Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis - IMESA

MAICON ALLAN GUIZELINI

SISTEMA DE CONTROLE DE CONDOMÍNIOS

Assis

2013

MAICON ALLAN GUIZELINI

SISTEMA DE CONTROLE DE CONDOMÍNIOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito de conclusão do Curso Superior de Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Osmar Aparecido Machado

Área de Concentração: Desenvolvimento de Sistema

Assis

2013

FICHA CATALOGRÁFICA

GUIZELINI, Maicon Allan

Sistema de Controle de Condomínios / Maicon Allan Guizelini. Fundação Educacional do Município de Assis - FEMA - Assis, 2013.

50 Páginas.

Orientador: Prof. Dr. Osmar Aparecido Machado

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis - IMESA.

1.Programas 2.Controle de Condomínios Residenciais

CDD: 001.61

Biblioteca da FEMA

CONTROLE DE CONDOMÍNIOS RESIDENCIAIS

MAICON ALLAN GUIZELINI

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso Superior de Análise e Desenvolvimento de Sistemas, analisado pela seguinte comissão examinadora:

Orientador: Prof. Dr. Osmar Aparecido Machado

Analisador: Prof. Domingos de Carvalho Villela Junior

Assis

2013

RESUMO

O trabalho refere-se ao desenvolvimento de uma aplicação em ambiente Java Web, com foco na principal na gestão de condomínios. O aplicativo disponibiliza desde consultas simples sobre o funcionamento do condomínio, até regras, compra e venda de lotes, casas e apartamentos, consultas avançadas, mensalidade, segurança, sugestões e oportunidades. Para o desenvolvimento Web será usado á linguagem JAVA. Já a aplicação será desenvolvida em JAVAWEB com a biblioteca PrimeFaces que tem muitos componentes para uma aplicação bem planejada e com muitas ferramentas para design, juntamente com banco de dados MySQL Server. Nesse sentido o trabalho inicia-se com uma revisão da bibliografia, apresentando conceitos, padrões e metodologias das linguagens de programação, ferramentas e componentes utilizados para o desenvolvimento da aplicação. Por fim, o aplicativo desenvolvido poderá ser comercializado por qualquer tipo de condomínio.

Palavras chave: Condomínio, JAVA.

ABSTRACT

The work concerns the development of a Java Web application environment, focusing on the main property management. The application is available from simple queries on the operation of the condominium, to rules, purchase and sale of lots, houses and apartments, advanced queries, monthly fees, security, ideas and opportunities. For Web development will be used Java language. Since the application will be developed in JavaWeb with PrimeFaces library that has many components to an application well planned and with many tools for design, along with the MySQL database server. In this sense the work begins with a review of the literature, presenting concepts, standards and methodologies of programming languages, tools and components used for application development. Finally, the developed application can be sold by any type of condominium.

Keywords: Townhouse, JAVA.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Com a rápida evolução tecnológica, cada vez mais as pessoas ficam dependentes dos recursos e de aparatos tecnológicos, como computadores, celulares, tvs digitais, tablets, dentre outros. Surgem diariamente novas formas de tecnologias, novos projetos e por conta dessa rápida evolução faltam mão-de-obra especializada e softwares em diversas áreas.

Nesse sentido, este estudo desenvolve um sistema de administração de condomínios, que utiliza alguns conceitos relacionados às novas tecnologias, como controle de entrada e saídas de visitantes, com identificação por câmeras por exemplo.

1.1. OBJETIVO

O objetivo deste trabalho é desenvolver um software para controle e gestão de condomínios, integrando desde o controle de acesso ao ambiente, pelas portarias, até o controle de pagamentos e recebimentos referentes às taxas específicas deste tipo de empreendimento. Para tanto será utilizada a linguagem JAVA (Web) com os componentes PrimeFaces para uma melhor aplicação e design juntamente com o banco de dados MySQL.

1.2. JUSTIFICATIVAS

O Software desenvolvido com o intuito de ajudar, resolver problemas do dia-a-dia de moradores e funcionários de condomínios automatizando o sistema de gerenciamento, cadastros e segurança para uma melhor administração. Os condomínios tem se expandido rapidamente e em grande escala em todo o país e na região de Assis isto não é diferente. Entretanto faltam sistemas eficientes para a

gestão especializada deste segmento, como controlar entrada e saídas de visitantes para maior segurança do local, dentre outras necessidades.

Além disso, é uma boa oportunidade de negócio que se apresenta e nesse sentido o sistema poderá vir a se tornar um referencial para gestão deste tipo de empreendimento.

1.3. FOCO DO TRABALHO

Estudar e procurar resoluções, formas para melhorar e ajudar os funcionários de condomínios a administrar melhor e fazer controle mais específico do local e segurança. O Foco Principal é procurar ajudar moradores com problemas em suas casas, se está bem cuidado, se tem alguma opinião para sobre o local, se precisa melhorar em algum setor do condomínio e também ajudará o síndico com opiniões de moradores sobre normas e regras.

Por fim, o foco do presente estudo é desenvolver um Sistema específico na resolução de problemas e formas para melhorar a vida de moradores e funcionários dos condomínios.

1.4. MOTIVAÇÃO

O tema do trabalho é um assunto pouco conhecido. Nesse sentido, é uma área que merece estudada, por causa da quantidade de pessoas que moram em condomínios e vem crescendo a cada dia, especialmente pela questão de segurança.

Dessa forma, estudar essa área e desenvolver um trabalho que tenha uma nova maneira de interação entre moradores e funcionários dos condomínios é instigante e ao mesmo tempo, pode conduzir à produção de um produto (software) que poderá ser comercializado junto a outros condomínios.

1.5. PERSPECTIVA DE CONTRIBUIÇÃO

O Software será uma alternativa de gestão para os condomínios, provendo maior agilidade e controle em suas atividades. Além disso, existe certa carência de aplicativos dessa natureza no mercado, por isso, o aplicativo pode vir a crescer por meio da implementação de melhorias, de forma que atenda a outras necessidades dos clientes, incorporando novas ideias e tecnologias.

Por fim, é uma contribuição significativa para em nicho de mercado que cresce continuamente e que carece de investimentos nos processos de gestão.

1.6. METODOLOGIA

A metodologia de análise utilizada neste projeto é a Análise Orientada a Objetos (AOO). Esta metodologia tem como característica ajudar na programação, na organização e na estrutura resolvendo muitos problemas que uma implementação ruim pode causar e irá enfrentar durante o período de desenvolvimento.

Contempla ainda os aspectos metodológicos, o levantamento bibliográfico e uma revisão da literatura, realizada em livros, artigos e revistas, em forma impressa e pela internet.

Quanto ao desenvolvimento propriamente dito, o projeto será construído utilizando os seguintes recursos:

- Sistema Operacional Windows 8
- Java
- Netbeans 7.4
- MySQL
- Apache Tomcat.
- PrimeFaces.

1.7. ESTRUTURA DO TRABALHO

O Trabalho será dividido em quatro capítulos distintos, sendo:

1° Capítulo - Tratará dos objetivos do projeto, do público alvo a ser atendido, das metodologias de desenvolvimentos, que são as linguagens e as tecnologias que serão usados no projeto.

2° Capítulo - Tratará do planejamento do projeto, a estrutura analítica do projeto, as definições do sistema, tais como Casos de Uso e Narrativas, Diagrama de Classes, Diagrama de Entidade Relacionamento, Diagrama de Atividades, Diagrama de Sequência e Cronograma.

3° Capítulo - Tratará das metodologias que serão ut ilizadas no projeto.

4º Capítulo - Tratará das especificações de custos, gerência de custos, os recursos necessários para o desenvolvimento, a estimativa de custos para as atividades, recursos e o orçamento final para o desenvolvimento do projeto e serão mostrados as modelagens dos sistemas.

5° Capítulo - Finalmente, será apresentada nesse ca pítulo a conclusão parcial do projeto.

2. REVISÃO DA LITERATURA

A revisão da literatura tem por objetivo apresentar os principais conceitos na literatura, sobre o assunto investigado. Isto é, deve-se, buscar em livros, revistas e artigos as teorias mais modernas, mais recentes que tratam do assunto. Esta revisão é uma garantia de que as ferramentas aqui apresentadas fazem jus ao uso, ou seja, pela natureza desse estudo, de conclusão de curso, é importante que as tecnologias utilizadas representem o estado da arte sobre a área, nesta área do conhecimento.

2.1. ORIENTAÇÃO A OBJETOS

O termo orientação a objetos pressupõe uma organização de software em termos de coleção de objetos discretos incorporando estrutura e comportamento. Esta organização é essencialmente diferente do desenvolvimento tradicional de software, onde estruturas de dados e rotinas são desenvolvidas de forma apenas fracamente acopladas. A grande diferença da programação orientada a objetos em relação a outros tipos de programação que também permitem definir estruturas e operações que está no conceito de herança.

Herança é um mecanismo que através do qual as definições existentes podem ser estendidas. Junto com a herança vem o polimorfismo com grande importância, que permite selecionar várias funcionalidades que um programa poderá utilizar de forma dinâmica, durante a sua execução. (FOWLER, Martin. UML Essencial 2ª Edição, 2000). Uma vantagem de usar orientação a objetos é que ajuda em muito na organização do projeto e com isso diminui muito na escrita do código e referências (polimorfismo), além de concentrar suas responsabilidades nos pontos certos do projeto, deixando sua aplicação flexível e encapsulando a lógica de negócios.

2.2. UML

UML é uma linguagem de modelagem e não como pensam alguns, que UML é um método. A linguagem de modelagem é uma notação, utilizada por métodos para mostrar, expressar projetos, é uma maneira de sugerir como e quais passos devem ser seguidos para elaboração de um projeto.

A modelagem em muitos aspectos é a parte mais importante de um método, ela seria a parte principal para a comunicação ou discussão de um projeto. Em softwares que usa orientado a objetos a UML é uma técnica muito importante e necessária para uma boa análise e desenvolvimento de um projeto (FOWLER, Martin. UML Essencial 2ªEdição, 2000).

A principal razão de usá-la é que essa linguagem permite a comunicação de alguns conceitos claramente, com bastante precisão e detalhes importantes. Permite ter uma visão geral do projeto, para melhores esclarecimentos, partes que podem ser questionadas do projeto que precisam ser mais bem entendidas e trabalhadas. (FOWLER, Martin. UML Essencial 2ªEdição, 2000).

Ao todo, nove diagramas compõem esta linguagem. Nas seções seguintes, serão expostos os diagramas, que serão utilizados neste estudo como parte da documentação da análise do aplicativo.

2.2.1. Casos de Uso

- ✓ **Ator**

É uma função que o usuário exerce em relação ao sistema. Um único ator pode de exercer muitos casos de uso e um único caso de uso pode ter vários atores o exercendo. Um ator não precisa necessariamente ser uma pessoa, pode ser um sistema externo que precisa de alguma informação, ou um acesso no sistema.

- ✓ **Associação entre Casos de Uso**

A associação de inclusão aparece quando uma parte do comportamento que é parecido em mais de um caso de uso. Então se cria um caso de uso de avaliação separado para cada situação e se referir a ele a partir dos casos de uso originais. Um caso de uso pode ter vários pontos de extensão, e um caso de uso de extensão pode ter ou aumentar um ou mais pontos de extensão, você indica quais serão estendidos na linha entre casos de uso no diagrama.

Generalização se usa quando há um caso de uso semelhante o outro, mas faz tem alguma função a mais. A generalização e a extensão permitem que você possa dividir um caso de uso, geralmente é divido quando um caso de uso fica muito complicado.

Os casos de uso são ferramentas essenciais para o levantamento de requisitos e para o planejamento e controle do projeto, é uma das tarefas básicas mais importante na fase de elaboração do projeto. É difícil dizer quando não usá-los.

2.2.1.1. Diagrama de Casos de Uso

São passos que descrevem uma interação entre o usuário e o sistema, são na verdade sequências de processos, com caminhos alternativos, adicionais para incluir maneiras para o usuário conseguir o que precisa, para dar certo. Quanto maior o risco no caso de uso, mais detalhes vai ter para que não haja confusão e para explicar, mostrar melhor a sequência do caso de uso, tudo na medida do necessário para a implementação do projeto.

2.2.1.2. Diagrama De Classes

O diagrama de classes embora seus elementos básicos sejam necessários, conceitos avançados do diagrama são menos utilizados. Portanto pode-se dividir em duas partes: básica e a avançada.

O diagrama de classes descreve os tipos de objetos em um sistema e os vários tipos de relacionamentos estáticos que existem entre os objetos. Há dois tipos principais de relacionamentos estáticos:

Por Exemplo:

- Associações (um para muitos).
- Subtipos (um Analista é um tipo de pessoa).

Diagramas de classes mostram atributos e operações de uma classe e suas restrições são as maneiras de como são interligados.

2.2.1.3. Perspectivas

O entendimento das perspectivas são essenciais para desenhar e ler os diagramas de classes, as linhas entre perspectivas não são rígidas, e na maioria dos modelos eles não se preocupam em ter as perspectivas classificadas para a modelagem.

2.2.1.4. Associações

Representam relações de conceitos entre classes. O diagrama indica que um Pedido só pode vir de um único Cliente e que um Cliente pode ter vários Pedidos nesse período de tempo. Cada Pedido tem várias Linhas de Pedido, e que cada uma e referente a um único Produto.

Cada associação tem duas pontas de associação; e cada uma é ligada a uma das classes na associação. Uma ponta pode ser rotulada. E este rótulo é chamado de nome de papel. (As pontas de associação são chamadas de papéis). A segunda

ponta da linha de Pedido é chamada de *linha de item*. Se não existe rótulo, você pode nomear uma ponta com o nome da classe-alvo, então, por exemplo, a ponta do Cliente da associação com Pedido poderia ser chamada de *cliente*.

Uma ponta de associação tem multiplicidade, que indica quantos objetos podem participar de um determinado relacionamento. O símbolo “*“ na ponta do Pedido de uma associação com o Cliente indica que o Cliente tem muitos Pedidos associados a ele, enquanto “1“ na outra ponta indica que um Pedido pode vir somente de um Cliente.

Dentro de uma especificação, as associações representam responsabilidades. Deve haver um modo de relacionar o Pedido ao Cliente.

As flechas que são associadas indicam a navegabilidade. A navegabilidade é uma das partes importantes dos diagramas de especificação e de implementação. Mas não tem um propósito útil nos diagramas conceituais.

Se existir navegabilidade em uma só direção, ela é chamada de associação unidirecional. E uma associação bidirecional contém navegabilidades nos dois sentidos. Em UML se usa associações sem flechas para indicar que a navegabilidade é bidirecional ou desconhecida. O projeto deve estabelecer um dos dois significados.

Associações bidirecionais incluem uma restrição, as duas navegações são inversas uma da outra. Há muitas maneiras para se nomear associações. Modeladores de dados gostam de nomear associações usando expressões verbais, de modo que o relacionamento possa ser usado na frase.

Uma Associação representa uma união entre dois objetos. Essa união existe durante toda a vida dos objetos. Portanto, uma referência a parâmetro, ou criação de objeto, não implica em associações, e você pode modelar como dependência.

2.2.1.5. Atributos

São muito parecidos com as associações. Especificando, o atributo indica que um objeto pode dizer o seu nome e tem várias maneiras de atribuir um valor a nome. Na implementação, o objeto tem um campo (também chamada de variável de instância) para armazenar seu nome.

Então qual é a diferença entre atributo e associação?

Não há diferença na perspectiva de conceito. Um atributo é somente mais um tipo de notação que pode ser usada. Geralmente, os atributos têm valor único. Um diagrama não indica se um atributo é obrigatório ou opcional, embora possa fazê-lo colocando multiplicidade depois do nome do atributo em colchetes quadrados. Exemplo, data-Recebimento [0..1]: Date;.

A diferença aparece nos níveis de especificação e de implementação. Atributo indica navegabilidade somente do tipo para um atributo e indica que o tipo tem sua própria cópia do objeto atributo, implicando que qualquer tipo usado como um atributo tem seu valor em vez de referência.

2.2.1.6. Operações

São processos que a classe pode realizar. Operações correspondem a métodos em uma classe. Normalmente, não se mostra as operações que manipulam atributos, porque podem inferidas. No entanto, poderá ter que identificar se um dado atributo é somente para leitura ou congelado (o seu valor nunca muda). No modelo de implementação, também pode querer mostrar operações privativas (private) e protegidas (protected).

Diferença entre operação e método. Uma operação é algo que é executado em um objetivo (procedimento de chamada), enquanto um método é o corpo do procedimento. Os dois são diferentes quando existe polimorfismo. Quando se tem um supertipo com três subtipos, cada um deles sobrescreve a operação do supertipo, com isso se tem uma operação e quatro métodos que a implementam.

As pessoas normalmente usam operações e métodos como sinônimos, mas em certas ocasiões em que é necessário ser preciso sobre a diferença.

2.2.2. Diagramas de Atividades

Ao contrário da maioria das outras técnicas da UML, as origens dos diagramas de atividades não são claras. Ao contrário, o diagrama de atividades combina ideias de varias outras técnicas: os diagramas de eventos de Jim Odell, técnicas de modelagem de estado SDL, modelagem de *workflow* e redes de Petri.

Estes diagramas são uteis em relação com *workflow* e na descrição de comportamento que tem muitos processamentos em paralelo.

Uma atividade é um estado de estar fazendo alguma tarefa, podendo ser um processo real, tal como escrever uma requisição, ou uma execução de *software, um* método em uma classe.

O diagrama de atividades é uma sequencia de atividades (passo-a-passo), com suporte condicional e paralelo de comportamento. Um Diagrama de atividades é uma variante de um diagrama de estados na maioria das vezes, se não todas, dos estados e estado de atividade. Portanto, sua terminologia segue a mesma terminologia de diagrama de estados.

Um estado de atividade pode ter múltiplas transições de saída e múltiplas transições de entrada. O losango deixa claros os desvios e as intercalações em um diagrama.

Comportamento paralelo é indicado por Separações que são chamadas de (Forks) e as Junções (Joins).

Separação (Fork) tem uma transição de entrada e varias transições de saída. Quando a transição de entrada é acionada (*triggered*), todas as transições de saída são executadas em paralelo. O diagrama de atividades permite que fazer a escolha e ordem que as tarefas serão executadas. De outro modo, simplesmente determina as regras principais de uma sequencia que se deve seguir. Esta é a diferença entre um diagrama de atividades e um fluxograma: os fluxogramas normalmente são

limitados por processos sequenciais, enquanto os diagramas de atividades lidam com processos paralelos.

Isso é importante para modelagem de negócios. Negócios frequentemente, tem processos não necessariamente sequenciais. Uma técnica como essa encoraja comportamento paralelo e de como é importante nestas situações para que as pessoas se afastem de sequências desnecessárias, nos seus comportamentos e a ajuda identificar oportunidades para fazer coisas em paralelo.

Isso pode melhorar a eficiência e o retorno de processos de negócio.

2.2.2.1. Decompondo uma Atividade

Uma atividade pode ser dividida em subatividades, funcionando como as funções de superestados e subestados em um diagrama de estados. Podendo somente mostrar os superestados no diagrama-pai, ou podendo mostrar superestado no seu comportamento externo dentro dele. Podemos também projetar transições diretamente para dentro, ou para fora do diagrama subsidiário. Uma atividade de entrega pode ser usada em vários outros contextos, e o diagrama-pai é desligado dos conteúdos do diagrama subsidiário, isso é uma vantagen dos diagramas de estados de fim e inicio.

Os diagramas de atividades podem dizem o que acontece em um projeto, mas não dizem quem faz e o quê faz. Na programação, significa que o diagrama não representa qual a classe que é responsável por cada atividade.

Em modelagem de domínio, significa que o diagrama não representa quais as pessoas e departamentos são responsáveis por cada atividade executada. Uma solução, para isso, seria rotular cada uma das atividades com a classe ou pessoa responsável.

Raias pode ser uma solução para esse tipo de problema. Para usar raias, você deve organizar os diagramas de atividades em zonas verticais separadas por linhas. Cada zona tem suas responsabilidades representadas por uma classe ou um departamento específico.

As Raias são úteis porque combina a descrição de lógica do diagrama de atividades junto com a descrição de responsabilidades do diagrama de interação. Mas podendo ser difíceis de serem projetadas em um diagrama complexo.

2.2.2.2. Quando Utilizar Diagramas de Atividades

Como na maioria das técnicas, os diagramas de atividades têm suas qualidades e alguns pontos fracos definidos, com isso a melhor maneira de usar o diagrama é em combinação com outras técnicas de modelagem.

A maior qualidade dos diagramas de atividades esta no fato de que ele suporta comportamento paralelo. Isso o tona uma grande ferramenta de *workflow*. E a desvantagem desses diagramas são que eles não deixam claras as ligações entre as ações e os objetos.

2.2.3. Diagrama de Sequência

Diagrama de sequência faz parte de um modelo de interação, que são modelos que descrevem como grupos de objetos ajudam em algum comportamento.

Normalmente um diagrama de interação pega o comportamento de um único caso de uso. Em um diagrama de sequência, um objeto é mostrado com uma caixa na parte superior de uma linha tracejada vertical, essa linha vertical é chamada de linha de vida do objeto. A Linha de vida representa a vida de um objeto durante essa interação. Cada mensagem é representada por uma seta entre as linhas de vida de dois objetos. A ordem na qual estas mensagens ocorrem é mostrada da parte superior à parte inferior da página. Cada mensagem é rotulada, com no mínimo, um nome na mensagem, também incluindo argumentos e informações de controle, podendo ainda mostrar uma autochamada, que é uma mensagem que um objeto

manda para si próprio, enviando a seta de mensagem de volta para a mesma linha de vida.

Para isso duas informações de controle são essenciais.

Primeiro a condição, que indica quando uma mensagem for enviada. A mensagem só é enviada se a condição for verdadeira. As condições são úteis para casos simples, mas para casos mais complicados, a melhor opção é projetar diagramas de sequência separados para cada caso de uso.

Segundo, o marcador de controle útil é o de iteração, que mostra que a mensagem é enviada várias vezes para muitos objetos receptores. Podendo mostrar a base da iteração entre colchetes.

O fluxo global de controle é uma das coisas mais difíceis de compreender em um programa orientado a objetos. Um projeto bom tem muitos pequenos métodos em classes diferentes, e podendo ás vezes ser mais difícil de entender a sequência global do comportamento. Os diagramas de sequência ajudam a entender estas sequências mais complexas.

2.2.4. Diagrama de Entidade Relacionamento

O Modelo Entidade-Relacionamento (E-R ou Entidade Associação) é usado na maioria para representação de métodos e ferramentas de auxilio. Um modelo de dados consiste em um conjunto de ferramentas conceituais que são usadas para representar graficamente a estrutura de um banco de dados de uma aplicação.

O Diagrama Entidade-Relacionamento (DER) é um dos mais utilizados para modelar Bancos de Dados Relacionais no conceito de banco de dados. Esse modelo é constituído por um grupo básico de objetos que são chamados de entidade e por relacionamentos entre objetos. O modelo representa graficamente o conceito do banco de dados por meio do Modelo (E-R). A relativa simplicidade e a clareza gráfica desta técnica, explica como é o uso deste diagrama.

Esse modelo foi proposto por Peter Chen em 1976 para facilitar o projeto e entendimento de bancos de dados.

- Entidades: Objetos.
- Associações: Ligações entre objetos.
- Atributos: Propriedades dos objetos.

Entidade é objeto (concreto ou abstrato) para representar o que queremos que tenha existência própria e que pode ser distintamente identificada, são representações de uma classe com as mesmas características.

Associações são as que fazem ligações entre entidades onde cada uma ocupa um tipo de indicação.

Atributos são as propriedades de um objeto que são associadas a um tipo de associação ou tipo de entidade.

3. TECNOLOGIAS UTILIZADAS

3.1. JAVA

Java é a uma base de desenvolvimento para todos os tipos de aplicações em rede, e é o padrão global de implementação e distribuição de aplicações, jogos, dispositivos móveis, conteúdo baseado em Web, softwares corporativos. Com milhares de pessoas que desenvolvem no mundo todo, e de forma eficiente, o Java possibilita que você desenvolva, implante, e use aplicações e serviços estimulantes.

Desde consoles de games a supercomputadores científicos, laptops, datacenters, telefones celulares, Internet, o Java está por todos os lugares! Muitos acham que Java é uma maneira de deixar as páginas da web mais bonitas, cheia de efeitos ou para fazer pequenos formulários na web. O Java tenta diminuir alguns problemas. Hoje em dia o foco do Java não é só aplicações web, applets. Apesar de ter sido construído com um propósito e lançada com outro, o Java ganhou destaque no lado de servidores.

3.2. NETBEANS

Ambiente de Desenvolvimento Integrado de código-fonte aberto gratuito para desenvolvedores de software. Todas as ferramentas necessárias para criar aplicações desktop profissionais, corporativas, Web e móveis com a plataforma Java, bem como C/C++, PHP, JavaScript, e Groovy.

O Netbeans IDE 7.4 oferece um melhor desempenho e experiência de codificação, com novos recursos de análise de código estático no Editor de Java e varredura do projeto mais inteligente. Outras funcionalidades incluem integração com o JavaFX

Scene Builder; suporte para vários frameworks PHP; suporte Groovy atualizado; e aprimoramentos para Java EE, Maven, C/C++ e a Plataforma Netbeans.

3.3. JAVA WEB

Aprenda a criar vários tipos aplicações completas, independentes e com tecnologias importantes para um desenvolvimento. Exemplos de aplicações: Servlets, JSP, Spring e Hibernate**,** além de técnicas de Design como Patterns, Ajax e MVC. O mercado WEB é o mais forte em que o Java esta presente e com mais oportunidades. O desenvolvimento em Java não é simples: é necessário conhecer com certa profundidade as APIs de servlets e de JSP, mesmo que você utilize frameworks como Struts, VRaptor ou JSF. Conceitos de HTTP, session e cookies também são necessários para poder entender e resolver os problemas que a aplicação enfrentará.

3.4. MYSQL

Falando em banco de dados, o MySQL é o mais popular banco de dados SQL livre, fornecido pela MySQL AB. MySQL AB é uma empresa comercial e seu negócio é fornecer serviços relacionados ao banco de dados MySQL. O MySQL é um sistema que faz gerenciamento de bancos de dados.

Um banco de dados é uma coleção de dados estruturados. Ele pode fornecer qualquer coisa, uma simples lista de compras, uma galeria de imagens ou até grandes quantidades de informações de uma rede coorporativa. Para que se possa

adicionar, acessar, e processar dados que foram armazenados no banco de dados digital, você necessita de um sistema gerenciador de bancos de dados semelhante ao MySQL.

Como os computadores são muito bons em lidar com grandes quantidades de dados, o gerenciamento de bancos de dados funciona como a engrenagem central na computação, como utilitários independentes, ou como partes de outras aplicações. O MySQL é um sistema gerenciador de bancos de dados relacional.

Bando de dados relacional armazena dados em tabelas separadas embora todos os dados estejam armazenados em um só local. Isso proporciona velocidade e flexibilidade. As tabelas são unidas por relações definidas tornando possível combinar dados de diferentes tabelas nas requisições. A parte SQL do MySQL atende pela "Linguagem estruturada de pesquisas" - a linguagem padrão mais comum usada para acessar bancos de dados.

O MySQL é um software Open Source. Open Source garante para qualquer pessoa o uso ou modificação do software. Qualquer pessoa pode fazer download do MySQL pela Internet e usá-lo. Qualquer pessoa dedicada pode estudar o código fonte e alterá-lo para adequá-lo as suas necessidades. O MySQL usa a GPL (Licença Pública Geral GNU) http://www.gnu.org, para definir o que você pode e não pode fazer com o software em diferentes situações. Se sentir desconforto com a GPL ou precisa embutir o MySQL numa aplicação comercial você pode adquirir a versão comercial.

Por que usar o MySQL?

O MySQL é extremamente rápido, confiável, e fácil de usar. O MySQL também tem um conjunto de recursos muito práticos desenvolvidos. MySQL foi desenvolvido para armazenar grandes informações de dados e de maneira muito mais rápida que outras soluções existentes e está sendo usado em ambientes de produção que exige alta demanda, e por diversos anos de maneira bem sucedida. Apesar de estar sempre em desenvolvimento, o MySQL hoje possibilita muitos recursos e conjuntos de funções. A conectividade, rapidez, e a segurança faz com que o MySQL seja altamente adaptável para acessar bancos de dados na Web.

3.5. APACHE TOMCAT

O software Tomcat, desenvolvido pela Fundação Apache, permite a execução de aplicações para web. A sua principal característica é em estar concentrada na linguagem de programação Java, mais especifico nas tecnologias de Servlets e Java Server Pages (JSP).

A Fundação Apache, conhecida pelo seu servidor web, permite no caso do servidor Apache, que o Tomcat seja usado livremente, para fins comerciais ou não.

O Tomcat está implementado em Java e, por isso, necessita que a versão Java 2 Standard Edition (J2SE) esteja instalada no computador de onde ele será executado. No entanto, não basta ter a versão runtime de Java instalada, pois o Tomcat necessita compilar (e não apenas executar) programas escritos em Java.

3.6. PRIMEFACES

Hoje em dia existem várias bibliotecas JSF disponíveis para aplicações Java EE (RichFaces, IceFaces, etc). Não da para falar qual biblioteca é a melhor. Cada fornecedor implementa à sua forma a especificação JSF e adiciona seus componentes de forma independentes. Existem pontos positivos e negativos em cada implementação.

O PrimeFaces é um pouco mais interessante que os demais fornecedores. Basta incluir em sua aplicação o jar primefaces-X.jar e usar os componentes. Nenhuma configuração é necessária para utilizar os componentes Prime. Isso já é um grande começo.

Outras vantagens são:

- Mais de 100 componentes ricos para sua aplicação;
- Componentes adicionais para desenvolvimento móvel;
- Total integração com JQuery;
- Documentação muito simples de utilizar com vários exemplos copy-and-paste (Copiar e Colar).

4. ANÁLISE DO PROJETO

4.1. ORÇAMENTO

Orçamento do Projeto = Estimativa de custos para as atividades + Estimativa de custos para os recursos.

- ✓ **Pessoal**

Analista	**Quantidade Horas** Horas relativas ao trabalho do analista.	**Custo/hora (R$)**	**Total (R$)**
Contratado	60	35,00	2.100,00
Custo Analista			**2.100,00**

Programadores	**Quantidade Horas** Horas relativas ao trabalho dos Programadores.	**Custo/hora (R$)**	**Total (R$)**
Maicon	80	25,00	2.000,00
Total Custo			**2.000,00**

Tabela 1 - Custo Pessoal

- ✓ **Equipamentos**

01 notebooks

Valor unitário = R$1.800,00

Dias (de uso) = 26 dias

Depreciação = R$ 1,800,00 / 24 meses

30 dias -> 75,00

26 dias -> x

X = R$ 65,00

Custo nos 26 dias = R$ 65,00.

Custo do Notebook = R$ 65,00

- ✓ **Software**

Netbeans = R$0,00

MySQL = R$0,00

Total = R$0,00

CUSTO TOTAL DO PROJETO = R$ 2.000,00 + R$65,00 + R$0,00 = **R$ 2065,00**

4.2. DIAGRAMAS DE CASOS DE USO

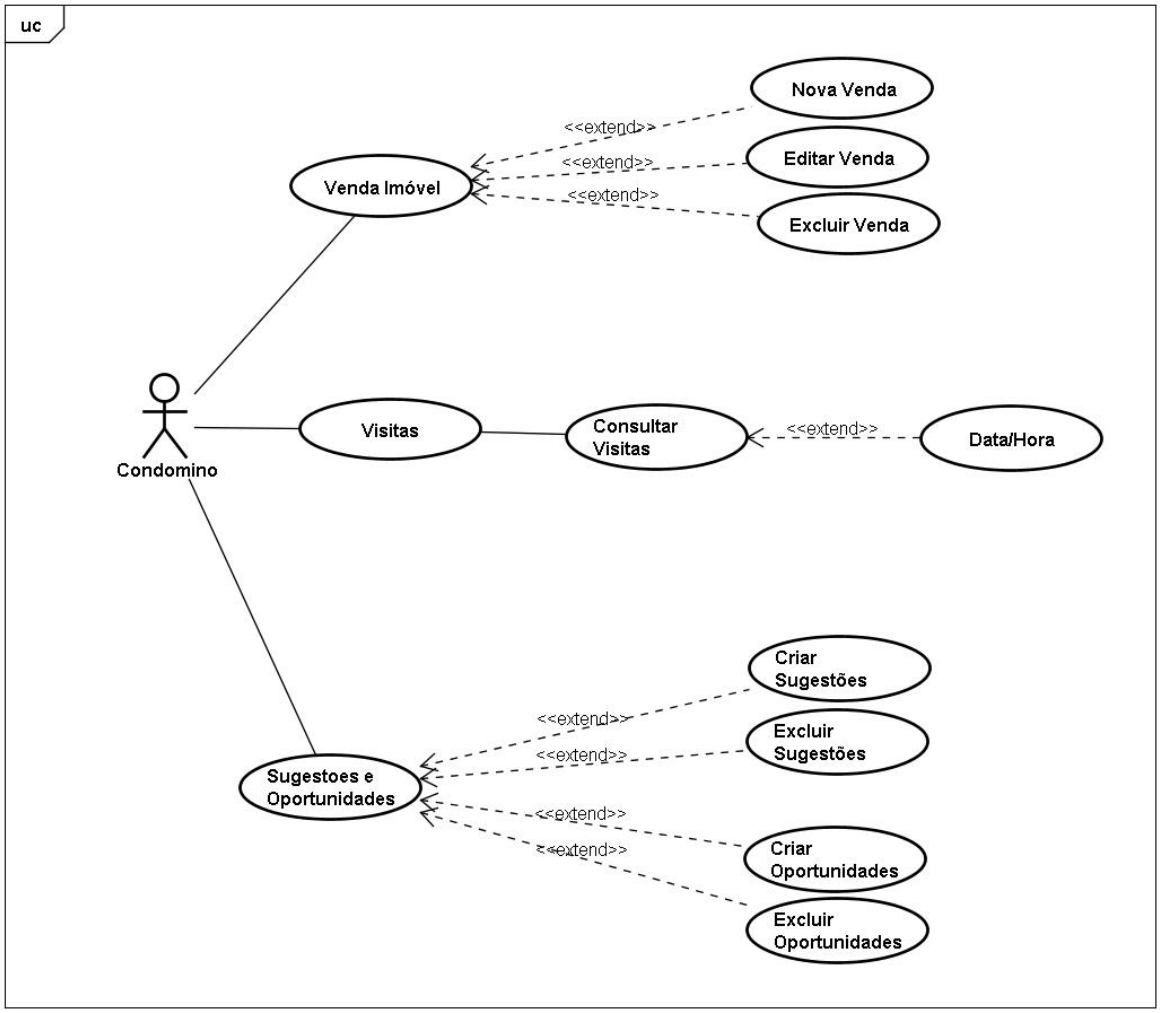

Figura 1 - Diagrama de Caso de Uso Geral - Condômino

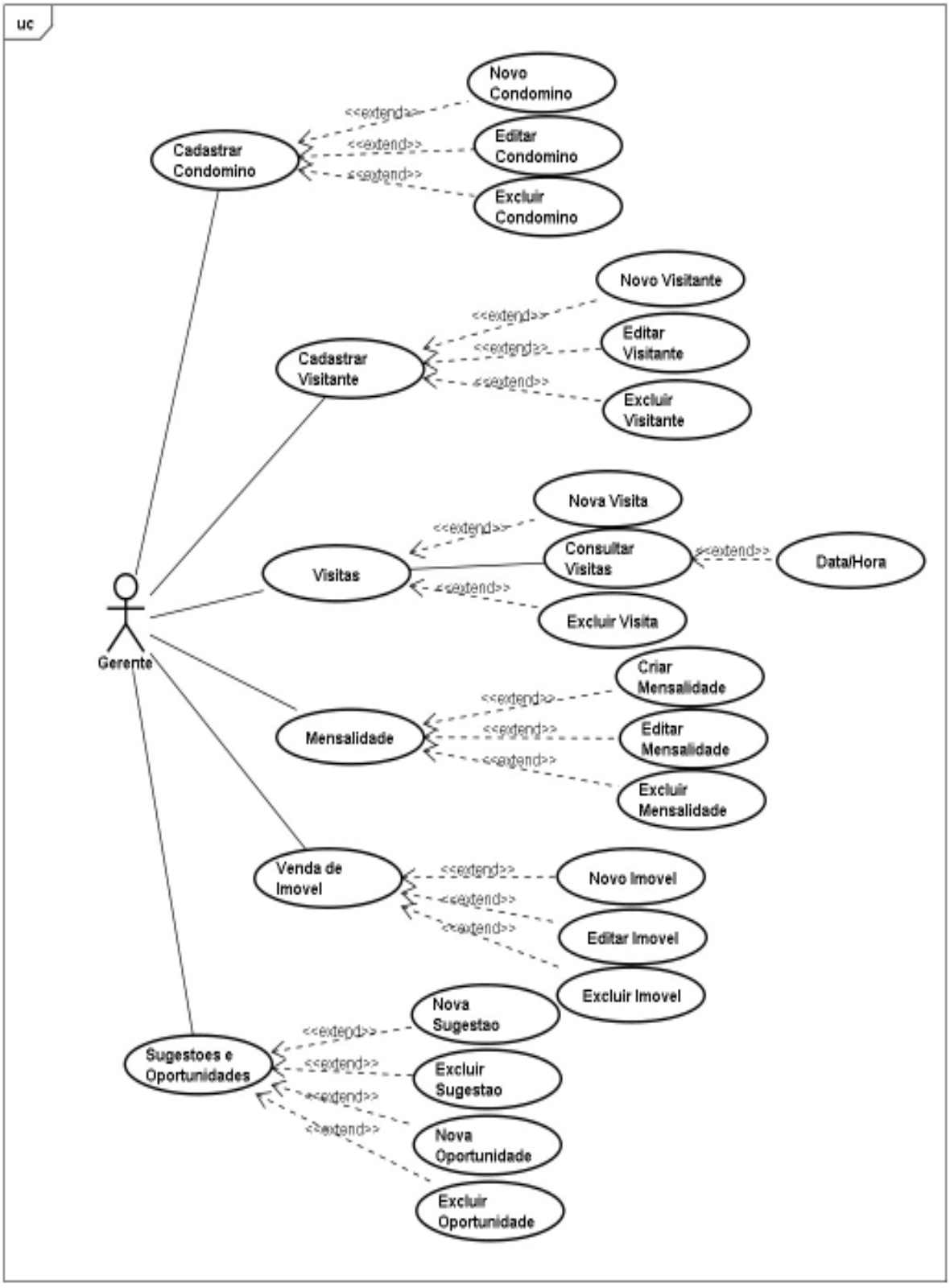

Figura 2 - Diagrama de Caso de Uso Geral – Gerente

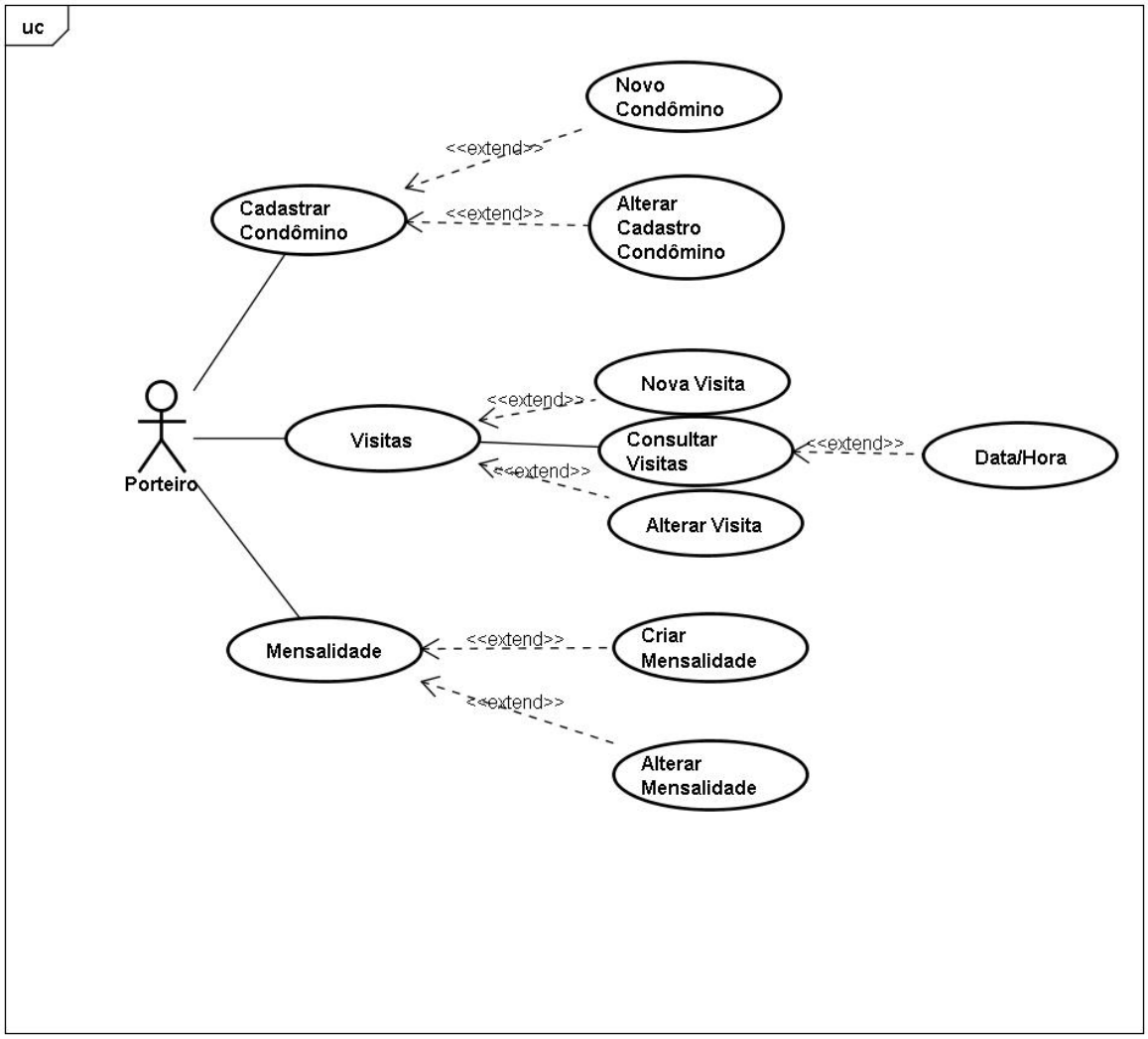

Figura 3 - Diagrama de Caso de Uso Geral – Porteiro

4.2.1. Especificações de Caso de Uso

4.2.1.1. Caso de Uso - Venda do Imóvel

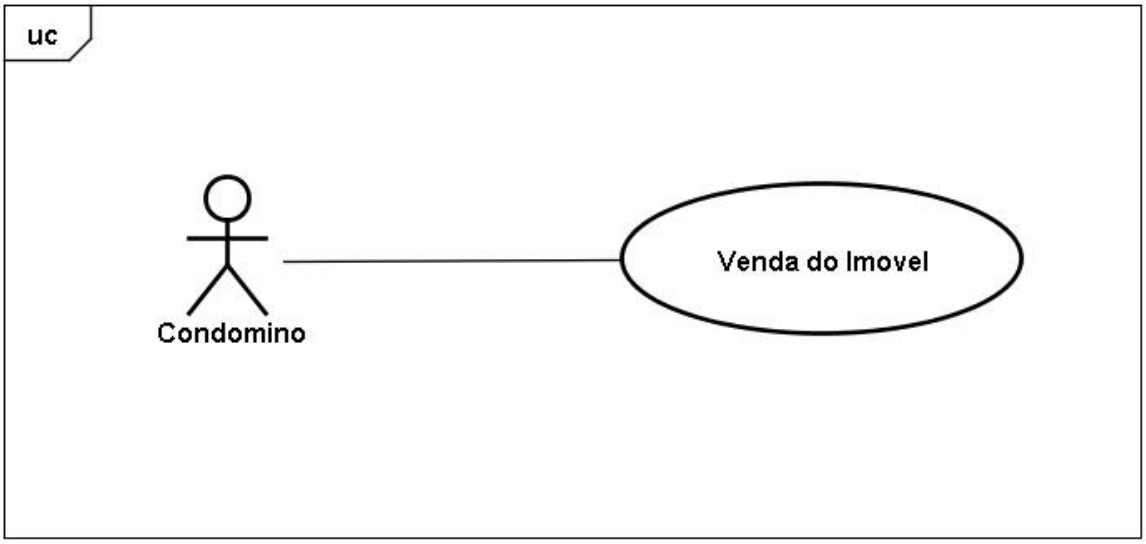

Figura 4 - Diagrama de Caso de Uso - Venda do Imóvel

Finalidade:	Venda do Imóvel
Ator:	Condômino
Pré-Condição:	O Condômino deve efetuar Login na aplicação.
Evento Inicial:	O Condômino seleciona a opção Venda Imóvel.
Fluxo Principal:	5A - A aplicação solicita o os dados do condômino e do imóvel; 5B - O Condômino informa os dados e confirma; (A1) 5C - A aplicação verifica se os dados e campos estão corretos; (E1) 5D - O Condômino confirma os dados; A2 5E - O Condômino envia a solicitação de Venda do Imóvel e confirma o envio; 5F - A aplicação solicita a confirmação Venda e o envio; (A3) 5G - O UC é encerrado;
Fluxo Alternativo:	A1 – Cancelar Operação: A) O Condômino cancela a operação de Venda do Imóvel; B) A aplicação retorna ao passo 5A;

	A2 – Venda do Imóvel: O Condômino solicita Venda do Imóvel;
Fluxo de Exceção:	E1- Dados incorretos: A) A aplicação verifica os dados do condômino e da Venda do Imóvel; B) A aplicação encerra o UC;

Tabela 2 - Descrição de Caso de Uso - Venda do Imóvel

4.2.1.2. Caso de Uso - Sugestões ou Oportunidades

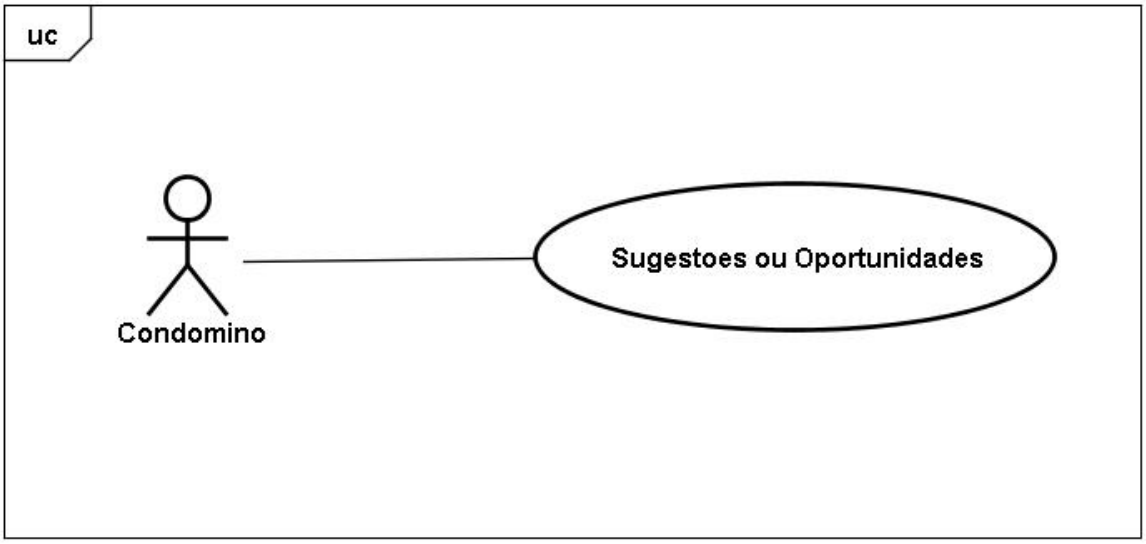

Figura 5 - Diagrama de Caso de Uso - Sugestões ou Oportunidades

Finalidade:	Sugestões ou Oportunidades
Ator:	Condômino
Pré-Condição:	O Condômino deve efetuar Login na aplicação.
Evento Inicial:	O Condômino seleciona a opção Sugestões e Oportunidades.
Fluxo Principal:	5A - A aplicação solicita o os dados do condômino e a opinião; 5B) O Condômino informa os dados e sua opinião e confirma; (A1) 5C - A aplicação verifica se os dados estão corretos; (E1) 5D - O Condômino confirma os dados;

	A2 5E - O Condômino envia a solicitação de Sugestões ou Oportunidades e confirma o envio; 5F - A aplicação solicita a confirmação de Sugestões e Oportunidades; (A3) 5G - O UC é encerrado;
Fluxo Alternativo:	A1 – Cancelar Operação: A) O Condômino cancela a operação de Sugestões ou Oportunidades; B) A aplicação retorna ao passo 5A; A2 – Sugestões ou Oportunidades: O Condômino faz a solicitação de Sugestões e Oportunidades I;
Fluxo de Exceção:	E1- Dados incorretos: A) A aplicação verifica os dados do condômino e da Venda do Imóvel; B) A aplicação encerra o UC;

Tabela 3 - Descrição de Caso de Uso - Sugestões ou Oportunidades

4.2.1.3. Caso de Uso - Cadastrar Condômino

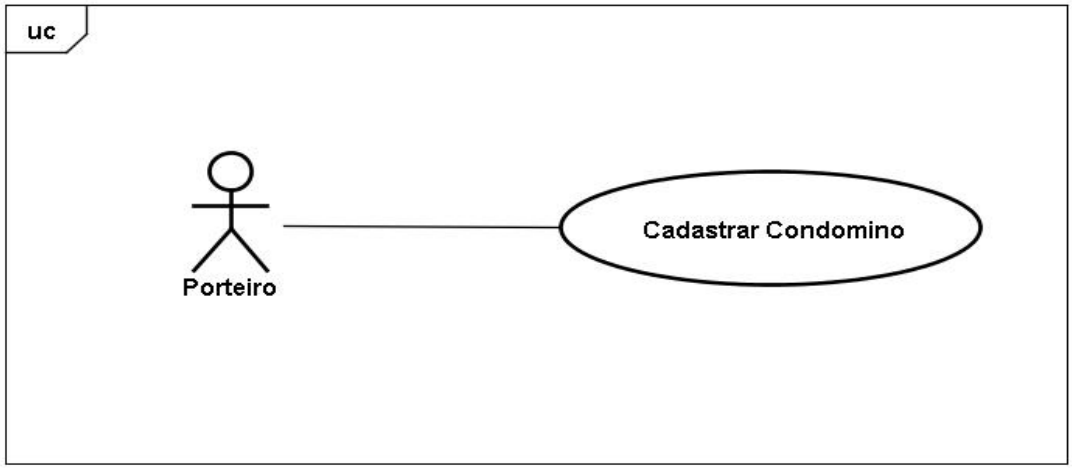

Figura 6 - Diagrama de Caso de Uso - Cadastrar Condômino

Finalidade:	Cadastrar Condômino
Ator:	Porteiro
Pré-Condição:	O porteiro deve efetuar Login no sistema.
Evento Inicial:	O porteiro seleciona a opção cadastrar condômino.
Fluxo Principal:	5A - O Sistema solicita o os dados do condômino; 5B - O porteiro informa os dados e confirma; (A1) 5C - O Sistema verifica se os dados e campos estão corretos; (E1) 5D - O porteiro confirma o cadastro; (A2) 5E - O porteiro envia o cadastro do condômino e confirma o envio; 5F - O Sistema solicita a confirmação do cadastro e o envio; (A3) 5G - O UC é encerrado;
Fluxo Alternativo:	A1 – Cancelar Operação: A) O porteiro cancela a operação de cadastro; B) O sistema retorna ao passo 5 A; A2 – Cadastrar Condômino: O porteiro cadastra um novo condômino; A3 – Condômino não cadastrado: A) O porteiro não confirma o cadastro; B) Retorna ao passo 5 D;
Fluxo de Exceção:	E1- Dados incorretos: A) O sistema verifica os dados do condômino; B) O sistema encerra o UC;

Tabela 4 - Descrição de Caso de Uso - Cadastrar Condômino

4.2.1.4. Caso de Uso - Cadastrar Visitantes

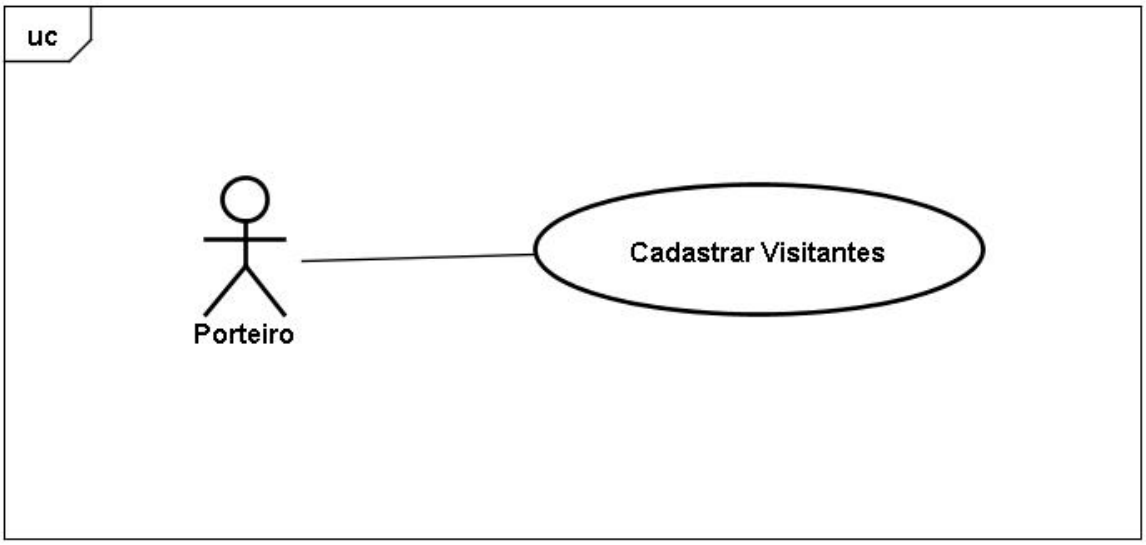

Figura 7 - Diagrama de Caso de Uso - Cadastrar Visitantes

Finalidade:	Cadastrar Visitantes
Ator:	Porteiro
Pré-Condição:	O porteiro deve efetuar Login na aplicação.
Evento Inicial:	O porteiro seleciona a opção cadastrar visitantes.
Fluxo Principal:	5A - A aplicação solicita o os dados do visitante; 5B - O porteiro informa os dados e confirma; (A1) 5C - A aplicação verifica se os dados e campos estão corretos; (E1) 5D - O porteiro confirma o cadastro; A2 5E - O porteiro envia o cadastro do visitante e confirma o envio; 5F - A aplicação solicita a confirmação do cadastro e o envio; (A3) 5G - O UC é encerrado;
Fluxo Alternativo:	A1 - Cancelar Operação: A) O porteiro cancela a operação de cadastro; B) A aplicação retorna ao passo 5A; A2 - Cadastrar Visitante: O porteiro cadastra um novo visitante; A3 - Visitante não cadastrado:

	A) O porteiro não confirma o cadastro; B) Retorna ao passo 5D;
Fluxo de Exceção:	E1 - Dados incorretos: A) A aplicação verifica os dados do visitante; B) A aplicação encerra o UC;

Tabela 5 - Descrição de Caso de Uso - Cadastrar Visitantes

4.2.1.5. Caso de Uso - Nova Visita

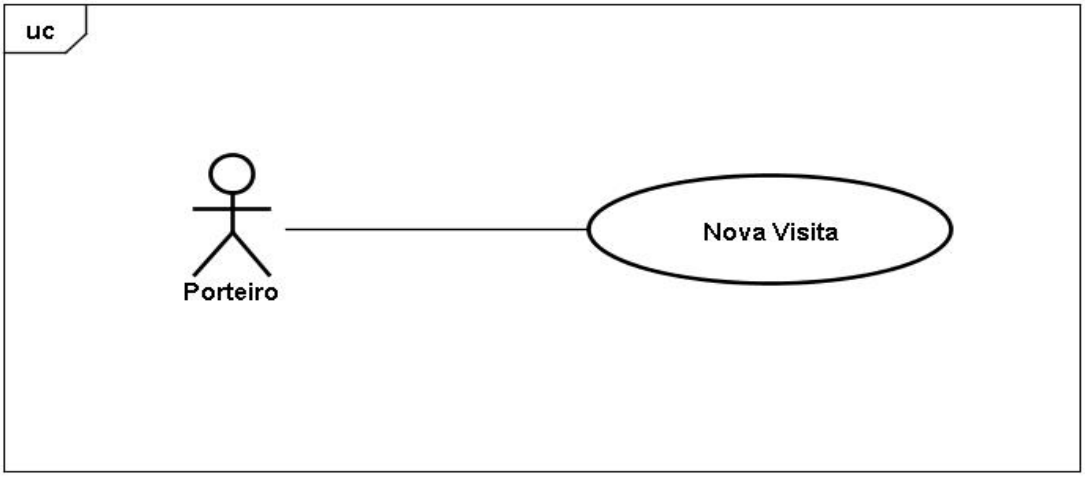

Figura 8 - Diagrama de Caso de Uso - Nova Visita

Finalidade:	Nova Visita
Ator:	Porteiro
Pré-Condição:	O porteiro deve efetuar Login na aplicação.
Evento Inicial:	O porteiro seleciona a opção Nova Visitas.
Fluxo Principal:	5A - A Aplicação solicita o os dados do visitante e quem ele vai visitar; 5B - O porteiro informa os dados e confirma; (A1) 5C - A Aplicação verifica se o visitante tem cadastro; (E1) 5D - O porteiro confirma a visita; A2 5E - O porteiro confirma os dados do visitante e confirma a visita; 5F - A Aplicação solicita a confirmação da visita e envia; (A3)

	5G - O UC é encerrado;
Fluxo Alternativo:	A1 – Cancelar Operação: A) O porteiro cancela a operação de visita; B) A Aplicação retorna ao passo 5A; A2 – Nova Visitas: O porteiro cadastra uma nova visita; A3 – Visitante não cadastrado: A) O porteiro não confirma o cadastro; B) Retorna ao passo 5D;
Fluxo de Exceção:	E1- Dados incorretos: A) A aplicação verifica os dados do visitante; B) A aplicação encerra o UC;

Tabela 6 - Descrição de Caso de Uso - Nova Visita

4.2.1.6. Caso de Uso - Mensalidade

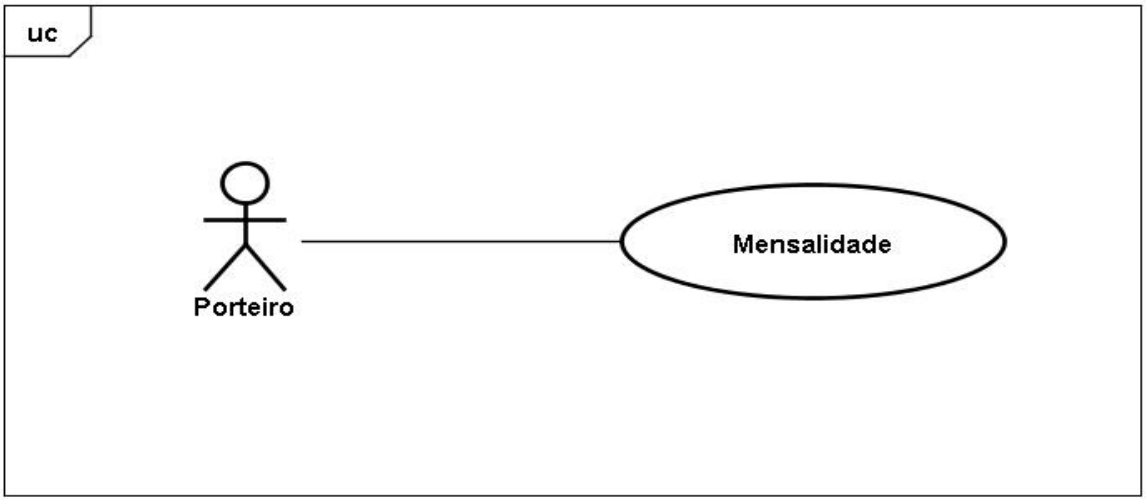

Figura 9 - Diagrama de Caso de Uso – Mensalidade

Finalidade:	Mensalidade
Ator:	Porteiro
Pré-Condição:	O porteiro deve efetuar Login na aplicação.
Evento Inicial:	O porteiro seleciona a opção mensalidade
Fluxo Principal:	5A - A aplicação solicita o os dados do condômino; 5B - O porteiro informa os dados e confirma; (A1) 5C - A aplicação verifica se os dados e campos estão corretos; (E1) 5D - O porteiro confirma os dados do condômino; A2 5E - O porteiro envia a solicitação de mensalidade e confirma o envio; 5F - A aplicação solicita a confirmação da mensalidade e o envio; (A3) 5G - O UC é encerrado;
Fluxo Alternativo:	A1 – Cancelar Operação: A) O porteiro cancela a operação de mensalidade; B) A aplicação retorna ao passo 5 A; A2 – Mensalidade: O porteiro solicita mensalidade do condômino; A3 – Condômino não encontrado: A) O porteiro não confirma a solicitação; B) Retorna ao passo 5 D;
Fluxo de Exceção:	E1 - Dados incorretos: A) A aplicação verifica os dados do condômino; B) A aplicação encerra o UC;

Tabela 7 - Descrição de Caso de Uso – Mensalidade

4.3. DIAGRAMA DE ATIVIDADES

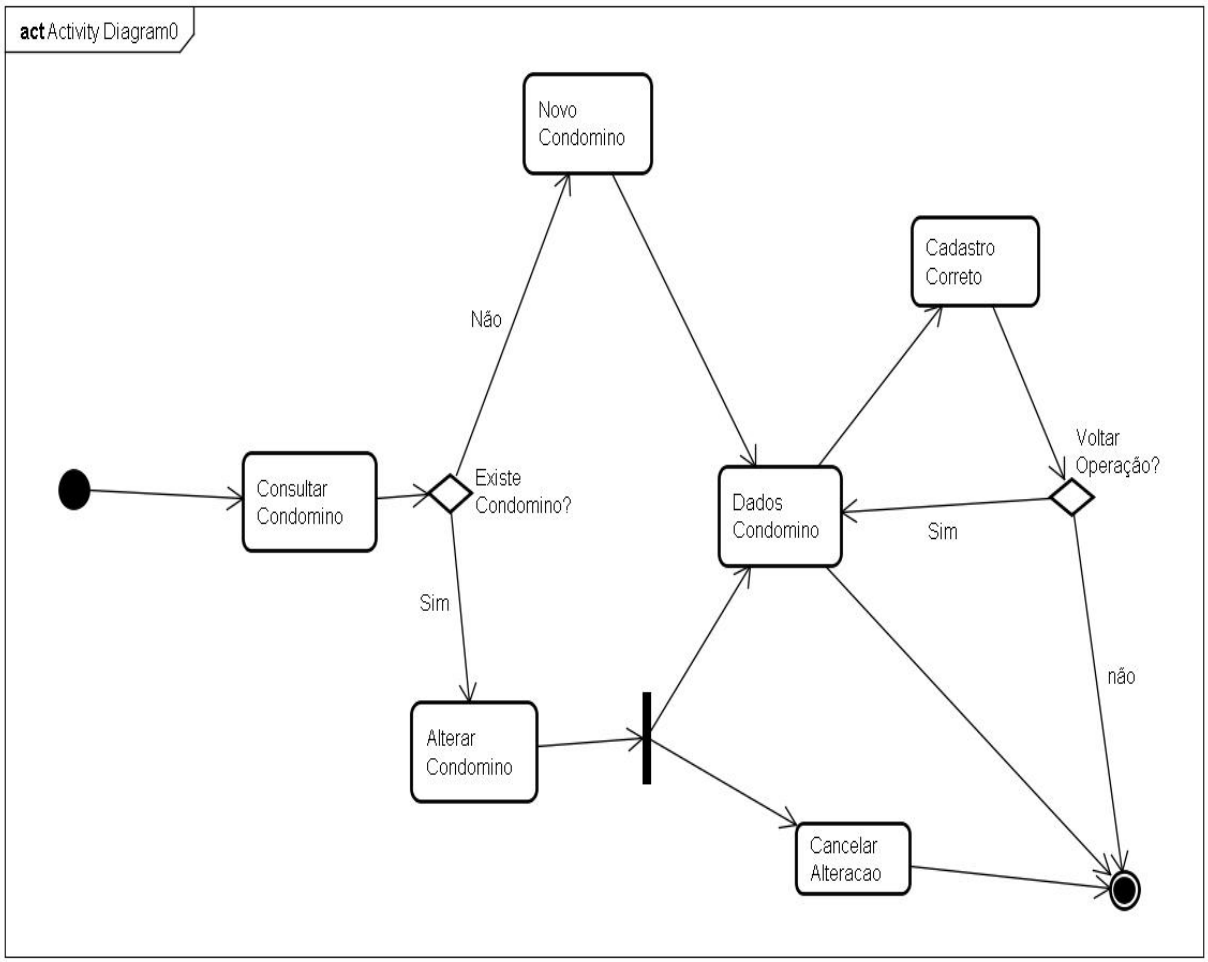

Figura 100 - Diagrama de Atividades - Porteiro

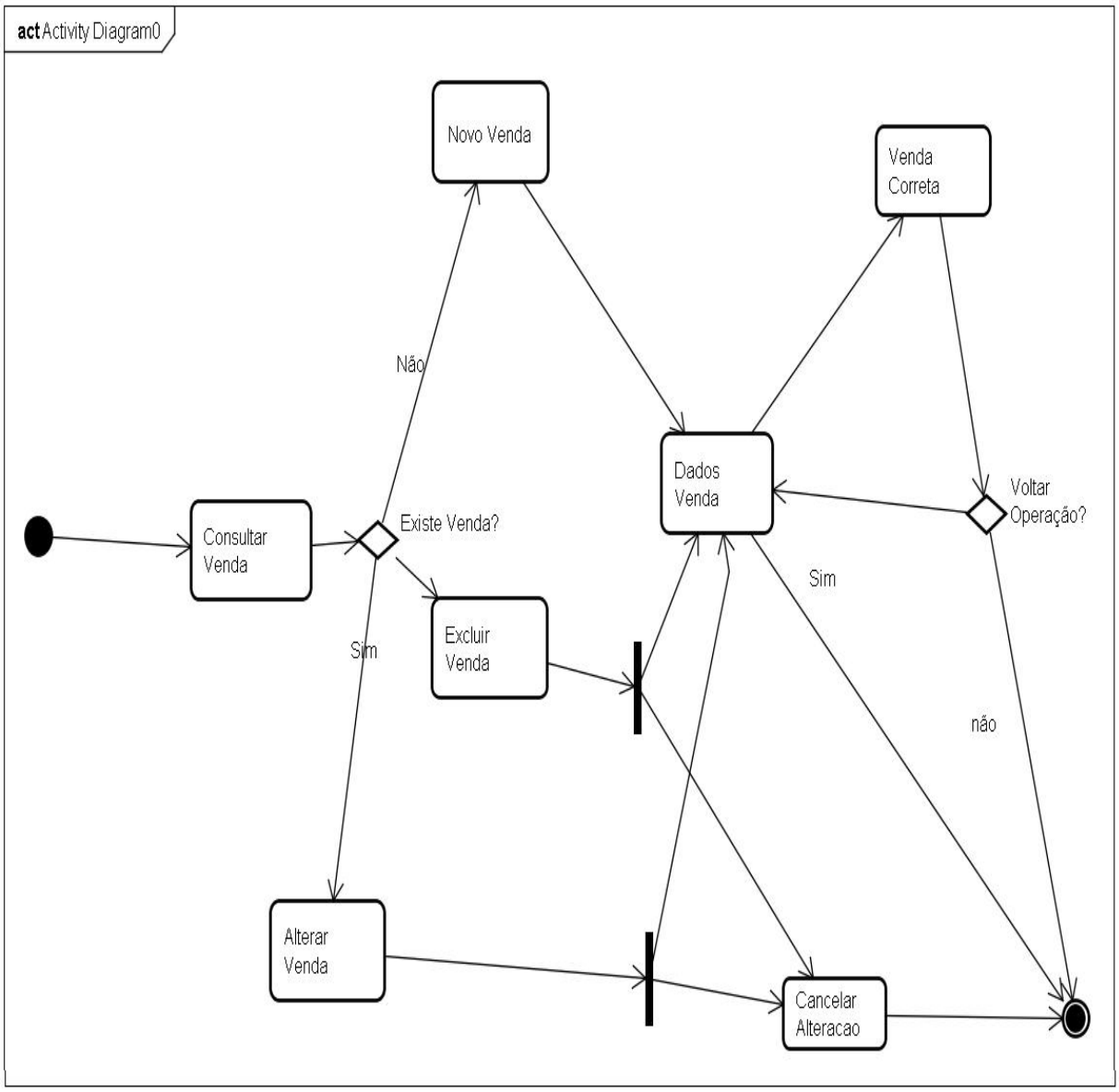

Figura 11 - Diagrama de Atividades - Condômino

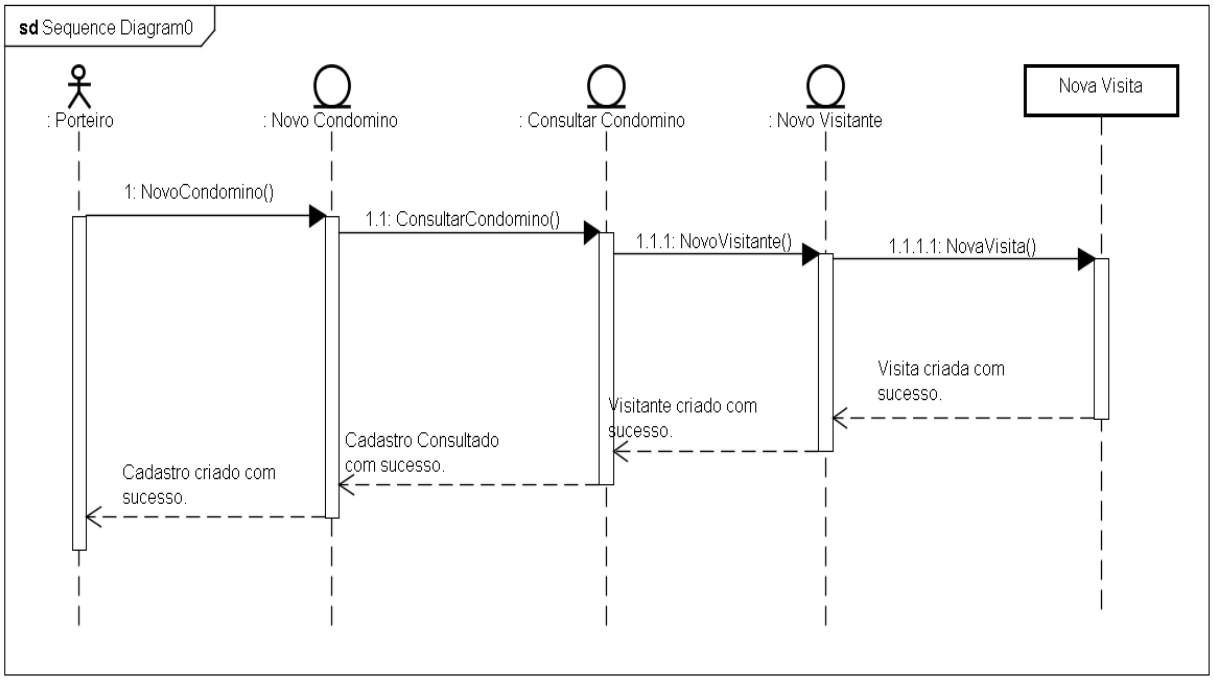

Figura 112 - Diagrama de Sequência - Porteiro

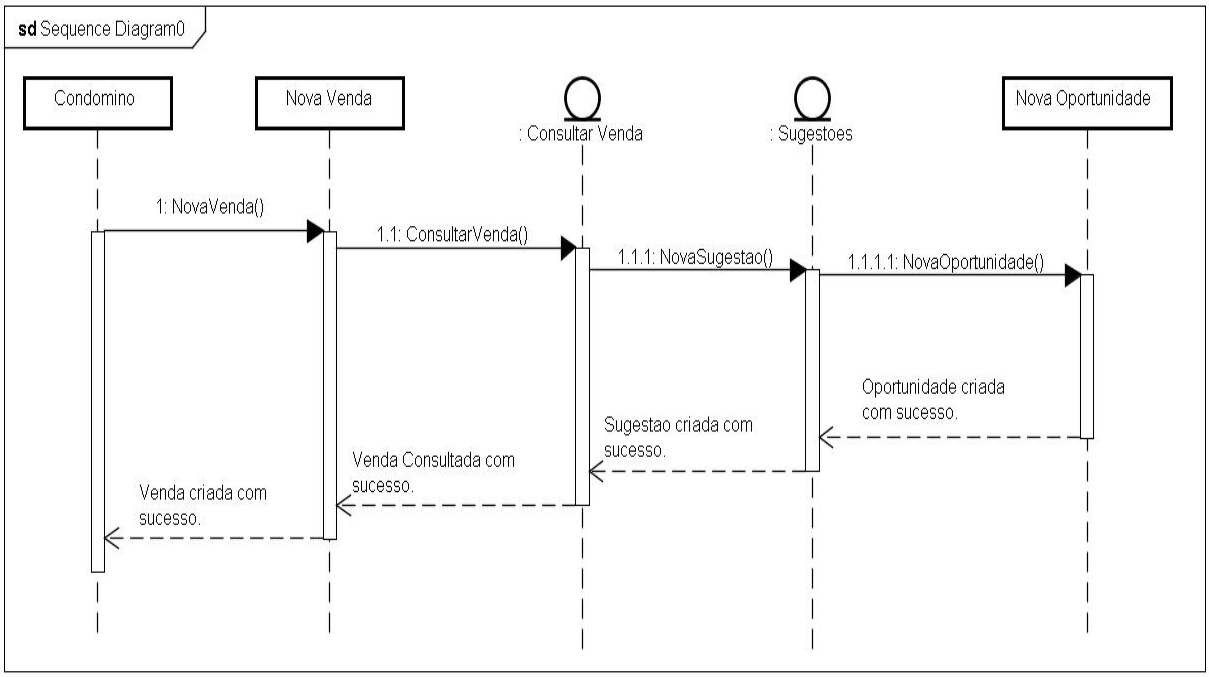

Figura 13 - Diagrama de Sequência - Condômino

4.4. DIAGRAMA DE ENTIDADE-RELACIONAMENTO

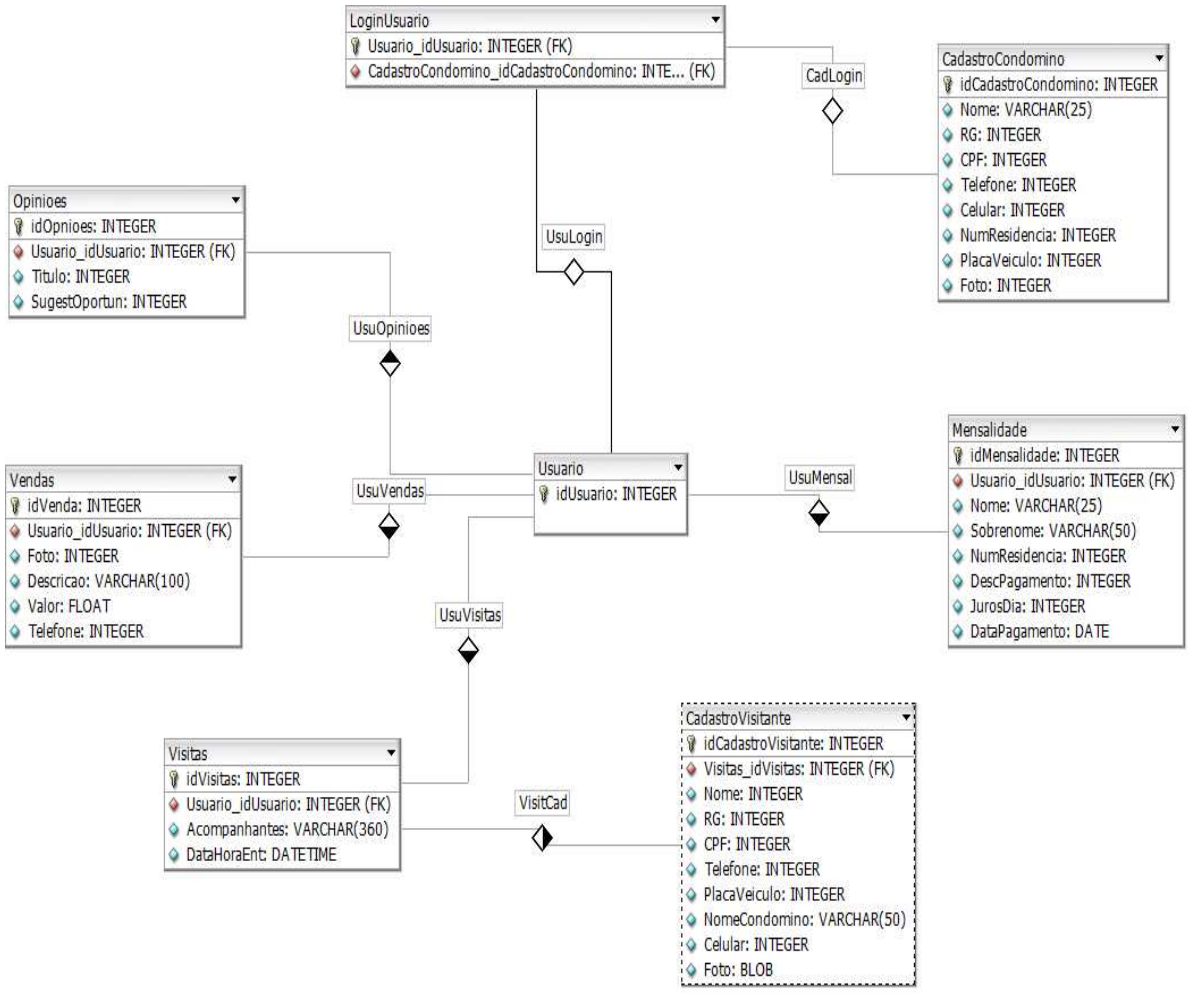

Figura 14 - Diagrama de Entidade Relacionamento - Condômino

5. CONCLUSÃO

Com a busca de agilidade e padronização no desenvolvimento de softwares, as aplicações web vêm crescendo mundialmente, para suprir as necessidades das empresas, com isso os softwares vêm para auxiliá-los. O software a ser desenvolvido nesse projeto visa atender a procura para ambiente Web na linguagem JAVA, provendo maior rapidez e controle das atividades do dia-a-dia nos condomínios, facilitando, ajudando a vida de moradores, usuários e administradores. A aplicação tende a crescer por meio do desenvolvimento, ajustes e melhorias para atender as necessidades de cada cliente e com isso se expandir no mercado.

O desenvolvimento deste trabalho contribui expressivamente para o crescimento profissional e reconhecimento do autor, proporcionando grandes conhecimentos específicos de tecnologias, as quais ainda não se tinha conhecimento para desenvolvimento de uma aplicação, já existente no mercado.

Para projetos futuros da aplicação, será feito um estudo, planejamento aprofundado da parte de mensalidade, taxas, cobranças e investimentos do condomínio, para atender melhor seus moradores e clientes.

REFERÊNCIAS

CAELUM. **Apostila do curso FJ-11 - Java e Orientação a Objetos**. Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos/>. Acesso em 6 de Março de 2013.

CAELUM. **Apostila do curso FJ-21 - Java para Desenvolvimento Web**. Disponível em: <http://www.caelum.com.br/apostila-java-web/>. Acesso em 6 de Março de 2013.

DEVMEDIA. **Conheça o Apache Tomcat**. Disponível em: <http://www.devmedia.com.br/conheca-o-apache-tomcat/4546>. Acesso em 10 de Junho de 2013.

FOWLER, Martin. **Um breve guia para linguagem-padrão de modelagem de objetos**. UML Essencial 2ª Edição, 2000.

OFICINA DA NET. **Apostila de HTML, XHTML e CSS**. Disponível em: <http://www.oficinadanet.com.br/apostilas/detalhe/710/apostila_de_html_xhtml_e_css>. Acesso em 5 de Junho de 2013.

ORACLE. **Netbeans IDE - A Forma Mais Inteligente de Codificar**. Disponível em: <http://netbeans.org/features/index_pt_BR.html>. Acesso em 9 de Março de 2013.

ORACLE. **Obtenha Informações sobre a Tecnologia Java**. Disponível em: <http://www.java.com/pt_BR/about>. Acesso em 5 de Março de 2013.

PEDROSO, Robertha Pereira. Niterói RJ (2007). **Apostila de HTML**. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/apostilas/HTML.pdf>. Acesso em 29 de Junho de 2013.

PEDROSO, Robertha Pereira. **Guia MySQL**. Niterói, Rio de janeiro, 2007.

RICARTE, Ivan Luiz Marques. **Introdução a Orientação a Objetos**. Disponível em: http://www.dca.fee.unicamp.br/cursos/POOCPP/node3.html>. Acesso em 5 de Março de 2013.