143701-15224,00.html>. Acesso em: 6 de junho de 2011.

REZENDE, Denis Alcides; **Engenharia de Software e Sistema de informação**. Rio de Janeiro: Brasport, 2005.

SCHMITZ, Daniel. **Dominando Adobe Flex 4**. 1ª Edição. Bauru: Canal6, 2010.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistemas de banco de dados.** 5ª Edição. Tradução Daniel Vieira. Rio de Janeiro: Elsevier, 2006.

SUN, Java. **Java 2 Platform, Enterprise Edition (J2EE) Overview.** Disponível em: <http://java.sun.com/j2ee/setstandard.html>. Acesso em: 30 de maio de 2011.

SUN, Developers. **Overview of the Java ME Platform.** Disponível em: <http://developers.sun.com/mobility/getstart/articles/survey/>. Acesso em: 30 de maio de 2011.

TIOBE. **Tiobe programming community index**. Disponível em: < http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 27 de outubro de 2011.

TWITTER4J. **Website oficial**. Disponível em: <http://twitter4j.org/en/index.jsp>. Acesso em 6 de junho de 2011.

VARGAS, Ricardo; **Gerenciamento de Projetos Estabelecendo Diferenciais Competitivos**. 6º Edição. São Paulo: Editora BRASPORT Livros e Multimídias, 2005.

ANEXOS

CRONOGRAMA PARA A ESTRUTURA DO DESENVOLVIMENTO

O cronograma das atividades utilizado durante o desenvolvimento deste trabalho de conclusão do curso é apresentado a seguir: