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

MATHEUS FARIA SANCHES

SISTEMA PARA GESTÃO DE OBRAS CIVIS

Assis
2012

MATHEUS FARIA SANCHES

SISTEMA PARA GESTÃO DE OBRAS CIVIS

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

Orientador: Dr° Almir Rogério Camolesi
Área de Concentração: Desenvolvimento de Sistemas

Assis
2012

FICHA CATALOGRÁFICA

SANCHES, Matheus Faria
Sistema para gestão de obras civis de pequeno a grande porte /
Matheus Faria Sanches. Fundação Educacional do Município de Assis, 2012.
56 p.

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

1. Gestão de Obras civis, Programação, Plataforma .NET, Linguagem de Programação C#, UML.

CDD: 001.61

SISTEMA PARA GESTÃO DE OBRAS CIVIS

MATHEUS FARIA SANCHES

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

Orientador: Dr° Almir Rogério Camolesi

Analisador: Esp. Célio Desiró

Assis

2012

DEDICATÓRIA

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

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem Ele, eu nada seria possível. Dele vem toda a sabedoria, tudo o que tenho e tudo o que sou.

Aos meus familiares, Sioneiva dos Santos Faria Rodrigues, Angelo Sanches Rodrigues e Aline Faria Sanches, por sempre estarem ao meu lado, apoiando-me e direcionando-me ao caminho correto a seguir.

Ao meu orientador, Dr° Almir Rogério Camolesi, pela orientação, durante todo o período deste trabalho e também por toda a caminhada acadêmica.

E a todos aqueles que estiveram ao meu lado direta ou indiretamente, torcendo sempre e ajudando nos momentos de dificuldade.

"Você pode encarar um erro como uma besteira a ser esquecida, ou como um resultado que aponta uma nova direção".

Steve Jobs

(1955 - 2011)

RESUMO

Atualmente, a tecnologia da informação vem crescendo cada vez mais, no Brasil e no mundo, um dos fatores que tem contribuído para esse crescimento, é uso de softwares específicos, devido à necessidade de um maior controle e levantamento de dados no processo de produção.

Neste trabalho serão apresentados a especificação e a realização de um aplicativo Web, destinado a empresas que atuam no ramo de construções civis, proporcionando soluções inovadoras, de fácil utilização, agilizando processos e reduzindo custos.

Para o desenvolvimento deste trabalho foi efetuado um estudo sobre tecnologias a serem utilizadas para a realização do sistema, como a linguagem de programação C#, o banco de dados Microsoft SQL Server 2008 R2.

ABSTRACT

Nowadays, information technology is growing increasingly, in Brazil and in the world, and one of the factors that has contributed to this growth is the use of specific software, due to the need for greater control and survey data production process.

In this work will be present the specification of a Web application aimed at companies operating in the field of civil construction, providing innovative, easy to use, streamlining processes and reducing costs.

For the development of this work was carried out a study of technologies to be used for the realization of the system, such as the C # programming language, database Microsoft SQL Server 2008 R2.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

Sumário

1 INTRODUÇÃO

A maneira como as informações são criadas, armazenadas e compartilhadas modificou-se muito no decorrer dos anos, devido aos avanços tecnológicos.

Atualmente, a área de tecnologia é uma das áreas que mais crescem no Brasil e no mundo. As empresas procuram cada vez mais softwares específicos, para auxiliar na gestão dos processos de produção.

Tendo em vista tais necessidades, o sistema *BuildManager* vem disponibilizar, para o mercado de construção civil, uma aplicação para gestão de obras civis de pequeno a grande porte, tornando fácil o controle de todos os processos e equipamentos utilizados por dia de trabalho.

O sistema *BuildManager* tem a finalidade de gerenciar e facilitar o controle diário das obras, contendo informações referentes às ocorrências diárias.

É o sistema ideal para auxiliar as empresas que prestam esse tipo de serviços, pois possui como principais funcionalidades: interfaces para cadastro de obras, empreiteiras, condições climáticas referentes ao dia da obra, ocorrências, equipamentos e maquinário utilizado na obra.

Os recursos disponibilizados pelo software proporcionarão ao cliente, facilidades que contribuem para o sucesso da obra.

1.1 OBJETIVO

Como objetivo principal, o sistema *BuildManager* será desenvolvido para o mercado de gerenciamento de obras civis, o qual facilitará os processos, controlará e reduzirá as despesas obtidas no decorrer da construção, tudo de forma flexível, tendo o controle de todas as tarefas antes e durante a construção.

1.2 JUSTIFICATIVA

A justificativa para o *BuildManager* ser um sistema Web, se dá pelo crescimento na utilização de tais meios de computação, facilitando o acesso às informações e trazendo grandes vantagens para seus usuários.

As empresas, buscam por softwares seguros, modernos e precisos para auxiliarem seus negócios, reduzindo custos e elevando a qualidade de seus serviços, portanto, o software *BuildManager* fará uso da linguagem de programação C# e do banco de dados SQL Server 2008 R2, objetivando a segurança das informações.

Contudo, o desenvolvimento da ferramenta atenderá as exigências descritas e acompanhará as evoluções tecnológicas, competindo com este nicho de mercado, além de contribuir para o crescimento e o enriquecimento profissional.

1.3 PÚBLICO ALVO

O software atende as necessidades das empresas voltadas à área de construções civis, e a empresas interessadas no sistema, com a finalidade de atender as necessidades organizacionais dos projetos da mesma.

1.4 ESTRUTURA DO TRABALHO

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

O primeiro capítulo apresentou a contextualização e a justificativa para o desenvolvimento da proposta de trabalho.

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

O terceiro capítulo apresenta as etapas e especificações do software contemplando o levantamento de requisitos, lista de eventos, caso de uso e

suas especificações e os principais diagramas UML (classe, sequência e atividade).

O quarto capítulo descreve a WBS – *Work Breakdown Structure*, o sequenciamento das atividades e o orçamento do software.

O quinto capítulo apresenta a implementação do sistema, exibindo um detalhamento sobre a aplicação desenvolvida assim como a distribuição das camadas, organização do projeto e interface criada para interagir com o usuário final.

Por fim, no ultimo capítulo serão apresentados a conclusão do trabalho, organograma atualizado e trabalhos futuros.

2 TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

Neste capítulo serão descritas as tecnologias e ferramentas utilizadas para o desenvolvimento do sistema *BuildManager.*

O *BuildManager* é um sistema desenvolvido com a tecnologia Microsoft .NET, que possui uma linguagem de alto nível orientada a objetos. O ambiente utilizado para o desenvolvimento do sistema foi o Visual Studio 2010 Ultimate. A arquitetura do projeto faz uso do padrão MVC – *Model Viewer Controller*, responsável por separar as regras de negócio das camadas de visualização e de controle, fazendo com que a reutilização de códigos seja possível, além de facilitar os momentos de manutenção do projeto.

Como base de dados, foi utilizado o banco de dados SQL Server 2008 R2*,* o qual responde a instruções de seleção e manipulação de dados através do framework *NHibernate*, utilizado para realizar o mapeamento objeto/relacional, facilitando as operações na camada de persistência da aplicação.

E finalmente para a avaliação gerencial de dados, utiliza-se a ferramenta *Crystal Reports*, para maior facilidade e praticidade na construção de relatórios e também para definir designers modernos através de uma interface gráfica intuitiva, sem a necessidade de escrever linhas de código.

1.5 C#

O C# é uma linguagem de programação orientada a objetos, desenvolvida pela Microsoft como parte da plataforma .NET e baseada na linguagem C++ (ALEXANDRE, 2012).

A linguagem de programação C# foi criada junto com a plataforma .NET e é considerada a linguagem símbolo .NET por algumas razões apresentadas abaixo (ALEXANDRE, 2012):

- ✓ Criada do zero para funcionar em uma nova plataforma, sem preocupações de compatibilidade com código;
- ✓ O compilador da linguagem C# foi o primeiro a ser desenvolvido;

- ✓ A maior parte das classes do .NET Framework foram desenvolvidas em C#.

A criação da linguagem é atribuída a *Anders Hejsberg,* apesar de ter sido criada por vários desenvolvedores, que hoje ocupado o cargo de *Distinguished Engineer* na Microsoft. *Anders* era desenvolvedor de compiladores na Borland, e entre suas criações mais conhecidas estão o Turbo Pascal e o Delphi (HAMILTON, 2008).

As estruturas de dados da linguagem de programação C# são objetos que correspondem a tipos em .NET. A liberação automática de memória por *garbage collector*, além de várias de suas abstrações tais como classes, interfaces, delegados e exceções, são nada mais que a exposição de recursos do ambiente .NET (ARAÚJO, 2008).

Quando compara-se C# com as linguagens C e C++, a linguagem é restrita e melhorada de várias formas incluindo:

- ✓ Ponteiros só podem ser utilizados em uma modalidade especial chamada de *unsafe mode* (modo inseguro) (INTERNATIONAL, 2006);
- ✓ Objetos só são liberados através de um processo de coleta de lixo (*garbage collector*) quando não existe nenhuma referencia aos mesmos (INTERNATIONAL, 2006);
- ✓ Destrutores não existem. Existe uma interface chamada *Disposable*, o qual permite que recursos alocados por um objeto sejam liberados prontamente (INTERNATIONAL, 2006);
- ✓ Não é permitida herança múltipla (INTERNATIONAL, 2006);
- ✓ É mais seguro com tipos que a linguagem C++, pois as únicas conversões implícitas por default são conversões seguras, tais como ampliação de inteiros e conversões de um tipo derivado para um tipo base (INTERNATIONAL, 2006).

Apesar de C# ser frequentemente tido como similar a Java, existem várias diferenças importantes entre as duas linguagens, mas a maioria é implementada de forma diferenciada em ambas as linguagens.

1.6 MICROSOFT SQL SERVER 2008 R2

O Microsoft SQL Server é um conjunto completo de tecnologias e ferramentas *enterprise-ready* que ajudam a gerar o máximo de informações possíveis a curto espaço de tempo, com altos níveis de desempenho, disponibilidade e segurança de dados, sendo possível empregar ferramentas de gerenciamento e desenvolvimento de aplicações (MICROSOFT, 2012).

Esse Sistema Gerenciador de Banco de Dados Relacional (SGBD), foi criado pela Microsoft em parceria com a Sybase em 1988 e inserido como produto complementar do Windows NT. Em 1994, a parceria foi rompida e a Microsoft continuou aperfeiçoando o produto.

É uma plataforma de dados confiável, produtiva e inteligente que permite a execução de aplicações de missão critica mais exigentes, reduzindo o tempo e o custo com o desenvolvimento e o gerenciamento de aplicações, além de ser um banco de dados robusto e usado por sistemas corporativos dos mais diversos portes (PORTAL EDUCAÇÃO, 2008).

1.7 NHIBERNATE

O NHibernate é um *framework* de mapeamento objeto/relacional para a linguagem de programação C# que facilita o desenvolvimento de aplicações e da independência de banco de dados, fazendo com que o programador se preocupe mais com seu modelo de objeto e seu comportamentos do que com tabelas.

O mapeamento objeto/relacional (ORM) refere-se a técnica de mapear os registros do banco de dados em objetos e persistir as informações contidas nos objetos em forma de linhas e colunas (MAULO, 2008).

É livre e de código aberto e é a versão portada do Java para o Microsoft .NET do Hibernate. Lida com a persistência para objetos de dados relacionais.

NHibernate gera automaticamente códigos SQL para carregar e guardar objetos. Suporta persistência transparente, o objeto não tem de seguir um modelo de programação restritiva.

Classes persistentes não precisam implementar nenhuma interface ou herdar de uma classe especial base, tornando possível desenvolver a lógica empresarial utilizando o plano de objetos .NET e Orientação a Objetos.

1.8 PADRÃO DE PROJETO MVC

Um padrão de projeto tem como função nomear, abstrair e identificar os aspectos-chaves de uma estrutura de projeto comum, tornando-a útil para a criação de um projeto orientado a objetos de maneira reutilizável (TAMIRES ALVES DA SILVA, 2011).

O padrão de projeto *Model View Controller* (MVC) tem como objetivo separar a lógica de negócios (*Model)* da interface de usuário (*View*) e do fluxo da aplicação (*Controller*), permitindo o desenvolvimento, teste e manutenção isolada de ambos. Permite a reutilização dos códigos implementados nas camadas.

- ✓ **Model ou Modelo:** Responsável por grande parte do código escrito. Nesta camada está presente a lógica de negócios, com o objetivo de definir e gerenciar toda a informação, bem como a notificação sobre possíveis mudanças ocorridas nos dados (EDUARDO, 2011).
- ✓ **View ou Visão:** Compreende a interface de usuário e é responsável pelo acesso aos dados contidos em *Model* e especifica como estes dados colhidos serão apresentados ao usuário (EDUARDO, 2011).

- ✓ **Controller ou Controle:** Responsável pelo fluxo da aplicação, assume o mapeamento das ações efetuadas pelo usuário na camada *View*, por meio de eventos e, com isso, permite que a camada *Model* seja alterada.

A Figura 1 ilustra o funcionamento do padrão de projeto MVC:

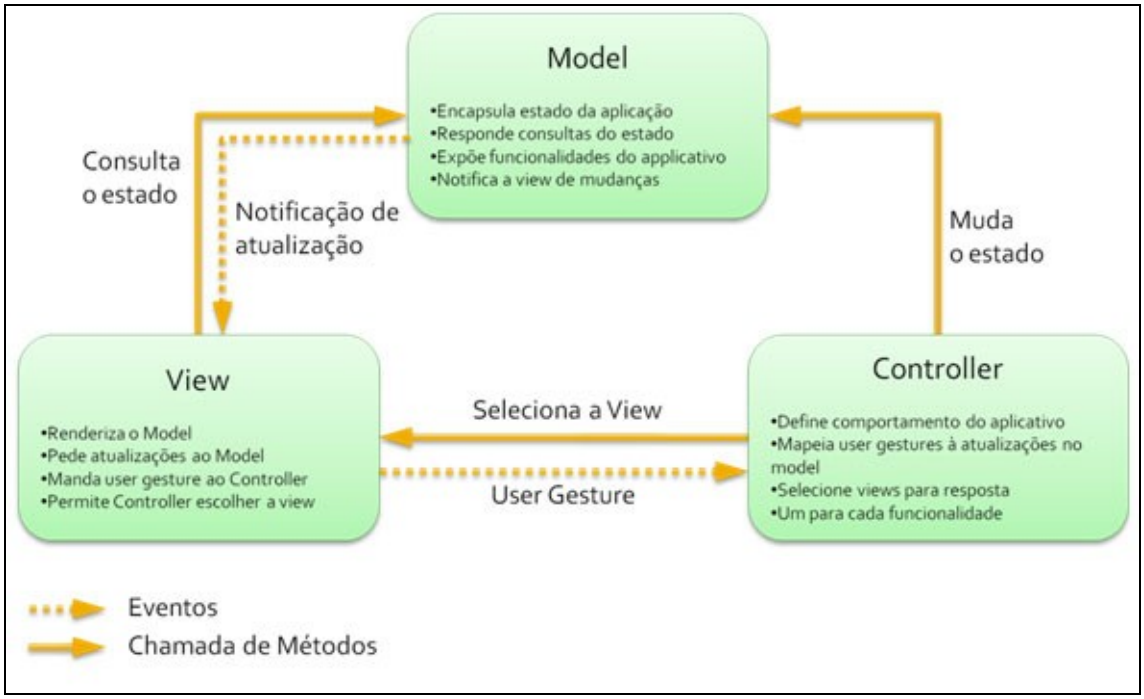


Figura 1 Funcionamento do padrão de projetos MVC

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 web.

1.9 CRYSTAL REPORTS

A ferramenta Crystal Reports permite a construção de relatórios, desde o mais simples ao mais complexo para aplicações .NET ou diretamente em aplicações web. Pertence à SAP Business Objects e é um dos mais utilizados no mundo. Permite a inserção de fórmulas nos relatórios e também o recebimento de dados de um sistema, via Recordset ou através de uma conexão direta a um banco de dados relacional.

3 ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Neste capítulo será apresentado a especificação e o projeto do sistema proposto.

1.10 MAPA MENTAL

Para facilitar o entendimento e a visualização do sistema é apresentado o mapa mental que é um método utilizado para armazenar, organizar e priorizar informações através de diagramas elaborados a partir de uma ideia principal. Sistematizado pelo inglês Tony Buzan, voltado a gestão de informações, de conhecimento e de capital intelectual (TRÍBOLI, 2004).

A Figura 2 mostra o mapa mental referente às atividades internas do software *BuildManager*:

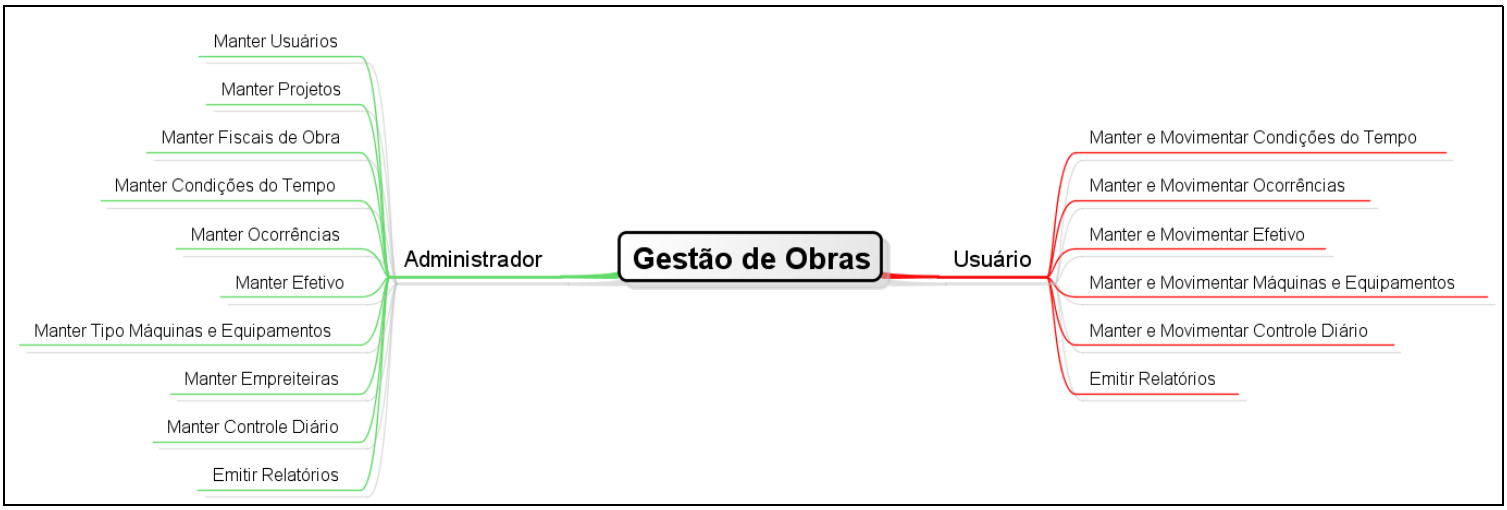


Figura 2 Mapa mental

1.11 LISTA DE EVENTOS

Para ser realizada a visualização, especificação, construção e documentação dos artefatos de um sistema, utiliza-se a *UML*, pois é uma linguagem padrão para a elaboração de estrutura de projetos de software. É adequada para a

modelagem de sistemas, o qual poderá incluir aplicações de vários níveis de complexidade, desde as aplicações simples ate sistemas complexos.

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 do sistema:

Lista de Eventos

1. Manter Fiscais de Obra
2. Manter Projetos
3. Manter Condições do Tempo
4. Manter Ocorrências
5. Manter Efetivo
6. Manter Empreiteiras
7. Manter Tipo de Máquinas e Equipamentos
8. Manter Controle Diário
9. Movimentar Projetos
10. Movimentar Ocorrências
11. Movimentar Efetivo
12. Movimentar Tipo de Empreiteiras
13. Movimentar Máquinas e Equipamentos
14. Emitir Relatório de Projetos por Período
15. Emitir Relatório de Projetos em Andamento
16. Emitir Relatório de Projetos Finalizados
17. Emitir Relatório de Projetos Futuros
18. Emitir Relatório de Resumo do Projeto
19. Emitir Relatório de Projetos
20. Emitir Relatório de Empreiteiras por Tipo
21. Emitir Relatório de Empreiteiras em Trabalho
22. Emitir Relatório de Ocorrências por Dia
23. Emitir Relatório de Ocorrências por Projeto
24. Emitir Relatório de Efetivo por Dia
25. Emitir Relatório de Efetivo por Projeto

26. Emitir Relatório de Máquinas e Equipamentos por Tipo
27. Emitir Relatório de Máquinas e Equipamentos
28. Emitir Relatório de Controle Diário

1.12 CASOS DE USO

O primeiro elemento da UML utilizado é o 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 3 mostra o caso de uso geral comum entre os atores.

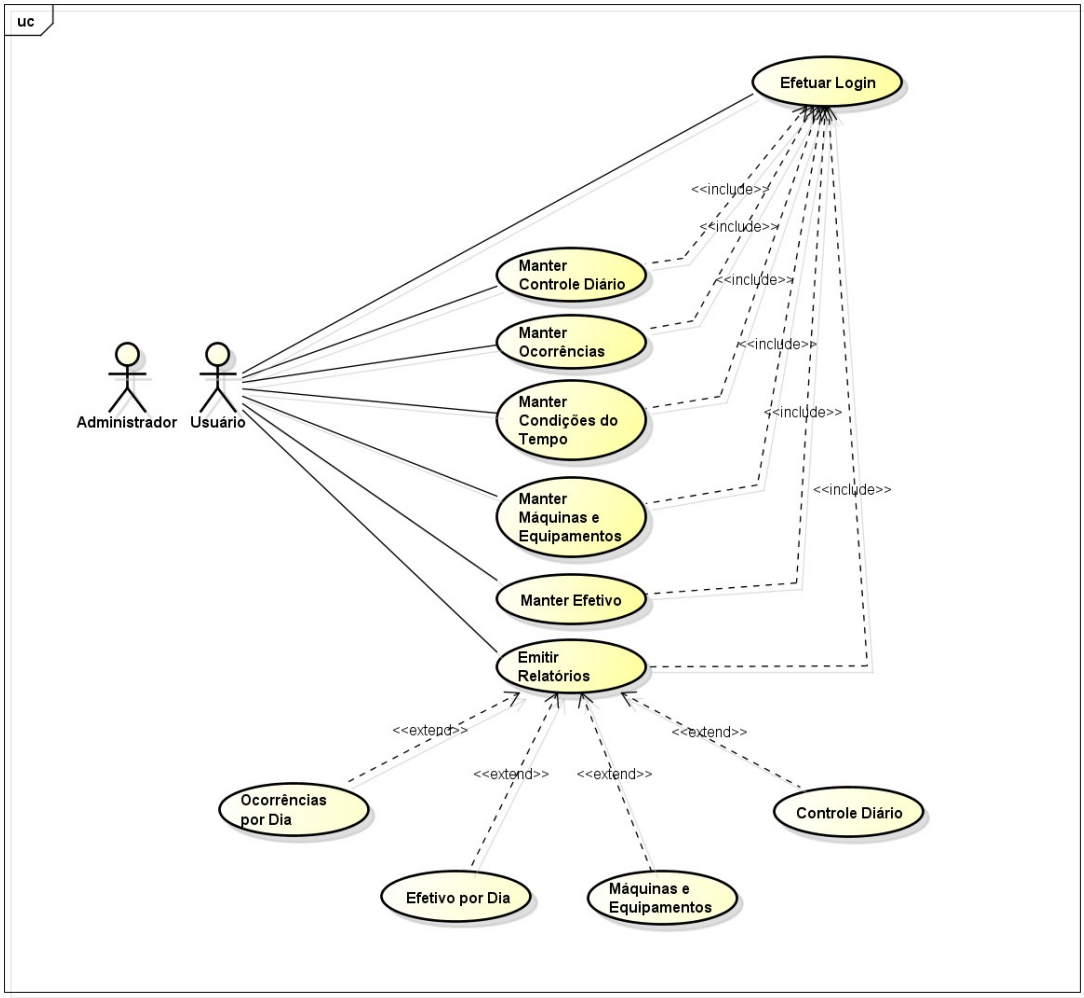


Figura 3 Caso de Uso Geral em comum – Administrador e Usuário

A Figura 4 mostra o caso de uso geral do Administrador.

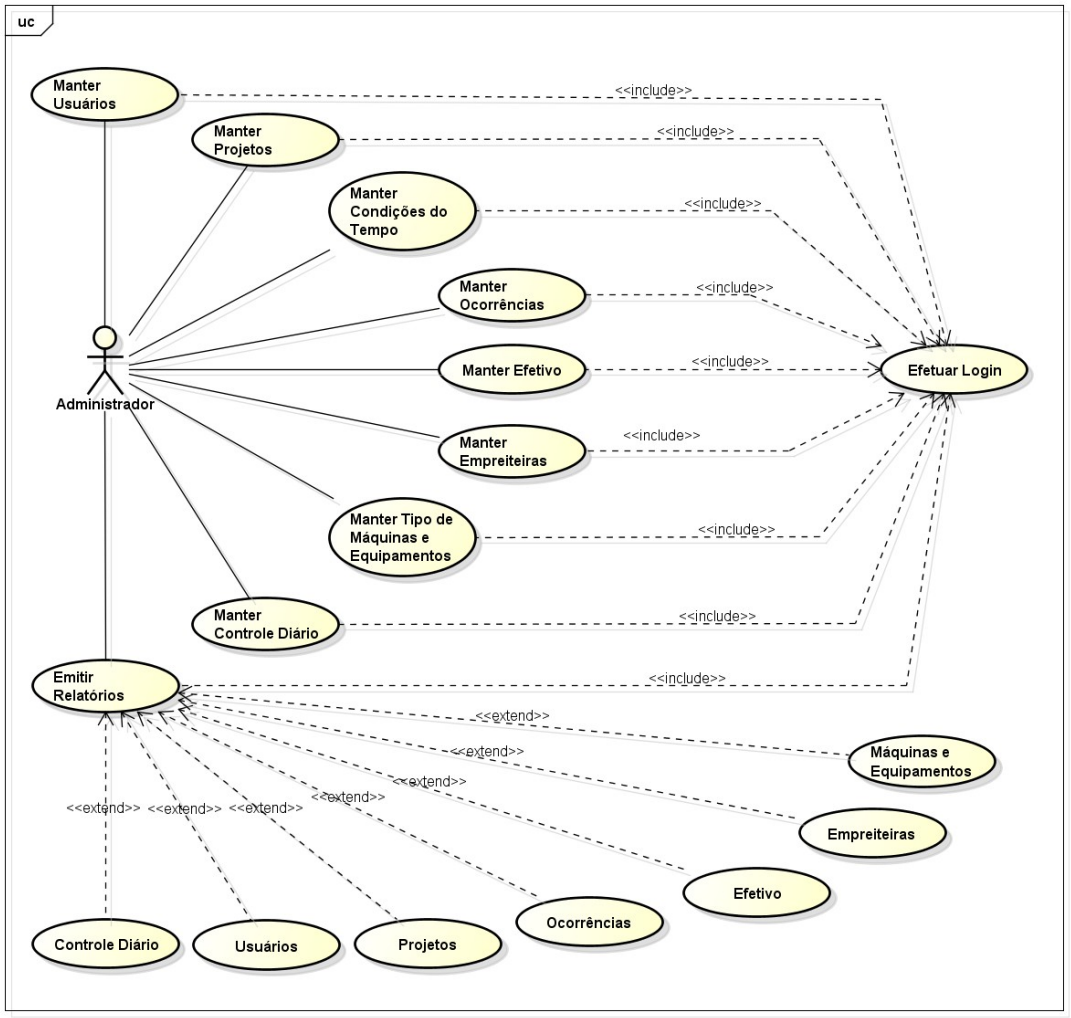


Figura 4 Caso de Uso – Administrador

A Figura 5 mostra o caso de uso geral do usuário.

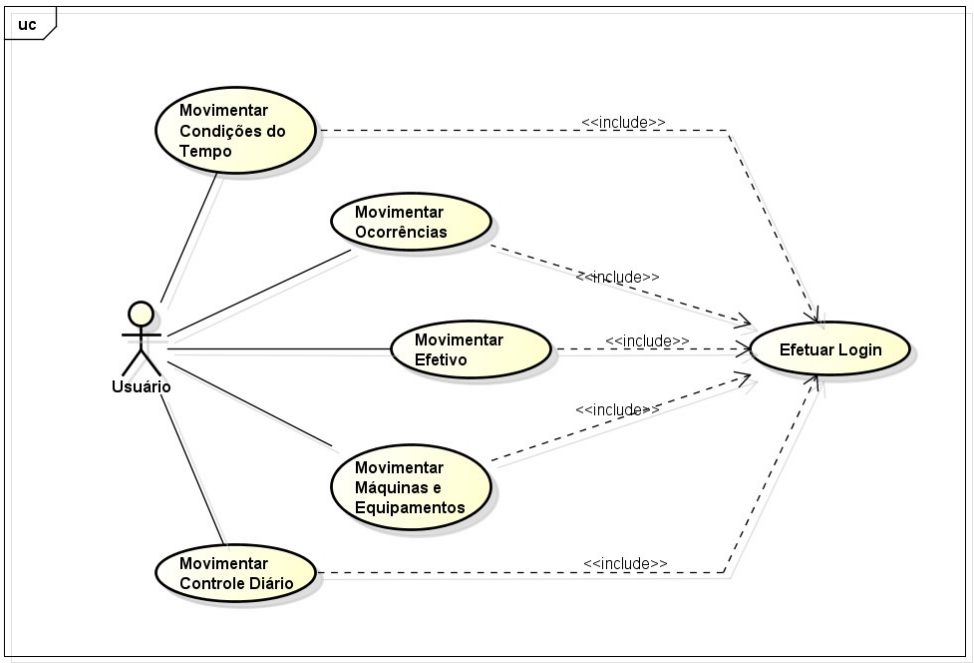


Figura 5 Caso de Uso – Usuário

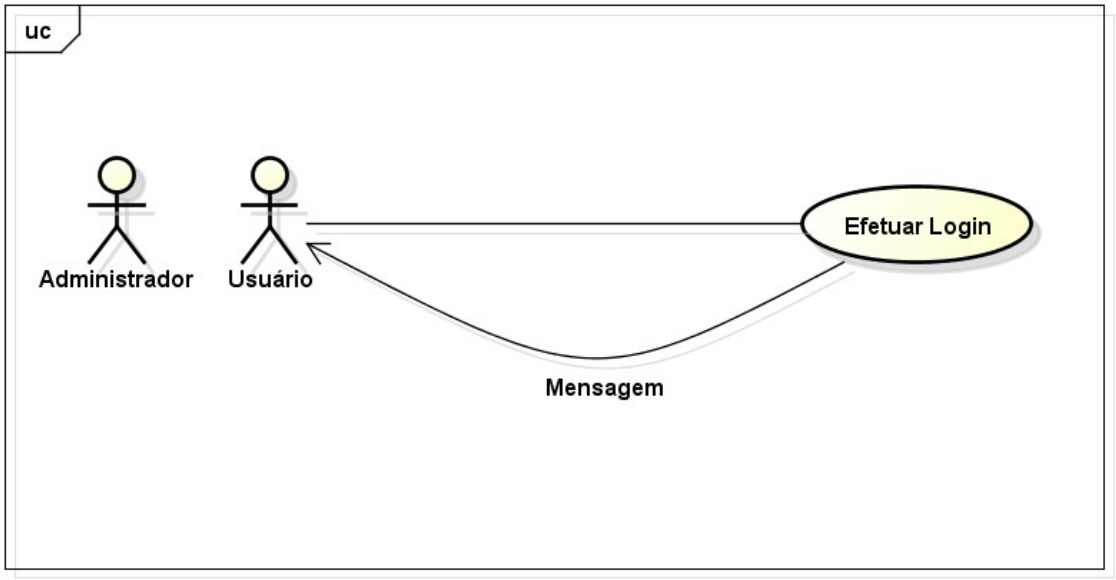
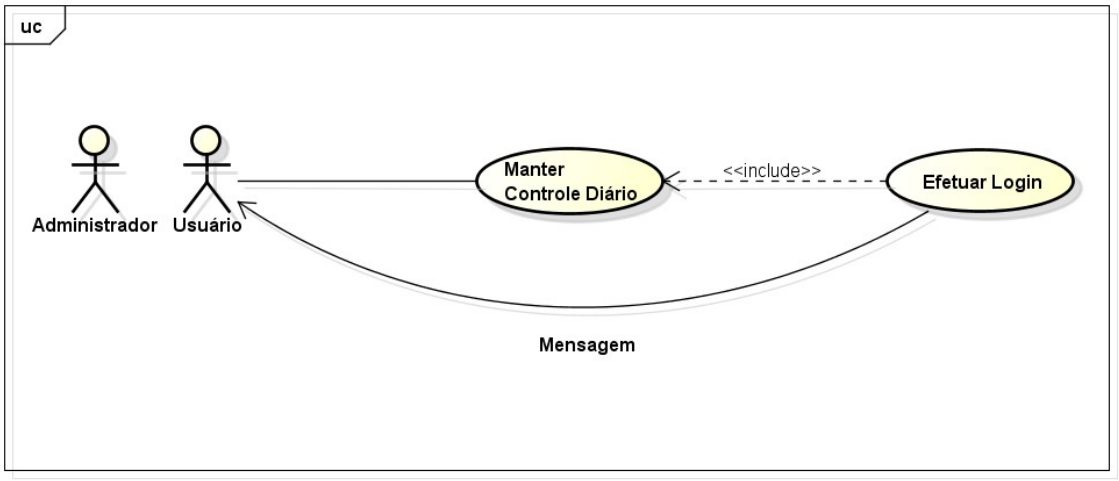


Figura 6 UC1 Diagrama de caso de uso Efetuar Controle Acesso

Nome do caso de uso 1	Efetuar Controle de Acesso
Atores	Administrador e 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.
Cenários alternativos	Não existe.
Casos de teste	4.1 – Se os dados estiverem corretos, executa a operação solicitada. 4.2 – Se os dados estiverem incorretos, cancela a operação e exibe mensagem de alerta.

Tabela 1 Efetuar Controle de Acesso

Figura 7 UC2 Diagrama de Caso de Uso Manter Controle Diário



Nome do caso de uso 2	Manter controle diário
Atores	Administrador e usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador ou usuário informa os dados do dia da obra. 2- O administrador informa um projeto (Caso de uso manter projetos). 3- O administrador informa uma empreiteira (Caso de uso manter empreiteiras). 4- O sistema valida os dados informados. 5- O administrador ou usuário seleciona a opção “Cadastrar”. 6- O sistema emite mensagem de sucesso. 7- O sistema cadastra o controle diário.
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 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 projeto. 5.2 O sistema verifica se o administrador informou uma empreiteira.

Tabela 2 Manter Controle Diário

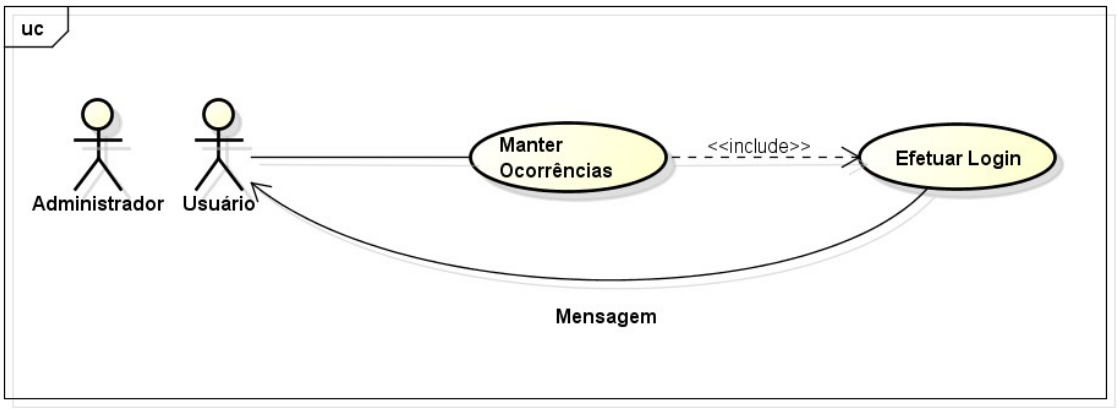


Figura 8 UC3 Diagrama de caso de uso Manter Ocorrências

Nome do caso de uso 3	Manter Ocorrências
Atores	Administrador e usuário
Pré-condições	Efetuar Controle de Acesso
Cenário principal	1- O administrador ou usuário informa as ocorrências. 2- O sistema valida os dados informados. 3- O administrador ou usuário seleciona a opção “Cadastrar”. 4- O sistema emite mensagem de sucesso. 5- O sistema cadastra a ocorrência.
Cenário alternativo	O administrador ou usuário poderá cancelar o processo durante o cadastro.
Casos de teste	Não existe.

Tabela 3 Manter Ocorrências

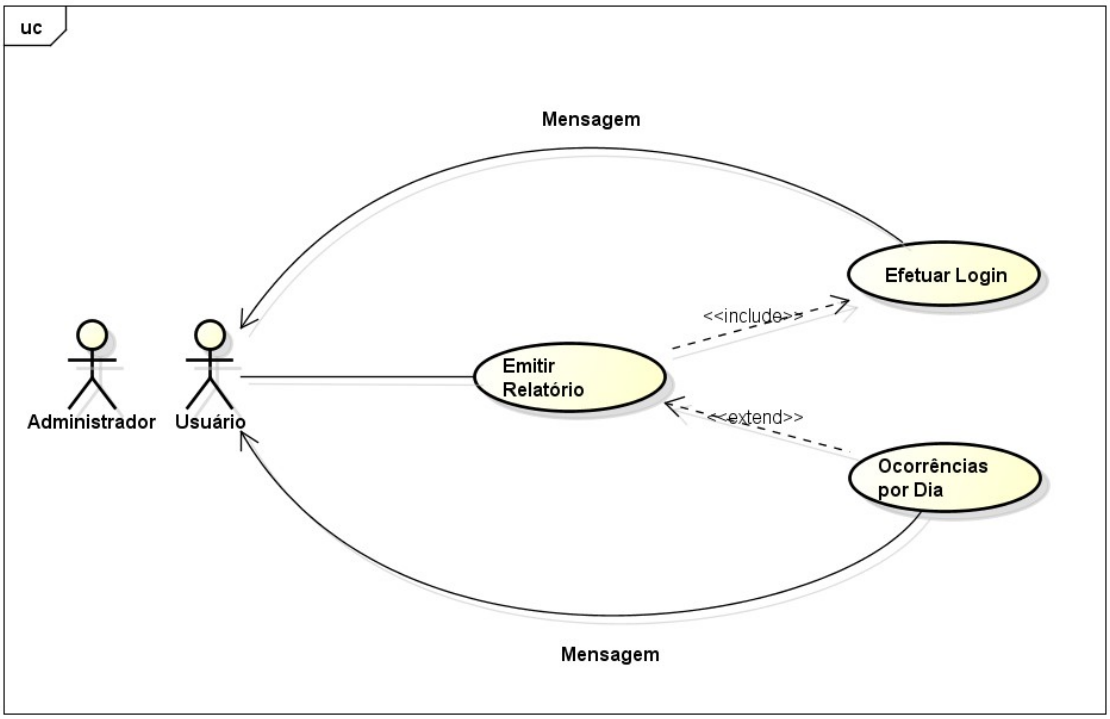


Figura 9 UC7 Diagrama de caso de uso Emitir Relatório de Ocorrências por Dia

Nome do caso de uso 7	Emitir relatório de ocorrências por dia
Atores	Administrador e usuário
Pré-condições	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 salvar. 4- O sistema salva o relatório com sucesso.
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não salvar.
Casos de teste	4.1 O administrador e/ou usuário cancela a operação.

Tabela 4 Emitir Relatório de Ocorrências por Dia

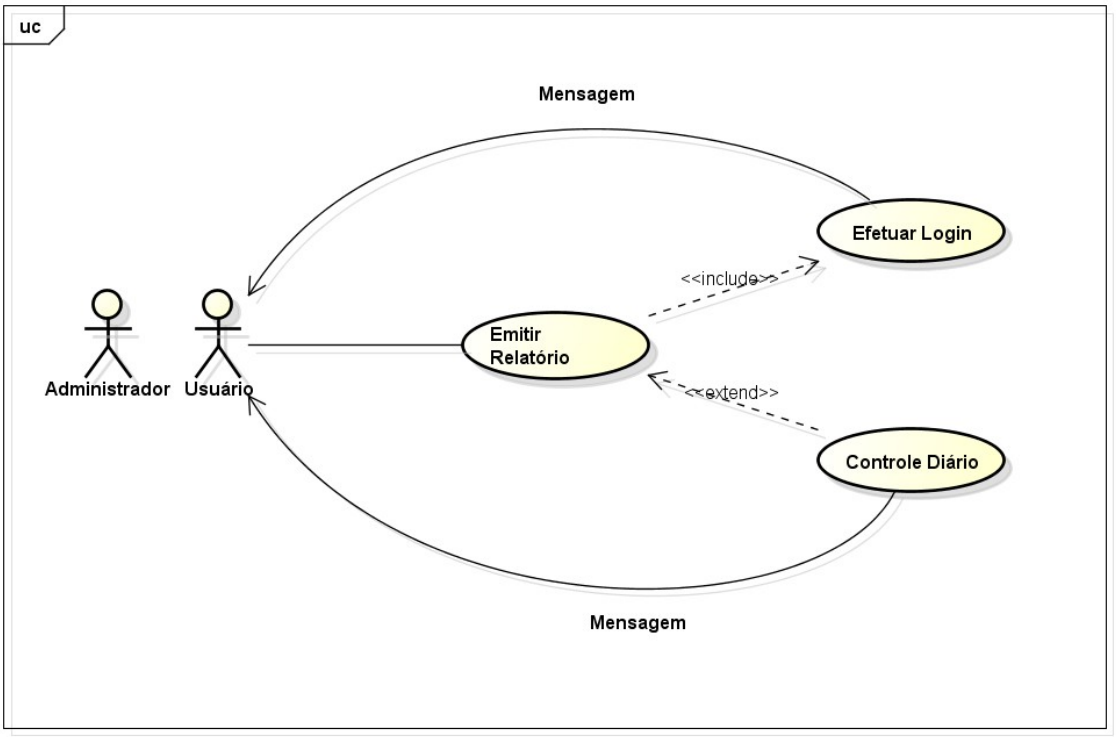


Figura 10 UC10 Diagrama de caso de uso Emitir Relatório de Controle Diário

Nome do caso de uso 10	Emitir relatório de controle diário
Atores	Administrador e usuário
Pré-condições	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 salvar. 4- O sistema salva o relatório com sucesso.
Cenário alternativo	O administrador e/ou usuário poderá visualizar o relatório e não salvar.
Casos de teste	4.1 O administrador e/ou usuário cancela a operação.

Tabela 5 Emitir Relatório de Controle Diário

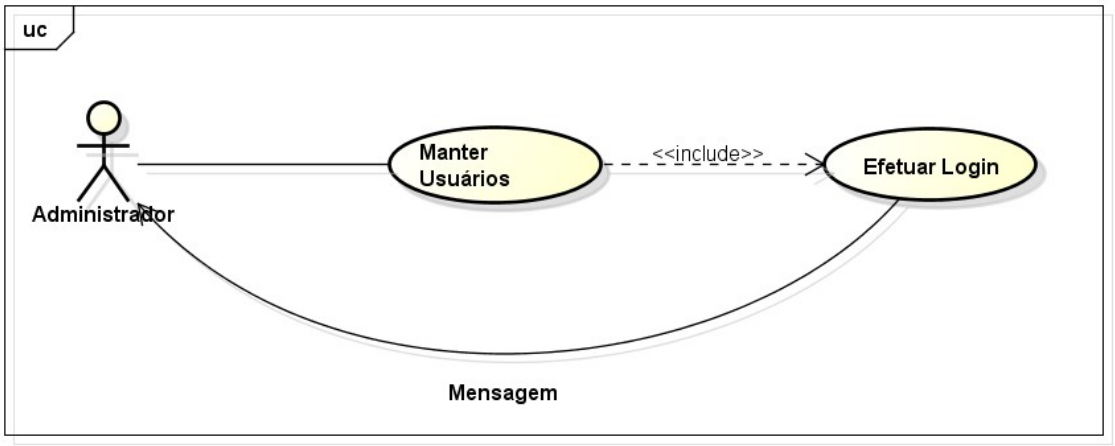


Figura 11 UC11 Diagrama de caso de uso Manter Usuários

Nome do caso de uso 11	Manter usuários
Atores	Administrador
Pré-condições	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa o login e senha do usuário. 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 usuário.
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 corretamente preenchidos. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 6 Manter Usuários

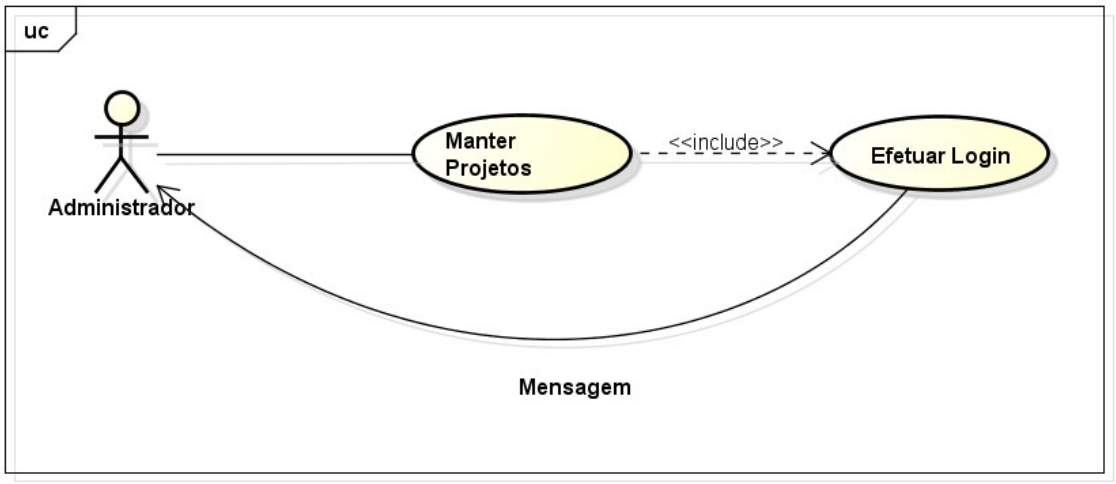


Figura 12 UC12 Diagrama de caso de uso Manter Controle Diário

Nome do caso de uso 12	Manter projetos
Atores	Administrador
Pré-condições	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados do projeto. 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 usuário.
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 corretamente preenchidos. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação.

Tabela 7 Manter Projetos

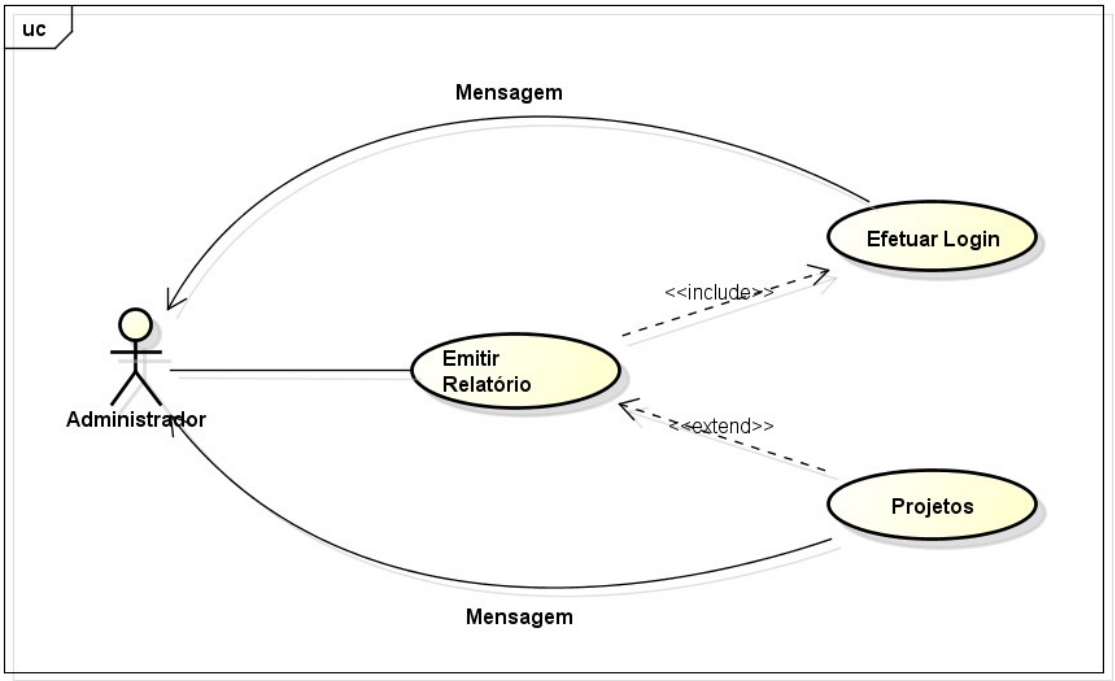


Figura 13 UC21 Diagrama de caso de uso Emitir Relatório de Projetos

Nome do caso de uso 21	Emitir relatório de projetos.
Atores	Administrador
Pré-condições	Efetuar Controle de Acesso
Cenário principal	1- O sistema disponibiliza os dados necessários para o relatório. 2- O administrador seleciona o botão visualizar relatório. 3- O administrador seleciona o botão salvar. 4- O sistema salva o relatório com sucesso.
Cenário alternativo	O administrador poderá visualizar e não salvar o relatório.
Casos de teste	4.1 O administrador cancela a operação.

Tabela 8 Emitir Relatório de Projetos

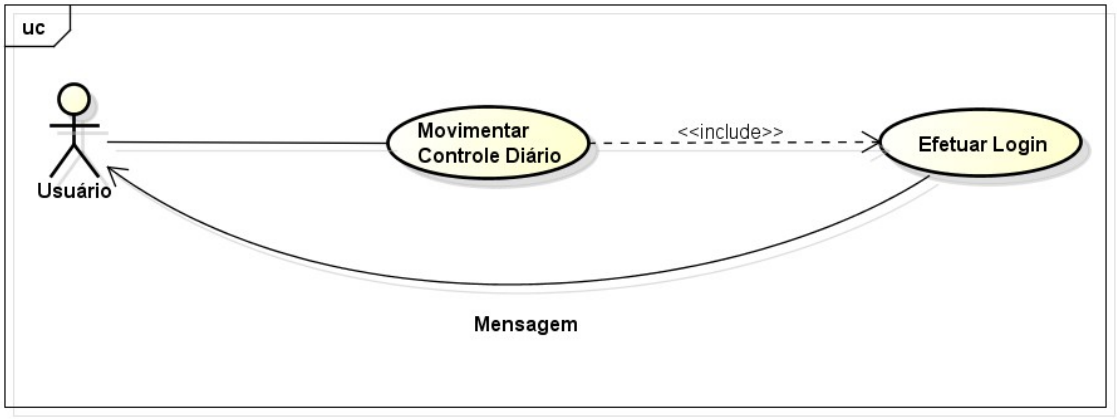


Figura 14 UC30 Diagrama de caso de uso Movimentar Controle Diário

Nome do caso de uso 30	Movimentar controle diário
Atores	Usuário
Pré-condições	Efetuar Controle de Acesso
Cenário principal	1- O usuário informa os dados da obra. 2- O usuário informa a data do controle. 3- O usuário informa as ocorrências do dia. 4- O usuário informa o efetivo do dia. 5- O usuário informa as máquinas e equipamentos utilizados no dia. 6- O usuário informa as condições do tempo no dia. 7- O sistema valida os dados informados. 8- O usuário seleciona a opção “Cadastrar”. 9- O sistema emite mensagem de sucesso. 10- O sistema cadastra o controle diário da obra.
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 corretamente preenchidos. 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 projeto. 7.2 O sistema verifica se o usuário informou uma empreiteira.

Tabela 9 Movimentar Controle Diário

1.13 DIAGRAMA DE ATIVIDADES

O diagrama de atividades representa os fluxos conduzidos por processamentos. É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra (BOOCH; JACOBSON; RUMBAUGH, 2000).

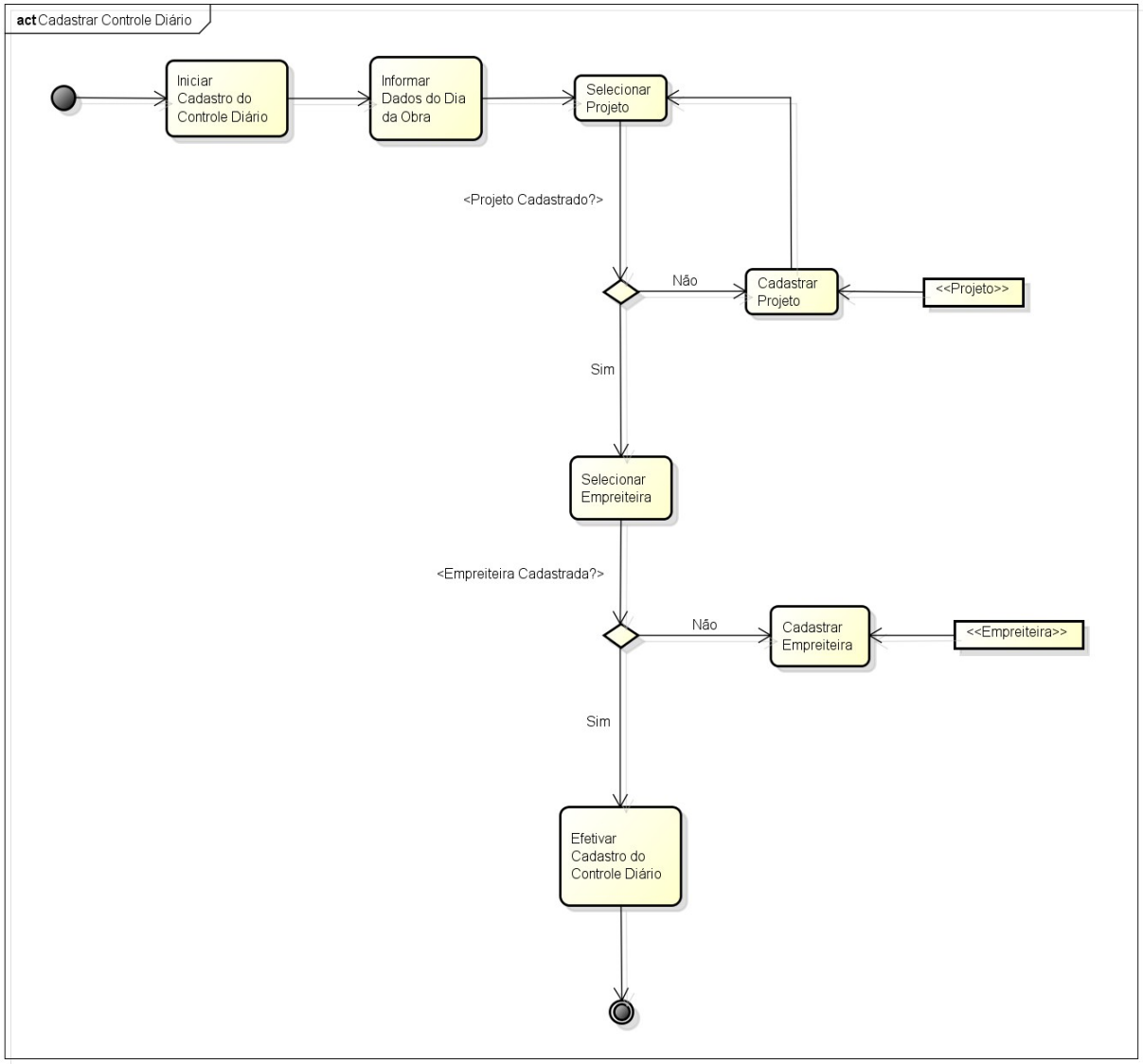


Figura 15 Diagrama de Atividades – Cadastrar Controle Diário

1.14 DIAGRAMA DE SEQUÊNCIAS

Um diagrama de sequencia determina a sequência de eventos que ocorrem em um determinado caso de uso. 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 (MACORATTI, 2012).

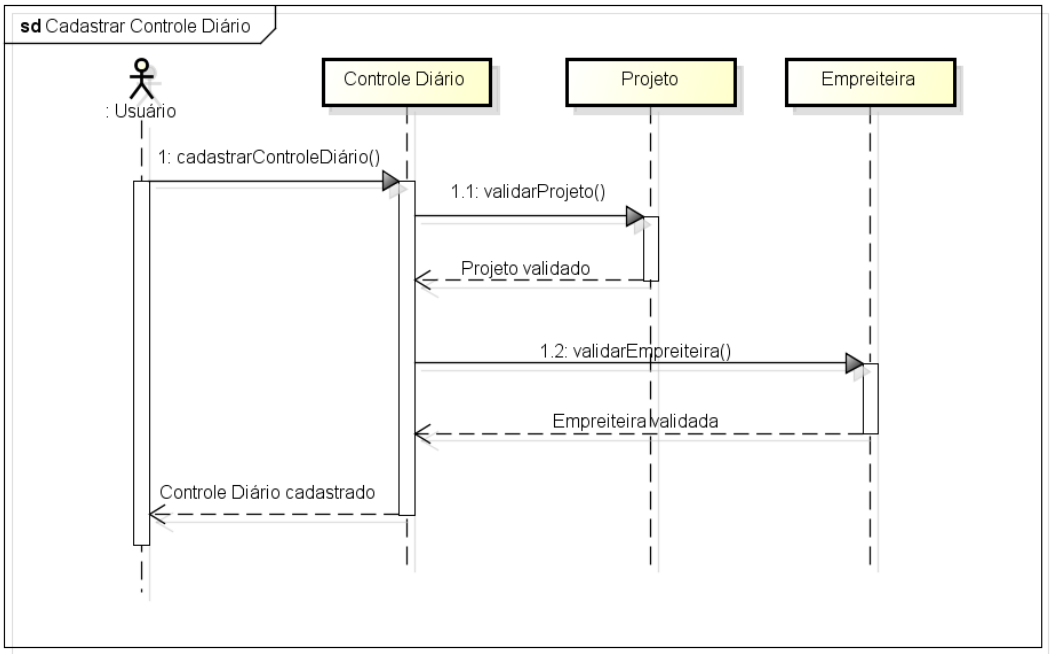


Figura 16 Diagrama de Sequências – Cadastrar Controle Diário

1.15 DIAGRAMA DE CLASSES

Um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos. É muito útil para o sistema, pois define todas as classes que o sistema necessita possuir (MACORATTI, 2012).

Abaixo, as Figuras 17 e 18, mostram o diagrama de classes de acordo com o padrão de projetos MVC.

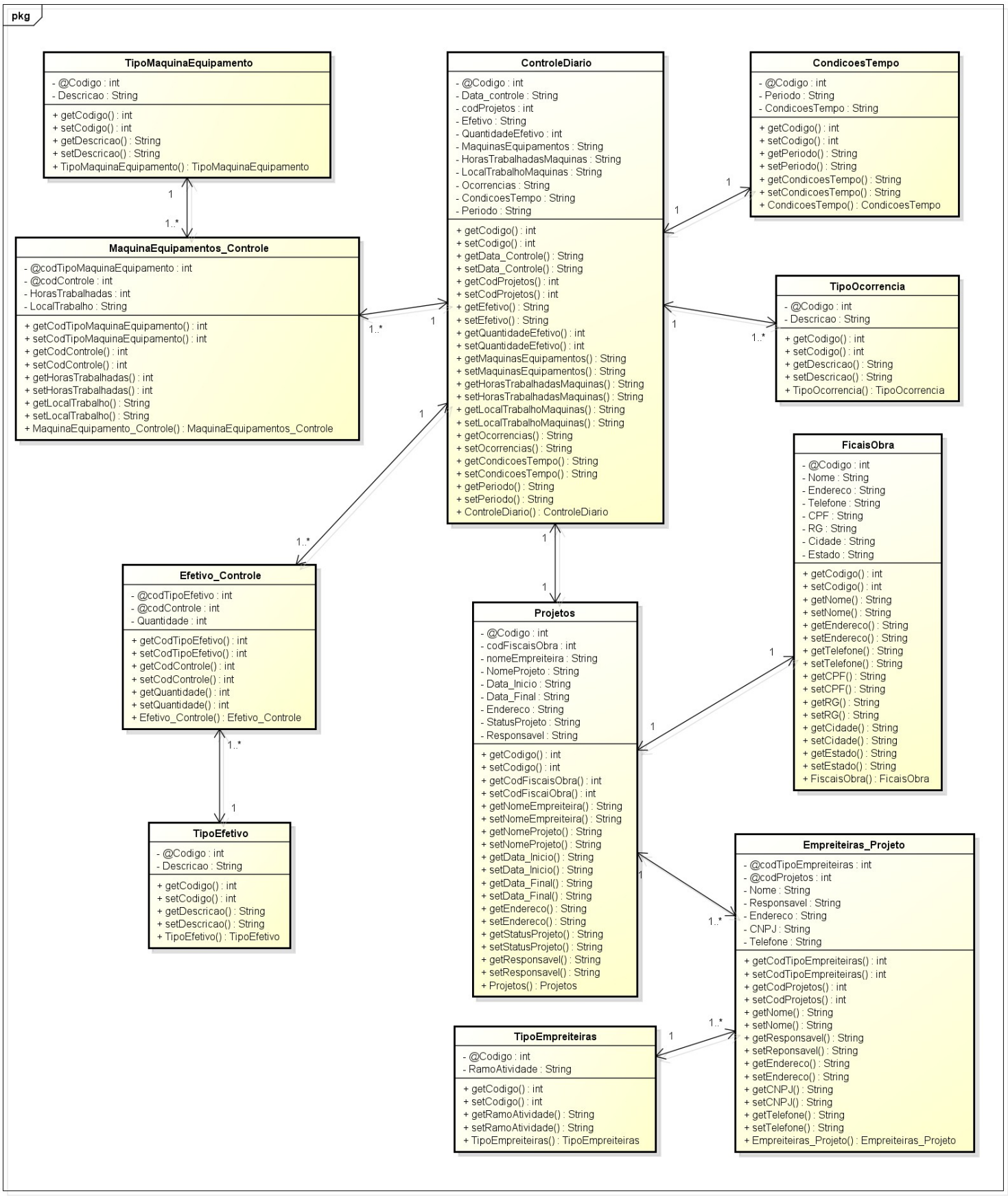


Figura 17 Diagrama de Classes - Model

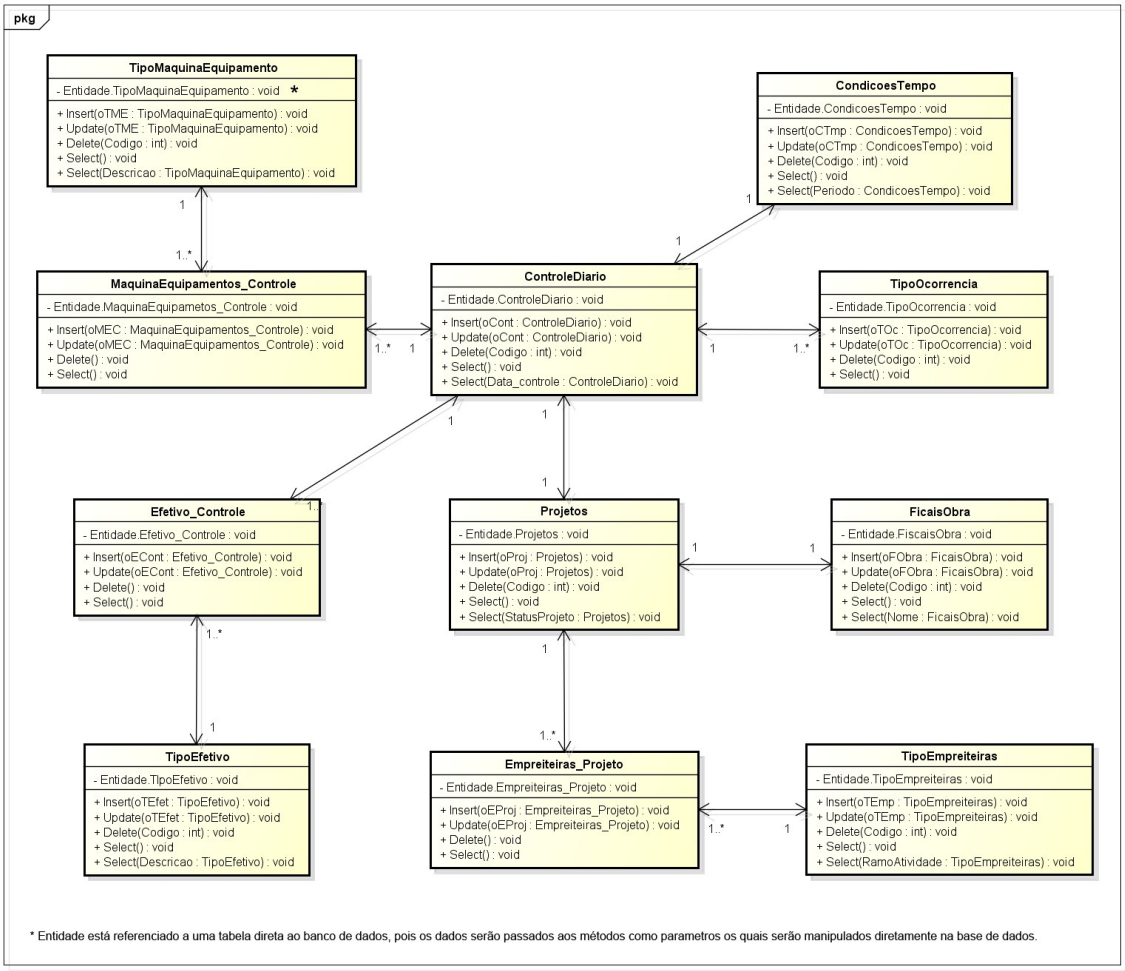


Figura 18 Diagrama de Classes – DAL e BLL

1.16 MODELAGEM DE ENTIDADE E RELACIONAMENTO

Um diagrama Entidade-Relacionamento é um modelo que descreve o modelo de dados de um sistema com alto nível de abstração. 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).

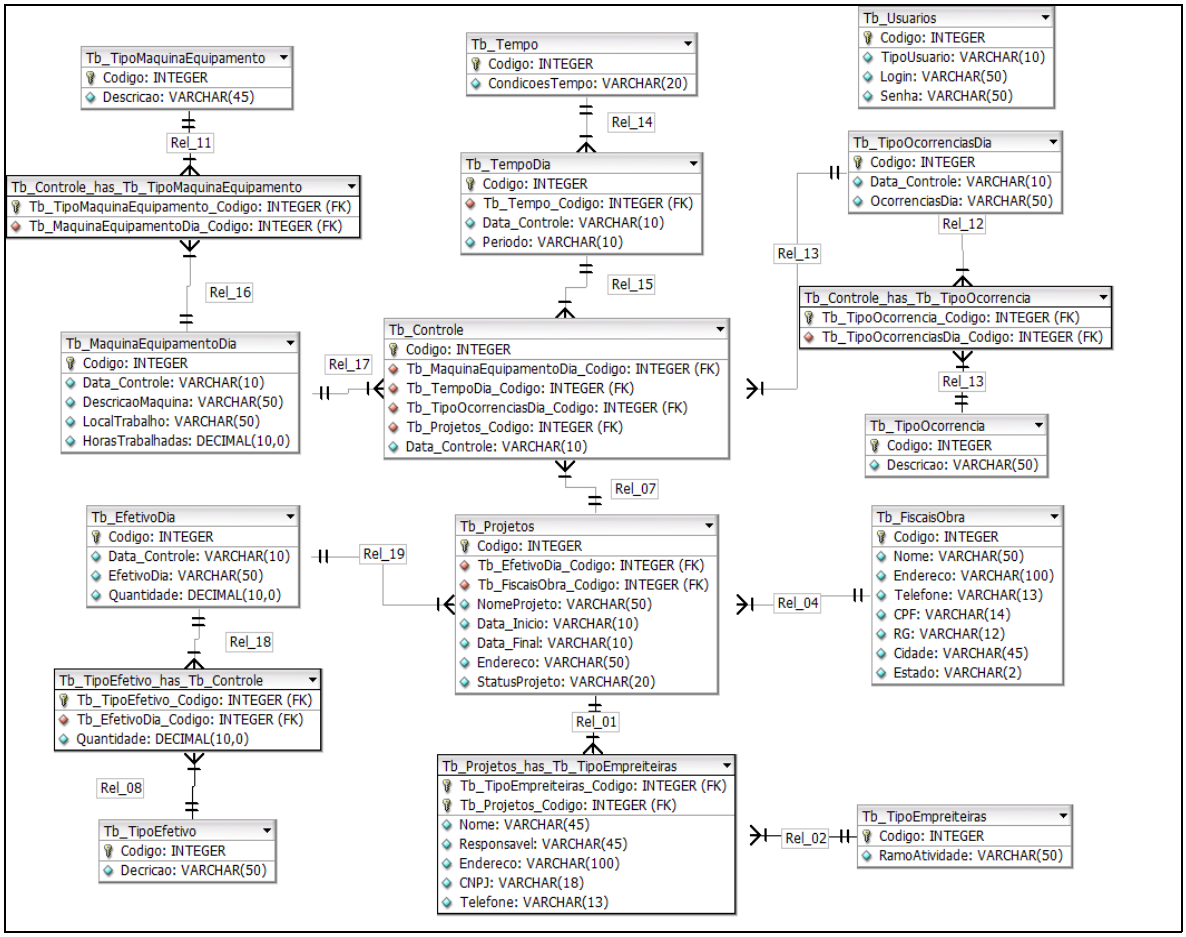


Figura 19 Modelo Entidade-Relacionamento

4 ESTRUTURA DO PROJETO

Neste capítulo será apresentado a metodologia de desenvolvimento utilizada para o trabalho. Tal metodologia consiste em fases e etapas. Para melhor visualizá-las, será apresentado o diagrama da *Work Breakdown Structure*, também conhecido como estrutura analítica de trabalho (WBS), o diagrama de sequenciamento de atividades e o orçamento do sistema.

1.17 ESTRUTURA ANALÍTICA DE TRABALHO

A WBS é uma ferramenta utilizada para subdividir o trabalho de um projeto em partes menores que podem ser gerenciadas com maior facilidade, fornecendo um ilustração detalhada do escopo do projeto, auxiliando na montagem da equipe e distribuição de trabalho, facilitando a identificação de riscos (TAMIRES ALVES DA SILVA, 2011).

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 deste projeto, serão desenvolvidas as seguintes tarefas como mostra a Figura 19, tendo em vista a organização do trabalho e a orientação dos resultados desejados.

Figura 20 Work Breakdown Structure

1.18 SEQUENCIAMENTO DE ATIVIDADES

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

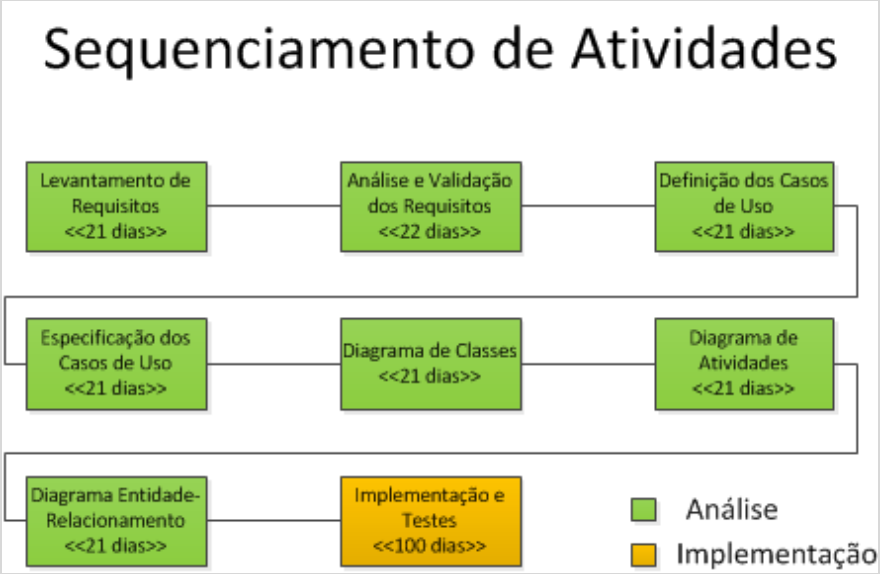


Figura 21 Sequenciamento de Atividades

1.19 ORÇAMENTO

Os recursos necessários para a análise e desenvolvimento de software BuildManager são:

- ✓ 01 Analista – Programador;
- ✓ 01 Impressora Multifuncional;
- ✓ 01 Notebook;
- ✓ Microsoft Visual Studio 2010;

Orçamento BuildManager			
Analista – Programador			
Analista - Programador	**Quantidade de Horas**	**Valor Hora**	**Total**
Matheus Faria Sanches	180	R$ 50.00	R$ 9000.00

Equipamentos			
Equipamento	**Valor**	**Depreciação Diária**	**Total**
Notebook	R$ 2100.00	R$ 5.00	R$ 900.00
Impressora	R$ 500.00	R$ 0.56	R$ 100.00
Microsoft Visual Studio 2010	R$ 1317.00	R$ 1.46	R$ 263.40

Total	**R$ 10263.40**

5 IMPLEMENTAÇÃO DO APLICATIVO

Para a implementação do sistema BuildManager, foi utilizado o ambiente de desenvolvimento *Visual Studio,* juntamente com a linguagem de programação C# e o *framework NHibernate*, no qual foram criados e configurados três projetos, com o objetivo de ganhar independência, reutilizar recursos e diminuir custos com manutenção. A figura 22 mostra o projeto de implementação, o projeto de regras de negócio e o projeto da camada de visualização.

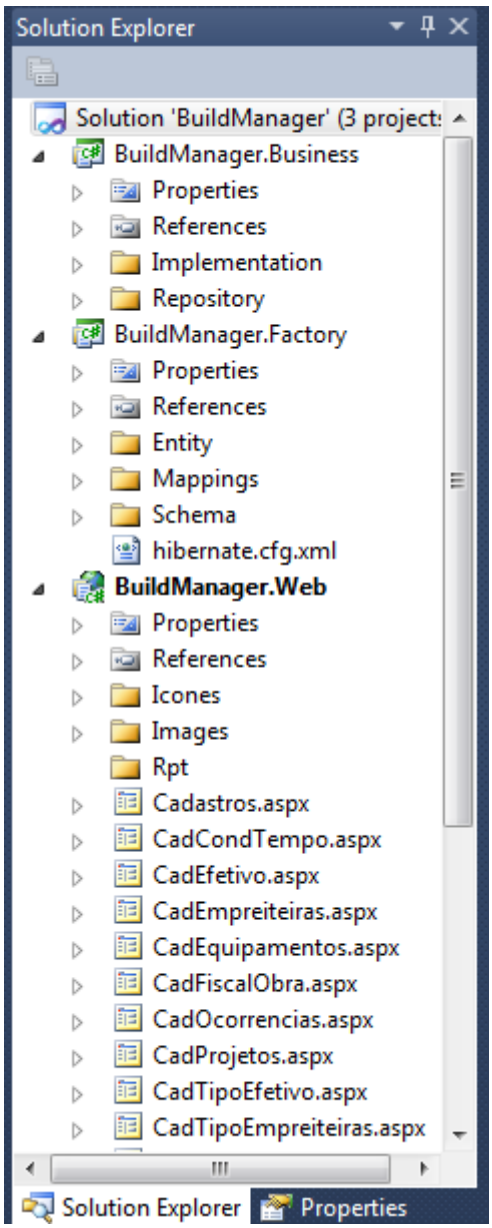


Figura 22 - Projeto de Implementação, Regras de Negócio e Camada de Visualização

- **BuildManager.Business:** Projeto responsável por todas as regras de negócio e acesso a dados, utilizando a linguagem de programação C# e o *framework* de persistência de dados *NHibernate*, além do banco de

dados *Microsoft SQL Server R2*, responsável por armazenar os dados de forma íntegra e segura.

- **BuildManager.Factory:** Projeto responsável por todos os objetos, métodos assessores e modificadores, além de arquivos de mapeamento para cada classe criada.
- **BuildManager.Web:** Projeto responsável pela camada de visualização do sistema, utilizando a linguagem de programação C#.

5.1 ORGANIZAÇÃO DO PROJETO BUILDMANAGER.BUSINESS

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

O projeto “BuildManager.Business” apresenta sua organização dividida em duas pastas: Implementation e Repository, conforme mostra a figura 23, sempre tendo base no padrão de projeto MVC:

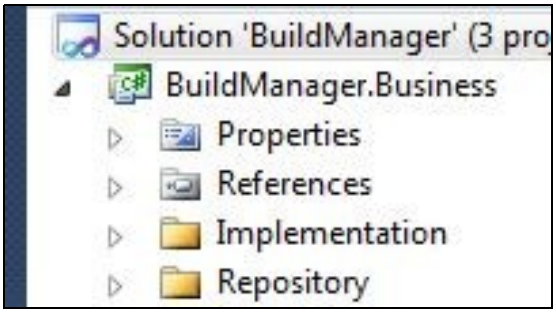


Figura 23 - Organização do Projeto BuildManager.Business

- **Implementation:** Pasta onde se localizam as classes, cuja instância é um objeto responsável por acessar os dados pela camada de persistência, como mostra a figura 24.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using NHibernate;
using NHibernate.Cfg;
using BuildManager.Business.Repository;

namespace BuildManager.Business.Implementation
{
    public class RepositoryBase<T> : IRepositoryBase<T>
    {
        private static ISessionFactory _sessionFactory;

        public static ISessionFactory SessionFactory
        {
            get
            {
                if (_sessionFactory == null)
                {
                    var configuration = new Configuration();
                    configuration.Configure();
                    configuration.AddAssembly(typeof(T).Assembly);
                    _sessionFactory = configuration.BuildSessionFactory();
                }
                return _sessionFactory;
            }
        }

        public void CloseSession()
        {
            if (!_sessionFactory.IsClosed)
            {
                 sessionFactory.Close();
```

Figura 24 - Exemplo de uma classe do pacote Implementation

- **Repository:** Pasta onde se encontra a classe ou objetos controladores das sessões de banco de dados, encarregados pelo controle transacional de informações, como mostra a figura 25.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BuildManager.Business.Repository
{
    public interface IRepositoryBase<T>
    {
        T Salvar(T entity);
        T Alterar(T entity);
        void Excluir(T entity);
        T ObterPorId(int id);
        IList<T> ObterTodos();
    }
}
```

Figura 25 - Exemplo de classe do pacote Repository

5.2 ORGANIZAÇÃO DO PROJETO BUILDMANAGER.FACTORY

Tendo em vista uma melhor organização, o projeto foi dividido duas pastas: Entity e Mappings, como proposto na figura 26:

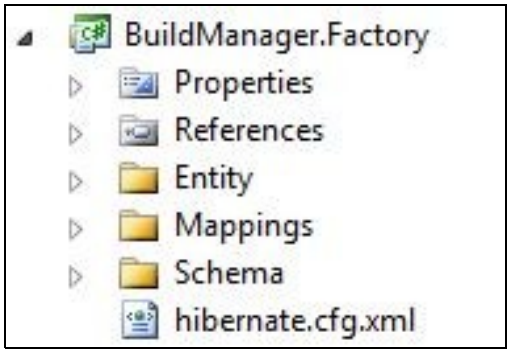


Figura 26 - Organização do Projeto BuildManager.Factory

- **Entity:** Diretório onde se localizam todas as classes de modelo, às quais possuem os métodos construtores e de acesso a valores atribuídos da classe, dos tipos *get* e *set*. A figura 27 apresenta o exemplo de uma classe do pacote Entity.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace BuildManager.Factory.Entity
{
    public class Controle
    {
        private virtual int Codigo { get; set; }
        private virtual string Data_Controle { get; set; }
        private virtual string ResponsavelProjeto { get; set; }
        private virtual int CodProjeto { get; set; }
        private virtual string NomeProjeto { get; set; }
        private virtual int CodEmpreiteira { get; set; }
        private virtual string NomeEmpreiteira { get; set; }
        private virtual string ResponsavelEmpreiteira { get; set; }
    }
}
```

Figura 27 - Exemplo de uma classe do pacote Entity

- **Mappings:** Diretório onde se localizam todos os arquivos XML de mapeamento da camada de persistência de dados. A figura 28 apresenta o exemplo de uma classe referente ao pacote Mapping.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" assembly="BuildManager.Factory" namespace="BuildManager.Factory.Entity">
  <class name="Controle" table="Tb_Controle">
    <id name="Codigo" access="property" column="Codigo" type="Int32">
      <generator class="native">
        <param name="native">controle_seq_id</param>
      </generator>
    </id>
    <property name="DataControle" not-null="true" access="property" type="String">
      <column name="Data_Controle" length="10" />
    </property>
    <property name="ResponsavelProjeto" not-null="true" access="property" type="String">
      <column name="ResponsavelProjeto" length="50" />
    </property>
    <property name="NumeroProjeto" not-null="true" access="property" type="String">
      <column name="Numero_Projeto" length="10" />
    </property>
    <property name="NomeProjeto" node="true" access="property" type="String">
      <column name="NomeProjeto" length="50" />
    </property>
    <property name="NomeEmpreiteira" not-null="true" access="property" type="String">
      <column name="NomeEmpreiteira" length="50" />
    </property>
    <property name="ResponsavelEmpreiteira" not-null="true" access="property" type="String">
      <column name="ResponsavelEmpreiteira" length="50" />
    </property>
  </class>
</hibernate-mapping>
```

Figura 28 - Exemplo de uma classe referente ao pacote Mapping

5.3 ORGANIZAÇÃO DO PROJETO BUILDMANAGER.WEB

A figura 29 mostra a organização do projeto:

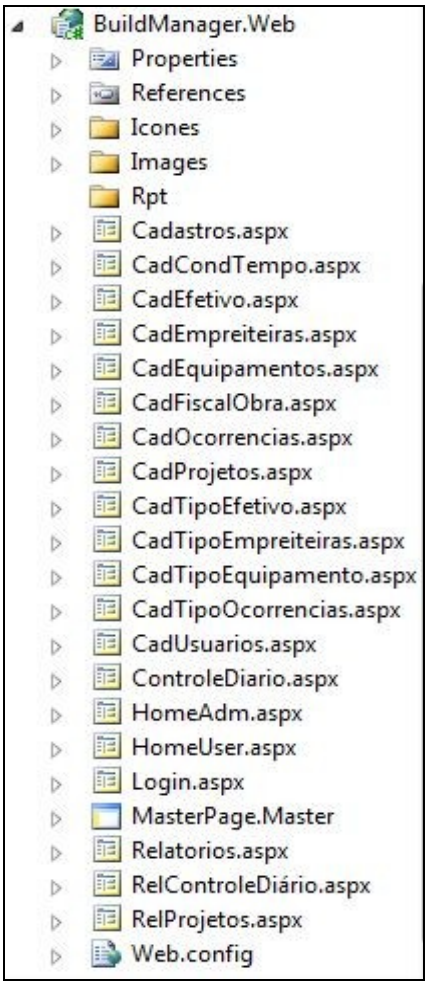


Figura 29 - Organização do Projeto BuildManager.Web

5.4 INTERFACES DO SISTEMA

Ao acessar o sistema, a página inicial abrirá, local onde é efetuado o login do usuário cadastrado no sistema. A figura 30 mostra a página do login do sistema.

Figura 30 - Página para Login

O sistema terá a disponibilidade de cadastro de dois tipos de usuários, sendo eles:

- **Admin:** Usuário com permissões especiais. Este usuário poderá realizar o cadastro de todas as opções presentes no Controle Diário, como: *Empreiteiras, Condições do Tempo, Projetos*, entre outros. Também terá acesso a relatórios referentes a todos os projetos: *Empreiteiras, Controle Diário,* entre outros. A figura 31 mostra a página inicial referente ao usuário do tipo *Admin*.

- **User:** Usuário com permissões simples, podendo apenas realizar o *Controle Dário de Atividades* e gerar o relatório referente ao controle cadastrado no dia. A figura 32 mostra a página inicial referente ao usuário do tipo *User*.

Figura 31 - Página Inicial para Usuário do tipo Admin

Figura 32 - Página Inicial para Usuário do tipo User

Ao selecionar a opção *Cadastros*, é apresentado um submenu onde se encontram os cadastros de projetos, fiscais de obra, ocorrências, efetivos, tipo de efetivos, máquinas e equipamentos, tipo de máquinas e equipamentos, entre outros.

Ao selecionar a opção *Controle Diário*, abrirá uma página para o cadastro do controle diário da obra.

Ao selecionar a opção *Relatórios*, abrirá um submenu contendo os relatórios que podem ser gerados a partir das informações cadastradas.

Quando o usuário desejar cadastrar uma empreiteira, deverá escolher a opção *Empreiteiras* no menu de cadastros. Tal opção apresenta ao usuário um formulário de cadastro de *Empreiteiras*. Ressalta-se que apenas os usuários do tipo *Admin* tem acesso ao menu de cadastros. A figura 33 mostra o formulário de cadastro de *Empreiteiras*, contendo as opções: Novo, Salvar, Limpar e também uma lista de Empreiteiras já cadastradas no sistema *BuildManager*.

Figura 33 - Formulário para Cadastro de Empreiteiras

Quando o usuário desejar fazer o *Controle Diário* da obra, deverá escolher a opção *Controle Diário*, localizado na página inicial. Após escolher a opção, aparecerá um formulário para o cadastro do *Controle Diário* da obra, como mostra a figura 34.

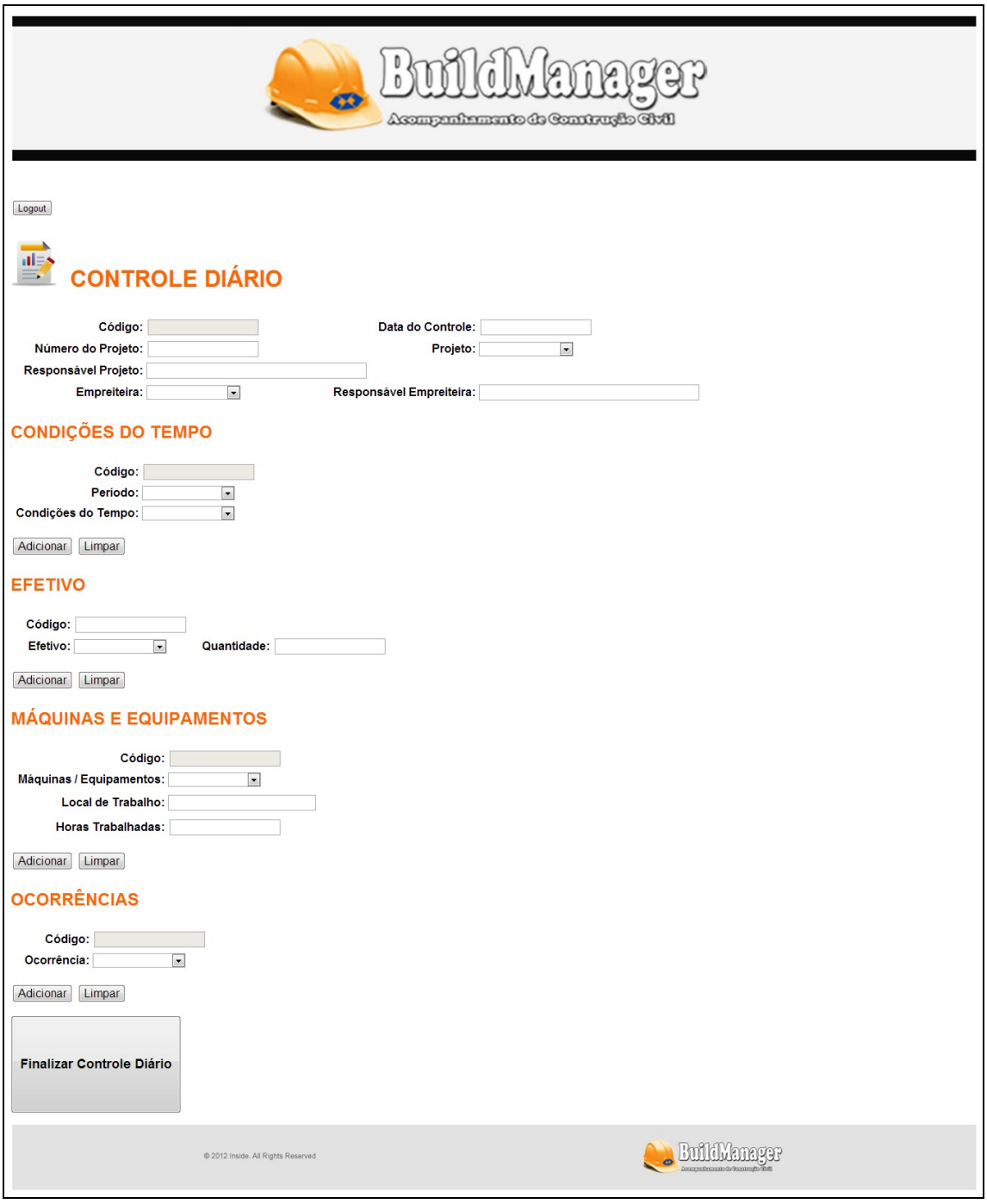


Figura 34 - Controle Diário

6 CONCLUSÃO

A área de tecnologia da informação tem crescido muito no Brasil e no mundo, e o uso de softwares específicos é o fator que tem contribuído para esse crescimento, devido a necessidade de um maior controle e levantamento de processo de produção.

O software *BuildManager* foi desenvolvido para o mercado de gerenciamento de construções civis, agilizando processos e reduzindo gastos. Tudo realizado de forma flexível, tendo em vista uma melhor facilidade no controle das tarefas antes e durante a construção, sempre satisfazendo as necessidades dos clientes.

Através dos diagramas UML elaborados, pode ser obtido uma visão completa e detalhada sobre o funcionamento do sistema, contribuindo para um melhor entendimento das funcionalidades do sistema.

Com a utilização da construção em camadas, a implementação do sistema possibilitou a separação das regras de negócio das regras de persistência, isolando a camada de visualização para ser responsável apenas pela interação com o usuário, o que facilita a manutenção do sistema.

Futuramente serão realizados alguns *packages* de atualização para o sistema, que ainda estão em fase de planejamento. Um dos *packages*, apresentará um sistema de BI para o controle de estoque de materiais de cada projeto, facilitando o controle de matéria prima utilizada na construção e diminuindo custos para a Organização que fará uso do sistema *BuildManager.*

REFERÊNCIAS

ALEXANDRE, Claudio**; Introduction to the C# Language and the .NET Framework.** Disponível em:
< http://msdn.microsoft.com/library/z1zx9t92 > Acesso em: 30 de Maio de 2012.

ARAÚJO, Everton Coimbra de; **Java e C#.NET - Um breve e introdutório estudo comparativo de suas sintaxes e convenções.** Disponível em:
< http://www.linhadecodigo.com.br/artigo/1620/java-e-csharpnetum-breve-.aspx> Acesso em: 30 de Maio de 2012.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **UML Guia do Usuário**. 2° 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.** 2° Edição. Tradução de Vera Pezerico e Christian Thomas. Porto Alegre: Bokkman, 2000.

Crystal Reports. Disponível em:
< http://www.crystalreports.com/ > Acesso em: 31 de Maio de 2012

EDUARDO; **Padrão MVC**. Disponível em:
<http://tiideia.blogspot.com.br/2011_10_01_archive.html> Acesso em: 01 de Junho de 2012.

HAMILTON, Naomi; **The A-Z of Programming Languages: C#.** Disponível em: < http://www.computerworld.com.au/article/261958/a-z_programming_languages_c_/?pp=7 > Acesso em: 28 de Maio de 2012.

INTERNATIONAL, Ecma; **ECMA – 334 C# Language Specification**. 4° Edição. Geneva: Ecma International, 2006. Disponível em:
< http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf> Acesso em: 28 de Maio de 2012.

MACORATTI; **Diagrama de Classes.** Disponível em:
< http://www.macoratti.net/net_uml1.htm > Acesso em: 06 de Junho de 2012.

MACORATTI; **Diagrama de Sequências .** Disponível em:
< http://www.macoratti.net/vb_uml2.htm > Acesso em: 06 de Junho de 2012.

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

MAULO, Fabio; **NHibernate.** Disponível em:
< https://community.jboss.org/wiki/NHibernateForNET?_sscc=t >
Acesso em: 28 de Maio de 2012.

MICROSOFT; **Microsoft SQL Server 2008 R2.** Disponível em:
< http://www.microsoft.com/sqlserver/pt/br/default.aspx > Acesso em: 12 de Junho de 2012.

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

PORTAL EDUCAÇÃO; **SQL Server 2008**. Disponível em:
< http://www.portaleducacao.com.br/informatica/artigos/6138/sql-server-2008/pagina-1 > Acesso em: 12 de Junho de 2012.

SILVA, Tamires Alves da; **Padrão de projeto MVC.** Trabalho de Conclusão de Curso. Assis: 2011.

TRÍBOLI, Edison Paulo De Ros; **Mapas Mentais: uma introdução.** Disponível em: < http://pt.scribd.com/doc/904729/Mapas-mentais > Acesso em: 31 de Março de 2012.

ANEXO I

CRONOGRAMA DE DESENVOLVIMENTO DO TRABALHO

A seguir é apresentado o cronograma das atividades realizadas durante a execução do projeto:

Id	Nome da tarefa	Duração	Início	Término
1	Levantamento de Requisitos	21 dias	Qua 01/02/12	Qua 29/02/12
2	Análise e Validação de Requisitos	22 dias	Qui 01/03/12	Sex 30/03/12
3	Definição do Caso de Uso	21 dias	Seg 02/04/12	Seg 30/04/12
4	Especificação do Caso de Uso	21 dias	Seg 02/04/12	Seg 30/04/12
5	Diagrama de Classes	21 dias	Seg 02/04/12	Seg 30/04/12
6	Diagrama de Atividades	21 dias	Seg 02/04/12	Seg 30/04/12
7	Diagrama Entidade Relacionamento	21 dias	Seg 02/04/12	Seg 30/04/12
8	Elaboração da Qualificação	65 dias	Ter 01/05/12	Seg 30/07/12
9	Exame da Qualificação	5 dias	Ter 31/07/12	Seg 06/08/12
10	Implementação e Testes	100 dias	Seg 02/07/12	Sex 16/11/12
11	Escrita Versão Final	75 dias	Seg 06/08/12	Sex 16/11/12
12	Apresentação Final	10 dias	Seg 19/11/12	Sex 30/11/12

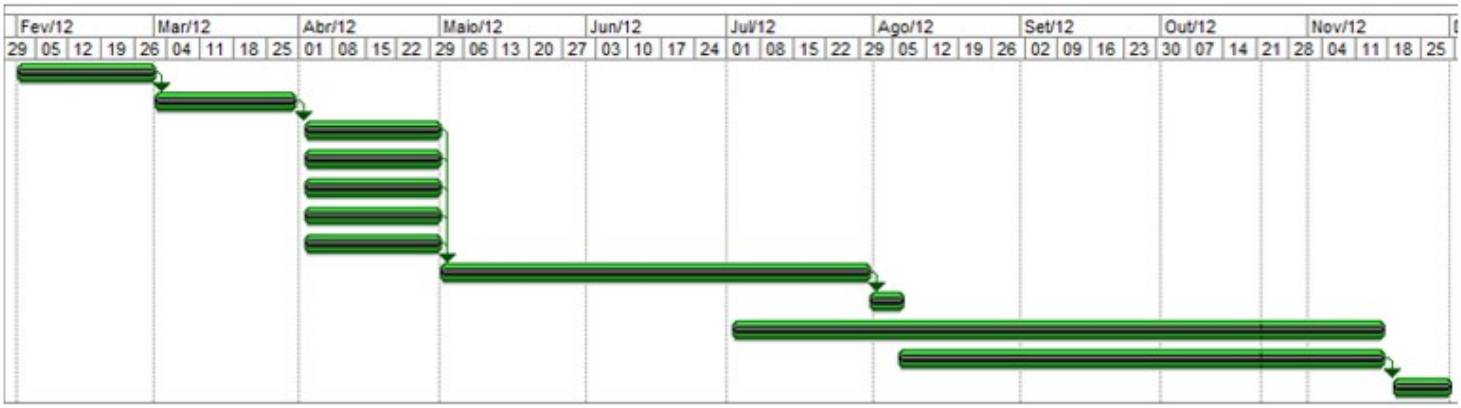


Figura 35 Cronograma Atualizado