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

MURILLO GONÇALVES ZAMPIERI

SISTEMA WEB PARA CONTROLE DE EGRESSOS

ASSIS-SP

2014

MURILLO GONÇALVES ZAMPIERI

SISTEMA WEB PARA CONTROLE DE EGRESSOS

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

Orientador: Dr° Almir Rogério Camolesi

Área de Concentração: Desenvolvimento de Sistemas

ASSIS

2014

FICHA CATALOGRÁFICA

ZAMPIERI, Murillo Gonçalves

Sistema Web Para Controle de Egressos / Murillo Gonçalves Zampieri. Fundação Educacional do Município de Assis – FEMA - Assis, 2014.

P 65.

Orientador: Dr. Almir Rogério Camolesi

Trabalho de Conclusão de Curso

Instituto Municipal de Ensino Superior de Assis – IMESA

1. Controle de Egressos 2. Sistema Web 3. Plataforma .NET 4. Linguagem de Programação C#.

CDD: 001.61

Biblioteca da FEMA

SISTEMA WEB PARA CONTROLE DE EGRESSOS

MURILLO GONÇALVES ZAMPIERI

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

Orientador: Dr° Almir Rogério Camolesi

Analisador: Me. Fabio Eder Cardoso

ASSIS

2014

DEDICATÓRIA

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

AGRADECIMENTOS

Primeiramente a Deus, pois sem ele nada seria possível. Dele que vem tudo o que sou, tenho e espero.

A minha esposa Suelen Soares Fragoso Zampieri pelo seu amor, companheirismo e compreensão em todos os momentos, deixando os meus dias mais felizes e belos.

Aos meus familiares, Uraci Zampieri, Neide Gonçalves de Souza Zampieri, por estarem sempre ao meu lado, apoiando-me e conduzindo-me durante minha caminhada.

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

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

“Torna-te quem tu és.”

Friedrich Nietzsche

(1844-1900)

RESUMO

Atualmente aqueles que atendem aos requisitos exigidos pelo mercado de trabalho e acompanham as evoluções da tecnologia, estão em um processo sem fim de busca de informação e atualização.

Neste trabalho serão apresentadas a especificação e a realização de um portal Web, onde seu público alvo são aqueles que estão no meio acadêmico. Dessa forma proporcionando ao egresso que ele se mantenha atualizado nas necessidades e novidades da sua área de formação, através da ajuda daqueles que os acompanharam na sua formação.

Para o desenvolvimento deste trabalho será realizado um estudo sobre tecnologias a serem utilizadas para a realização do sistema, como a linguagem de modelagem UML, linguagem de programação C# e para o banco de dados Microsoft SQL Server.

Palavras-Chave: Sistema Web; Plataforma .NET; Linguagem de Programação C#.

ABSTRACT

Currently those who meet the requirements demanded by the labor market and keep pace with developments in technology, are in an endless process of information search and update.

In this paper the specification and development of a web portal, where your target audience are those who are in academia will be presented. Thus providing the egress that it can keep up on the needs and news from your area of training through the help of those who accompanied them in their training.

To develop this work a study of technologies to be used for the realization of the system as the modeling language UML, C # programming language and the Microsoft SQL Server database will be performed.

Keywords: Web System; NET platform; C # Programming Language.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Atualmente a sociedade vive em um momento onde a troca de informações é frequente, a atualização do que está acontecendo ao seu redor e a busca constante de conhecimentos são os principais requisitos de quem quer entrar e/ou se manter no mercado de trabalho.

O ambiente onde mais se pode encontrar pessoas interessadas nesta mesma busca de desenvolvimento, sem dúvidas é no meio acadêmico. Após a conclusão do ensino superior e o ingresso no mercado de trabalho, como continuar aprendendo sem ter que abrir mão da família, amigos e lazer? Para isso existe um grande aliado, a Internet.

A velocidade das mudanças e o desenvolvimento tecnológico transformam constantemente o ambiente e a forma de trabalho, não há dúvidas que o “estudo” e “formação” não são apenas uma etapa da vida, mas uma constante ao longo de toda a carreira (COLLETTO, 2005).

O desenvolvimento científico e tecnológico na sociedade vem causando transformações constantes nos ambientes de trabalho e consequentemente, exigindo um profissional capaz de adaptar-se às mudanças e motivado a continuar aprendendo ao longo do tempo. Neste contexto, o aparecimento de recursos interativos e de bases de formação contribui para aumentar a propagação do conhecimento e superar a relação tempo e espaço, oferecendo tanto oportunidades para construção e acesso ao conhecimento, como possibilitando interações individuais e coletivas de forma integrada e permanente (GUIMARÃES; GODOY, 2014).

Nos dias atuais, a maioria das pessoas recorrem a tecnologia para se relacionar com outras pessoas, fazer cursos e até ver o que está acontecendo do outro lado do mundo. Por isso nada melhor que uma página na web para permitir o relacionamento das pessoas que estão em busca dos mesmos objetivos.

O portal Web para controle de egressos proposto neste trabalho é ideal para as pessoas que estão em busca de constante atualização das novidades da área de

formação e que querem manter o contato com aqueles que acompanharam sua formação.

O portal apresenta como principais funcionalidades: interfaces para cadastro de instituições, cursos, matérias, professores e alunos.

Os recursos disponibilizados pelo portal proporcionam ao egresso, facilidades que contribuem para o sucesso de seu desenvolvimento intelectual.

1.1 OBJETIVO

Como objetivo principal, o sistema Portal para Controle de Egressos foi desenvolvido para o gerenciamento das informações de forma dinâmica e colaborativa, onde os egressos estarão trocando informações de novidades tecnológicas e novos métodos criados, com seus antigos companheiros de estudo e ainda podendo contar com o auxilio de seus antigos professores e pessoas que estão vivenciando as novidades no dia-a-dia.

1.2 JUSTIFICATIVA

A justificativa para que o Portal Web para Controle de Egresso ser um sistema Web se dá pelo enorme crescimento no uso de tais meios de comunicação, dessa forma proporcionando enormes vantagens devido à facilidade de acesso as informações.

As pessoas estão em busca de sistemas seguros, modernos e precisos para auxiliar na tomada de decisões, dessa forma melhorando a qualidade de seus serviços e também o seu ganho intelectual, portanto, o Sistema Web para Controle de Egressos fará uso da linguagem de programação C#.

Contudo, o desenvolvimento do Sistema Web para Controle de Egressos atenderá as exigências descritas e acompanhará as evoluções tecnológicas com finalidade de competir com este nicho de mercado. Além de o presente trabalho contribuir para com meu enriquecimento acadêmico e profissional.

1.3 PÚBLICO ALVO

O sistema desenvolvido atende as necessidades de pessoas que desejam continuar se atualizando com as novidades dá área de formação, que não querem perder o contato com sua turma de formação e também para os professores e instituições que desejam ajudar mais na formação de seus antigos alunos.

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, casos de uso e suas especificações e os principais diagramas UML (classe e atividade).

O quarto 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 último capítulo serão apresentados a conclusão do trabalho, organograma atualizado e trabalhos futuros.

2 TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

Neste capítulo foram relatadas as ferramentas e tecnologias utilizadas para o desenvolvimento do portal web para controle de egressos.

Para o desenvolvimento deste trabalho, foi necessária a modelagem do Sistema, para essa tarefa foi utilizada a linguagem UML (DA SILVA; VIDEIRA, 2001) 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 até sistemas complexos.

O portal web para controle de egressos é um sistema web desenvolvido com a tecnologia Microsoft.NET e a linguagem de programação C# (GREENE; STELLMAN, 2008), que possui uma linguagem de alto nível orientada a objetos. O ambiente de desenvolvimento foi baseado no IDE (Integrated Development Environment) Visual Studio 2012 (LOUREIRO, 2014).

Como base de dados, foi utilizado o banco de dados SQL Server 2012.

2.1 UML

A UML surgiu da união de três metodologias de modelagem: o método de Boock, o método *Object Modeling Technique* (OMT) de Jacobson e o método *Object-Oriented Software Engineering* (OOSE) de Rumbaugh. Essas eram, até meados da década de 1990, as três metodologias de modelagem orientada a objetos mais populares entre os profissionais da área de engenharia de software. A união dessas metodologias contou que o amplo apoio da Rational Software, que incentivou e financiou tal União (DA SILVA; VIDEIRA, 2001).

A UML é a especificação mais conhecido do *Object Management Group* (OMG) e é a norma de indústria da informática para descrever graficamente “software”. A UML é uma linguagem ou notação visual para especificação (modelagem) de sistemas de informação orientados a objetos. Ela não apresenta um processo fixo para o desenvolvimento de software, ou seja, não é uma metodologia de sistemas, é apenas uma notação e pode ser usada com várias metodologias. A UML possui

diversos mecanismos de extensão que permitem que ela possa ser utilizada em vários domínios diferentes. (GOES, 2014).

Deve ficar bem claro, no entanto, que a UML não é uma linguagem de programação, mas uma linguagem de modelagem, cujo objetivo é auxiliar os engenheiros de software a definir as características do software, tais como seus requisitos, seu comportamento, sua estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema deverá ser implantado. Todas essas características são definidas por meio da UML antes de o software começar a ser realmente desenvolvido. (GOES, 2014).

2.1.1. Casos de Uso

O primeiro elemento da UML é o caso de uso. O caso de uso descreve as principais funcionalidades do sistema e a iteração dessas funcionalidades com os usuários do mesmo sistema. (RIBEIRO, 2014).

2.1.2. 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).

2.1.3. Diagrama de Classe

Os diagramas de classes são responsáveis por ilustrar os atributos e operações de uma classe e as restrições e como os objetos devem ser conectados. (MACORATTI, 2014).

2.1.4. Modelagem de Entidade e Relacionamento

Um diagrama Entidade-Relacionamento é uma modelagem que descreve o modelo de dados de um sistema. Foi desenvolvido para facilitar o projeto de banco de

dados, onde é responsável por representar a estrutura lógica geral do bando de dados. (REZENDE, 2005).

2.1.5. Work Breakdown Structure

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

2.1.6. Sequenciamento de atividades

O diagrama de Sequencia da á visão geral do tempo de duração para a realização de cada atividade.

2.2 C#

A plataforma .NET de desenvolvimento da Microsoft tem como foco principalmente o desenvolvimento de serviços WEB XML (LIMA; REIS, 2014).

A proposta de Microsoft .NET é de ser uma plataforma de desenvolvimento, que envolva linguagens de programação, compiladores, modelo de objetos etc., englobando de uma forma completamente integrada todos esses requisitos (LIMA; REIS, 2014).

Abaixo algumas característica de .NET:

- Independência de linguagem de programação: o que permite a implementação do mecanismo de herança, controle de exceções e depuração entre linguagens de programação diferentes.
- Reutilização de código legado: o que implica em reaproveitamento de código escrito usando outras tecnologias como COM, COM+, ATL, DLLs e outras bibliotecas existentes.

- Tempo de execução compartilhado: o *runtime* de .NET é compartilhado entre as diversas linguagens que a suportam, o que quer dizer que não existe um *runtime* diferente para cada linguagem que implementa .NET.
- Sistemas autoexplicativos e controle de versões: cada peça de código .NET contém em si mesma a informação necessária e suficiente de forma que o ambiente de execução - *runtime* não precise procurar no registro do Windows mais informações sobre o programa que está sendo executando. O *runtime* encontra essas informações no próprio sistema em questão e sabe qual a versão a ser executados, sem acusar aqueles velhos conflitos de incompatibilidade ao registrar bibliotecas (DLLs) no Windows.
- Simplicidade na resolução de problemas complexos. (LIMA; REIS, 2014).

O C# é uma linguagem orientada a objetos criada em 2000 junto da plataforma. NET, que permite aos seus desenvolvedores construírem uma variedade de aplicações seguras e robustas. (ALEXANDRE, 2014).

A linguagem C# foi influenciada por várias linguagens, como por exemplo, JAVA e C++. Na verdade, ela é uma junção das principais vantagens dentre essas linguagens, melhorando suas implementações e adicionando recursos, fazendo a linguagem atrativa para desenvolvedores que queiram migrar para o Microsoft .NET. (ARAÚJO, 2014).

Vários desenvolvedores participaram do projeto de criação da linguagem, mas o principal envolvido no projeto foi o engenheiro Anders Hejlsberg, que além do C# foi criador do Turbo Pascal e do Delphi. (ARAÚJO, 2014).

A Microsoft define o C# como a principal linguagem de programação para uso da tecnologia .NET. Por ser uma derivação da linguagem C++, sem as suas limitações, e é uma linguagem bastante simples de se implementar. (VAMBERTO, 2014).

2.3 MICROSOFT SQL SERVER

O SQL Server é um SGBD (sistema gerenciador de banco de dados) da Microsoft, criado em parceria com a Sybase, em 1988, inicialmente como um complementar do

Windows NT, sendo que depois passou a ser aperfeiçoado e vendido separadamente. A parceria coma Sybase terminou em 1994, e a Microsoft continuou a melhorar o programa após isto. (PACIEVITCH, 2014).

O SQL Server 2012 é uma plataforma de informações pronta para a nuvem, projetada para organizações que procuram proteger, utilizar e aumentar o poder de seus dados na estação de trabalho, em dispositivos portáteis, no datacenter e na nuvem pública ou privada. (MICROSOFT, 2014).

2.4 PADRÃO DE PROJETO MVC

Um padrão de projeto tem como função nomear, abstrair e identificar aspectos-chaves de uma estrutura de projeto comum, tornando-a útil para a criação de um projeto orientado a objetos de maneira reutilizável (DA SILVA; VIDEIRA, 2001).

O padrão de arquitetura Model-View-Controller (MVC) é uma forma de quebrar uma aplicação em três camadas: o modelo, a visão e o controlador (MEDEIROS, 2014).

- **Modelo ou *Model*:** Utilizado para manipular informações de forma mais detalhada, sendo recomendado que, sempre que possível, se utilize dos modelos para realizar consultas, cálculos e todas as regras de negócio do nosso site ou sistema. É o modelo que tem acesso a toda e qualquer informação sendo essa vinda de um banco de dados, arquivo XML (BASTOS, 2014).
- **Visão ou *View*:** Responsável por tudo que o usuário final visualiza toda a interface, informação, não importando sua fonte de origem (BASTOS, 2014).
- **Controlador ou *Controller*:** Responsável por executar uma regra de negócio (*model*) e repassa a visualização para a visualização (*view)* (BASTOS, 2014).

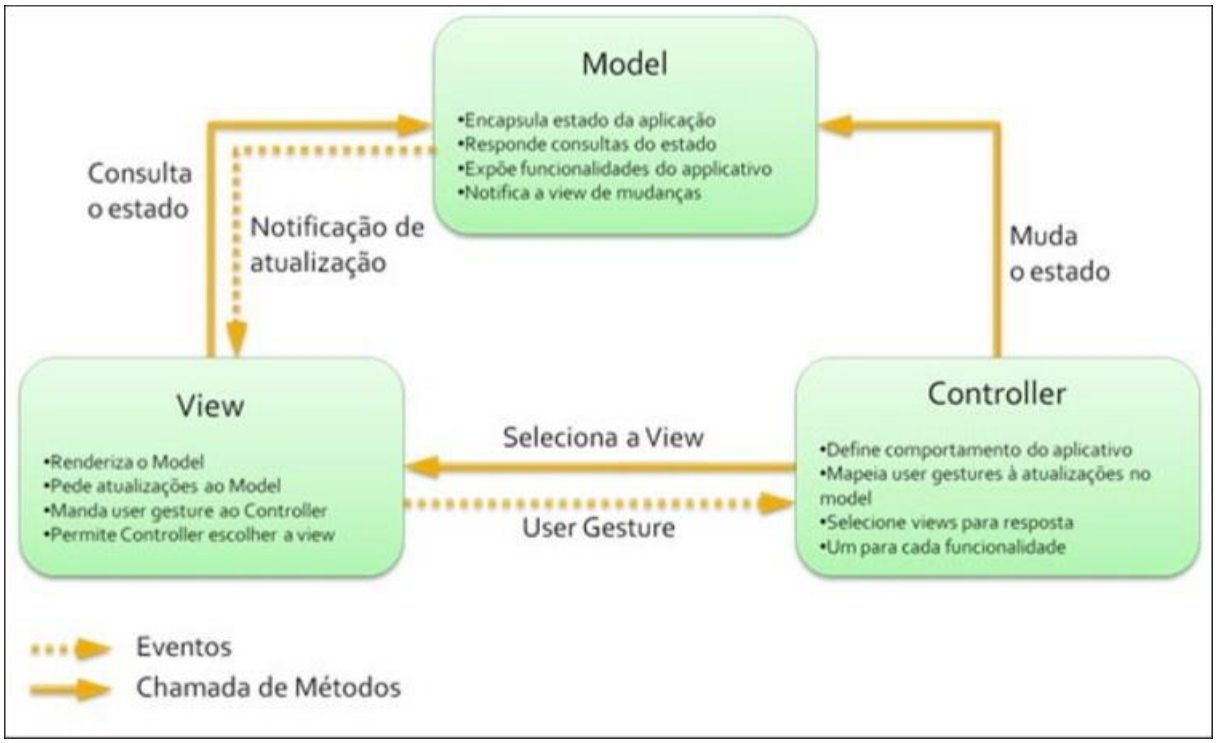

Figura 1 - Funcionamento do Padrão de Projetos MVC Fonte: Blog do Moura[1]

O padrão de projeto MVC é uma forma de desenvolvimento que ajuda na manutenção do sistema, sendo um padrão muito aceito durante implementação de aplicações web (SANCHES, 2012).

2.5 MAPA MENTAL

Para melhorar o entendimento e visualização do sistema é apresentado o mapa mental, que é um método utilizado para armazenar, organizar e priorizar informações através de diagramas elaboradas a partir de uma ideia principal. Segundo Tony Buzan, considerado o inventor do mapa mental, afirma que os mapas mentais são desenhados como neurônios e projetados para estimular o cérebro (BUZAN, 2014).

[1] http://www.blogomoura.com/2011/07/entendendo-o-padrao-de-projeto-mvc/

3 ANÁLISE, ESPECIFICAÇÃO E PROJETO DO SISTEMA

Este capítulo apresenta as especificações, modelagem e análise do sistema.

3.1. MAPA MENTAL

A Figura 2 ilustra o mapa mental referente às atividades internadas do sistema.

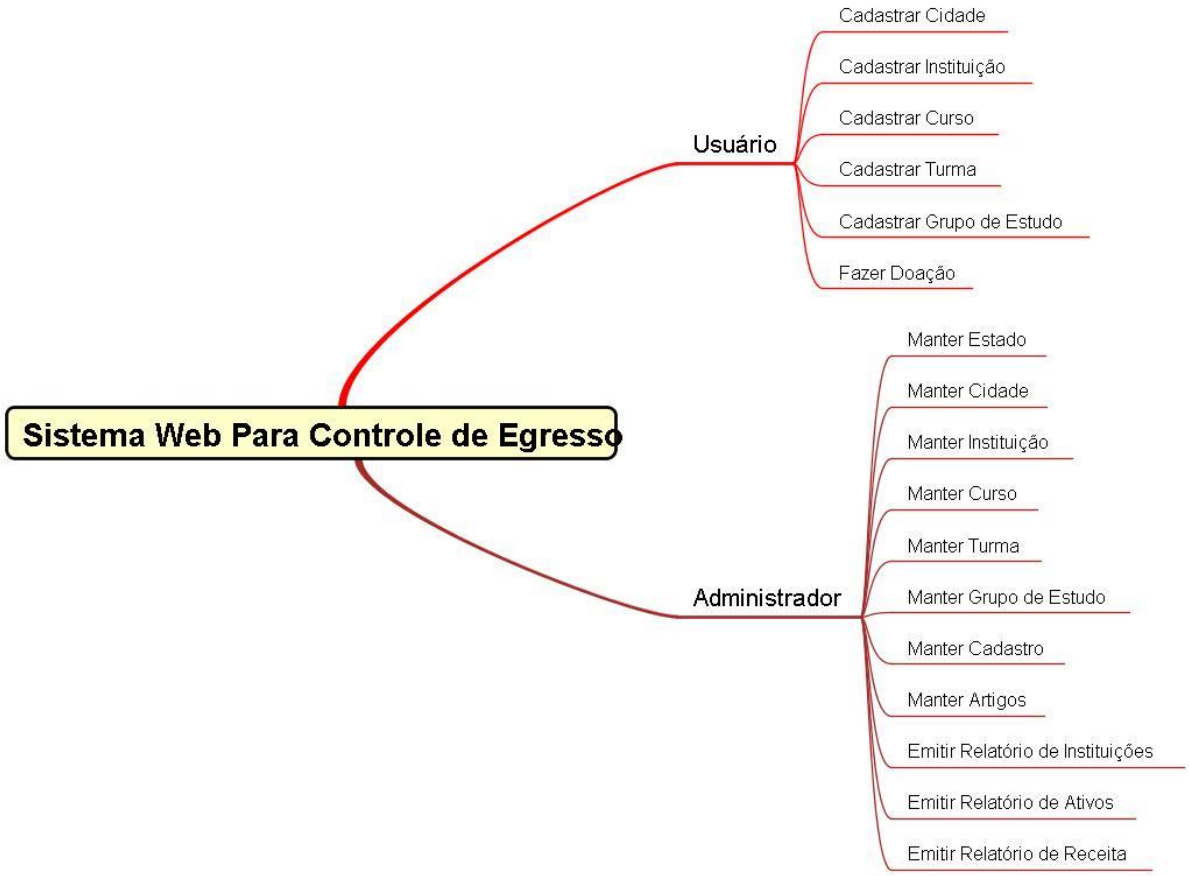

Figura 2 - Mapa Menta do Sistema Web para Controle de Egresso

3.2 LISTA DE EVENTOS

A seguir são descritos os principais eventos do sistema:

1. Efetuar Controle de Acesso
2. Manter Estado

3. Manter Cidade
4. Manter Instituição
5. Manter Curso
6. Manter Turma
7. Manter Grupo de Estudo
8. Manter Usuário
9. Manter Material
10. Cadastrar Estado
11. Cadastrar Cidade
12. Cadastrar Instituição
13. Cadastrar Curso
14. Cadastrar Turma
15. Cadastrar Grupo de Estudo
16. Disponibilizar Material
17. Cadastrar Usuário
18. Fazer Doação
19. Emitir Relatório de Instituições
20. Emitir Relatório de Ativos
21. Emitir Relatório de Receita

3.3 CASOS DE USO

A Figura 3 mostra o caso de uso geral do sistema:

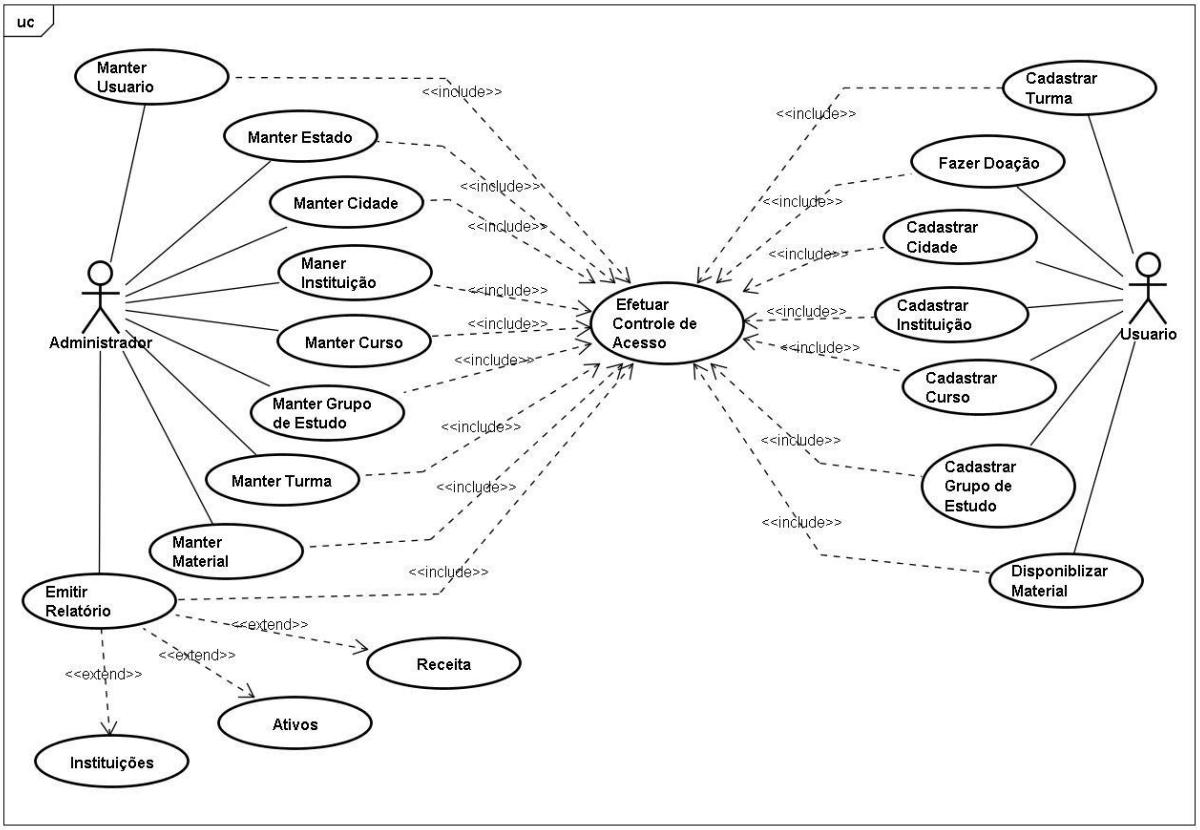

Figura 3 - Diagrama de Caso de Uso Geral do Sistema

A Figura 4 mostra o caso de uso geral do administrador

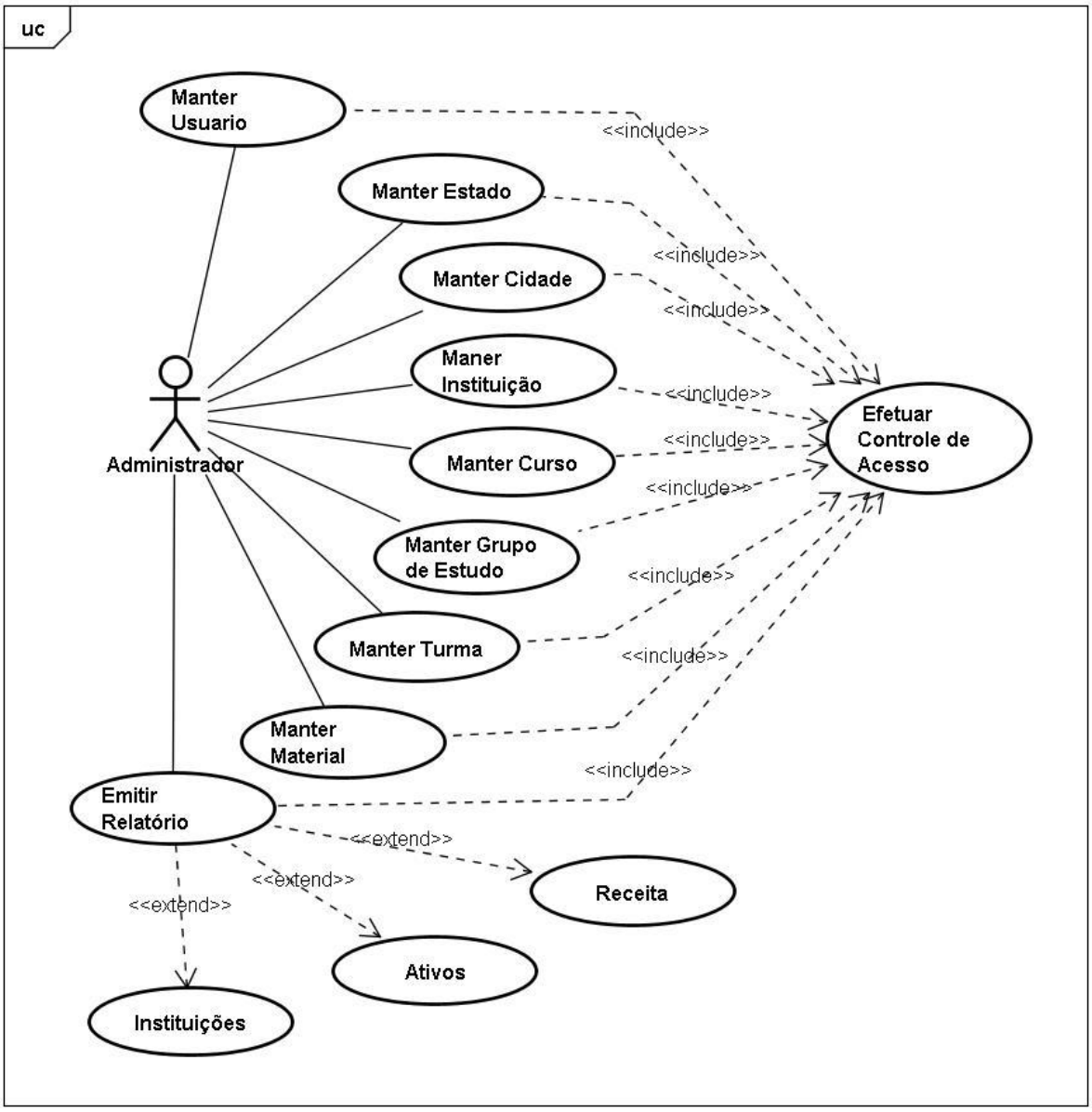

Figura 4 - Diagrama de Caso de Uso Geral do Administrador

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

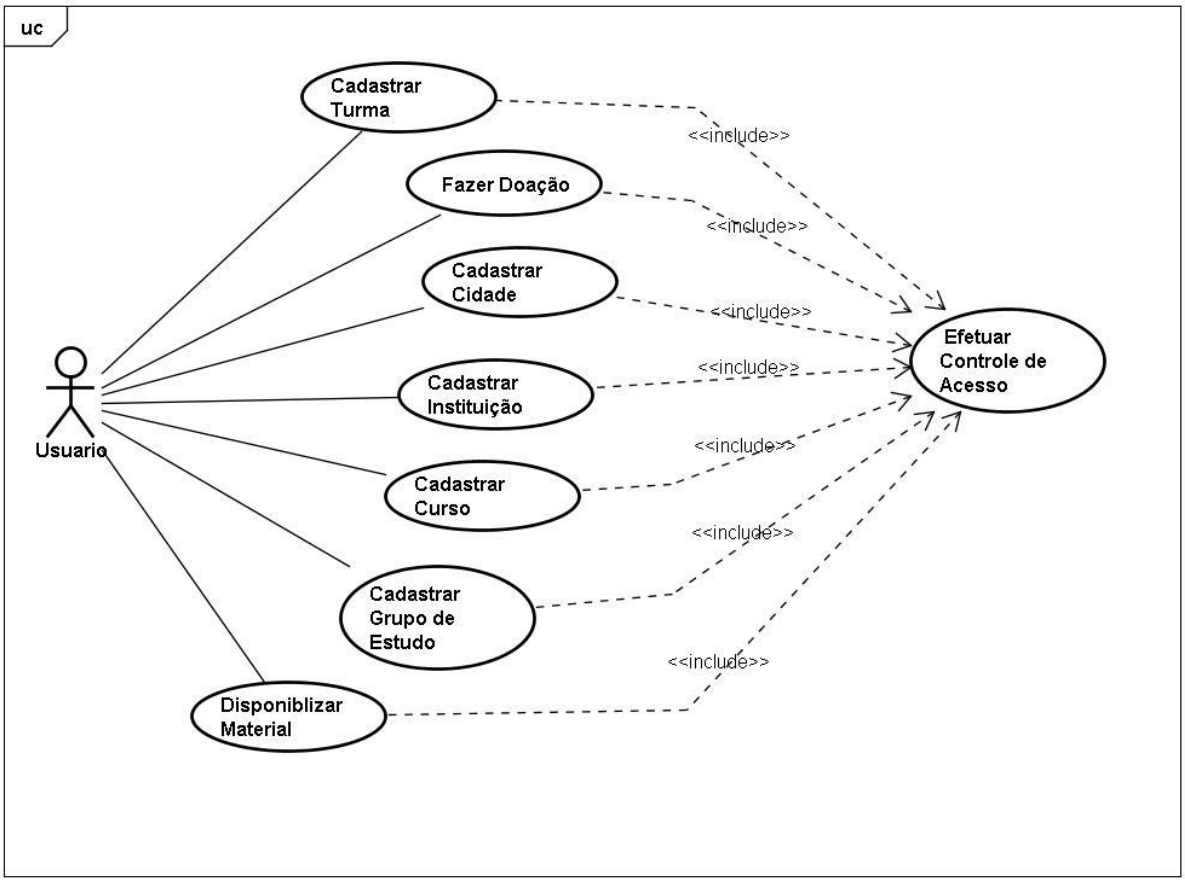

Figura 5 - Diagrama de Caso de Uso Geral do Usuário

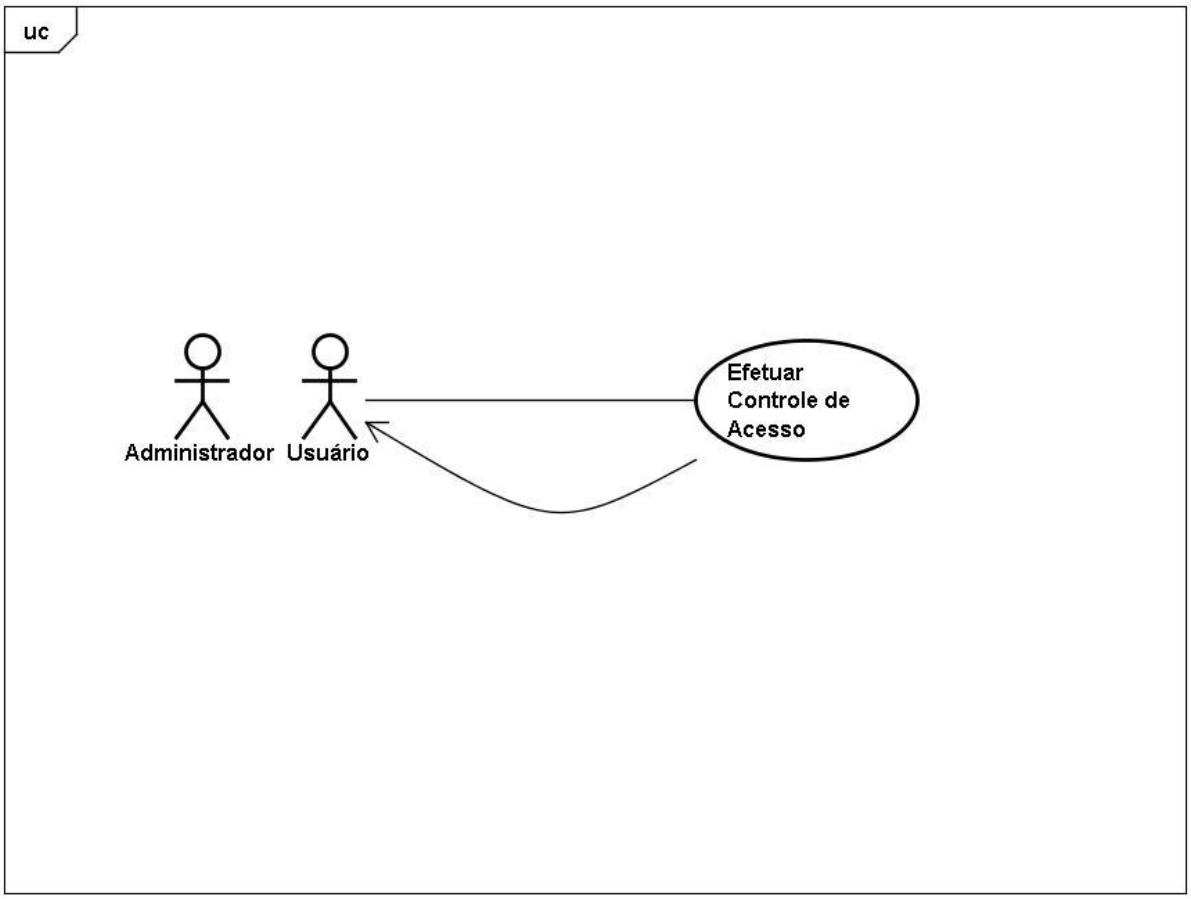

Figura 6 - Diagrama de Caso de Uso Efetuar Controle de Acesso

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 os dados 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 Caso os dados estejam corretos executa operação solicitada. 4.2- Caso os dados estejam incorretos cancela a operação e exibe mensagem de alerta.

Tabela 1 - Especificação Efetuar Controle de Acesso

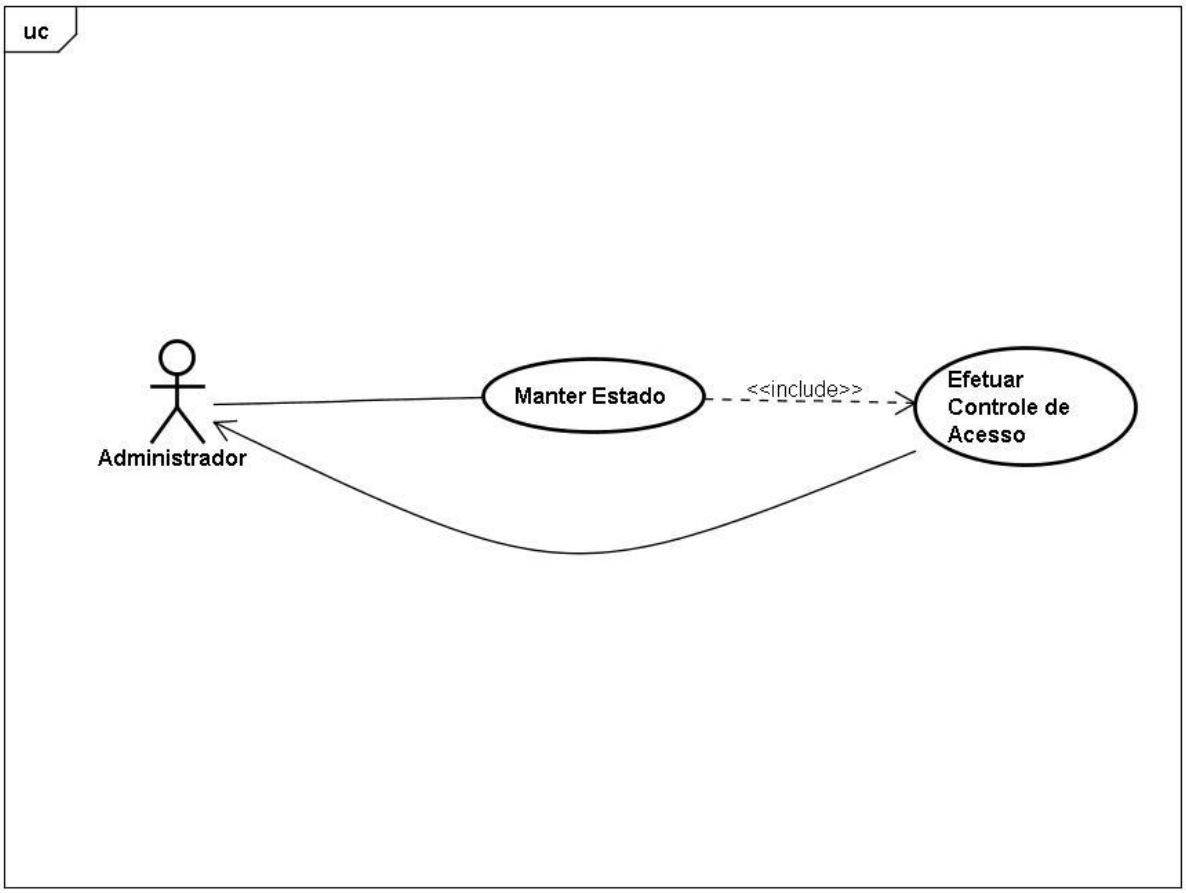

Figura 7 - Diagrama de Caso de Uso Manter Estado

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

Tabela 2 - Especificação Manter Estado

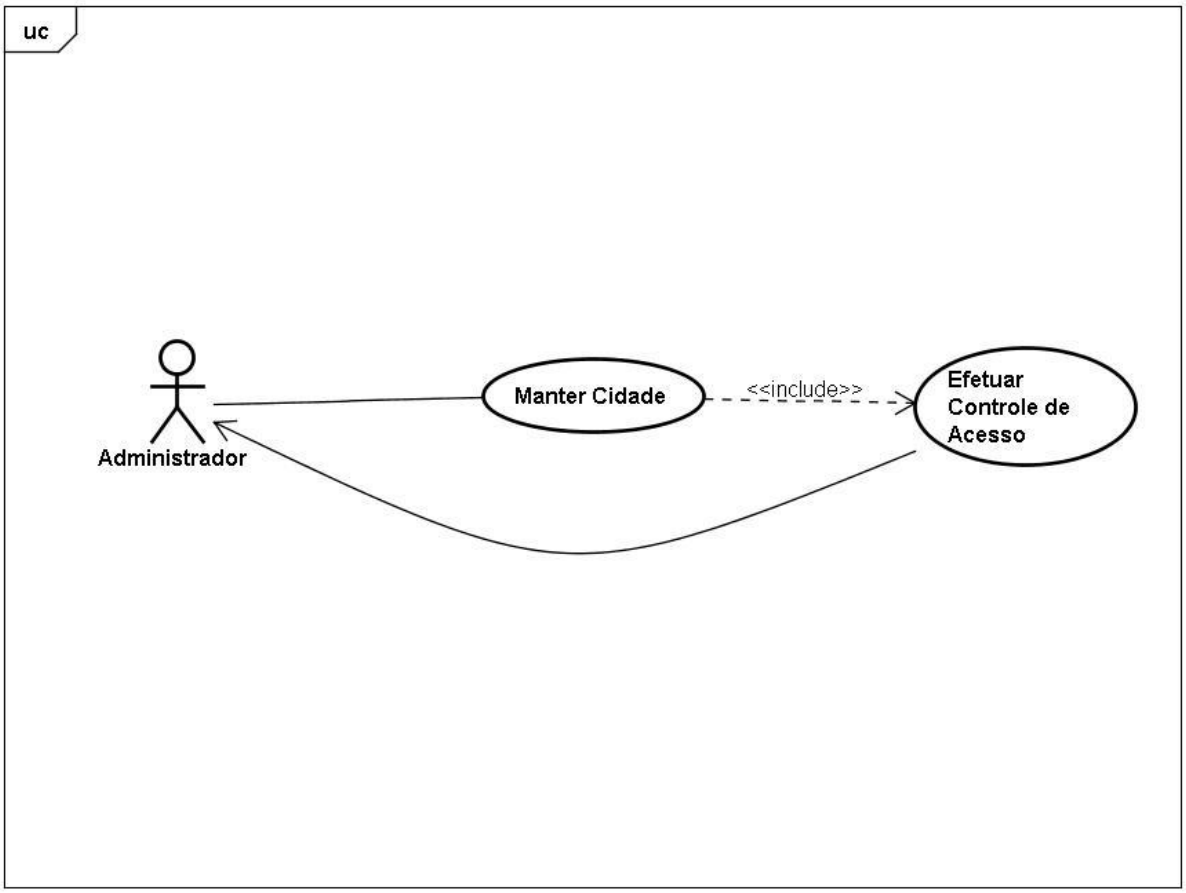

Figura 8 - Diagrama de Caso de Uso Manter Cidade

Nome do caso de uso 3	Manter Cidade
Atores	Administrador
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1- O administrador informa os dados da Cidade. 2. O sistema informa um estado (Caso de uso Manter Estado). 3. O sistema valida os dados informados. 4. O administrator seleciona o botão "cadastrar". 5. O sistema emite mensagem de sucesso. 6. O sistema cadastra a Cidade
Cenários Alternativos	Caso o sistema não valide os dados informados pelo administrador exibirá mensagem de alerta. O administrador poderá cancelar o processo durante o cadastro
Casos de Teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 1.3 O sistema cancela a operação

Tabela 3 - Especificação Manter Cidade

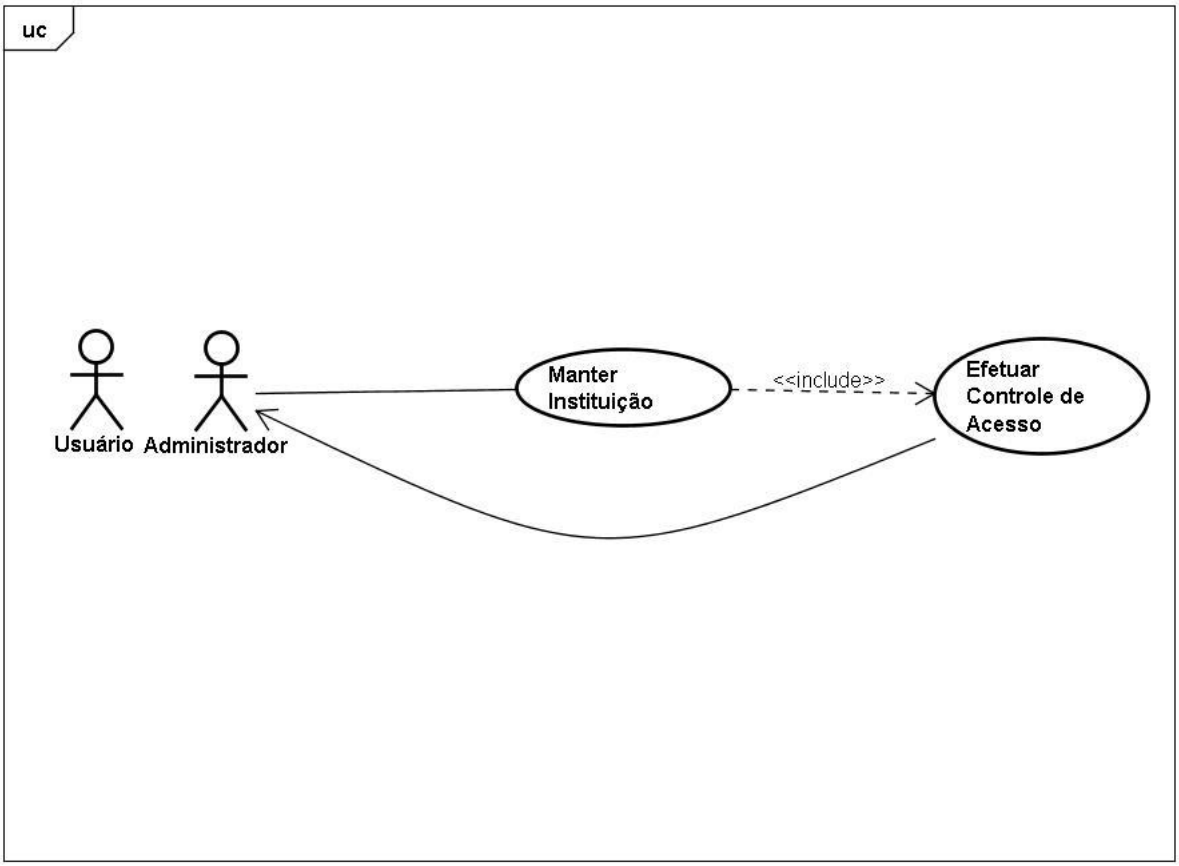

Figura 9 - Diagrama de Caso de Uso Manter Instituição

Nome do caso de uso 4	Manter Instituição
Atores	Administrador e Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1. O Administrador ou Usuário informa os dados da Instituição. 2. O Administrador ou Usuário informa um Estado (Caso de Uso Manter Estado). 3. O Administrador ou Usuário informa uma cidade (Caso de Uso Manter Cidade). 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 a instituição
Cenários Alternativos	Caso o sistema não valide os dados informados pelo Administrador ou Usuário exibirá mensagem de alerta. O Administrador ou Usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 2.1 O sistema verifica se o Administrador ou Usuário informou um estado. 2.2 O sistema emite uma mensagem de alerta. 3.1 O sistema verifica se o Administrador ou Usuário

	informou uma cidade. 3.2 O sistema emite uma mensagem de alerta.

Tabela 4 - Especificação Manter Instituição

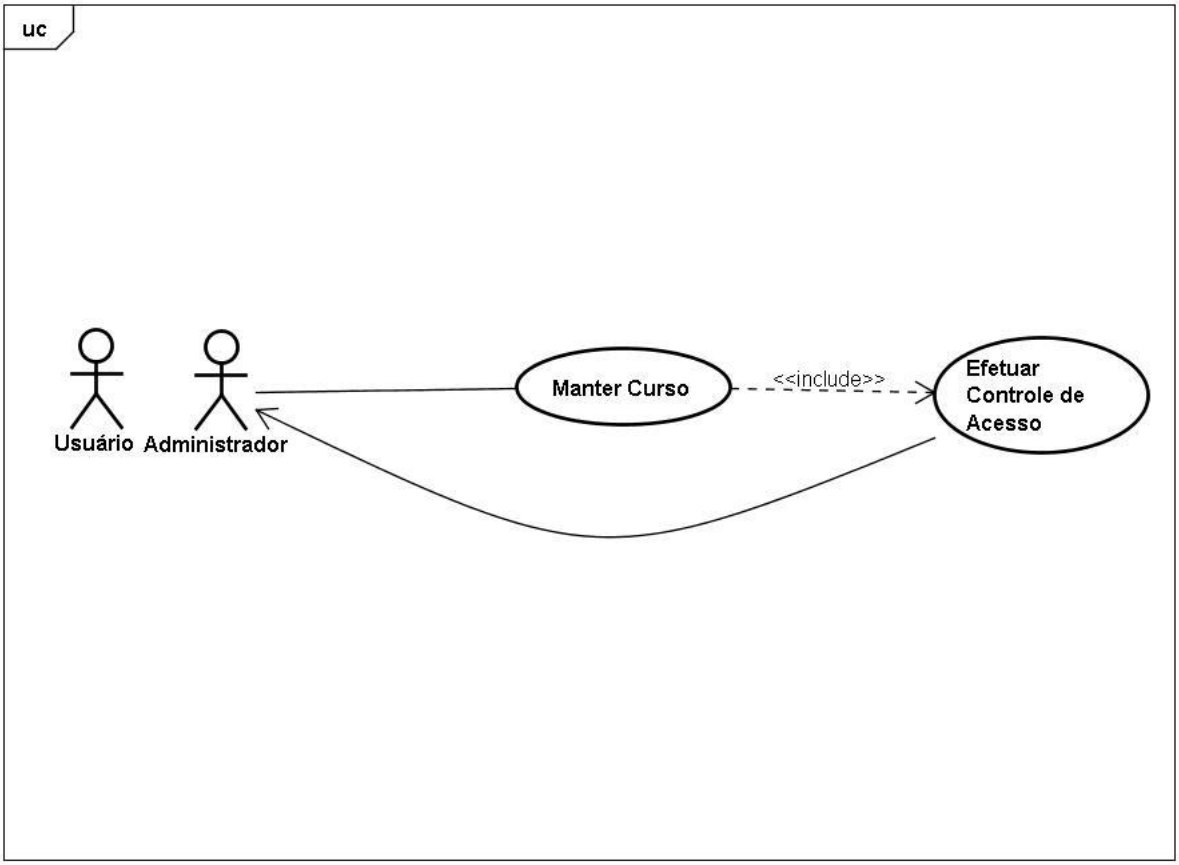

Figura 10 - Diagrama de Caso de Uso Manter Curso

Nome do caso de uso 5	Manter Curso
Atores	Administrador e Usuário
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1. O Administrador ou Usuário informa os dados do curso. 2. O Administrador ou Usuário informa uma instituição (Caso de Uso Manter Instituição). 3. O sistema valida os dados informados. 4. O Administrador ou Usuário seleciona a opção "cadastrar". 5. O sistema emite mensagem de sucesso. 6. O sistema cadastra o curso
Cenários Alternativos	Caso o sistema não valide os dados informados pelo Administrador ou Usuário exibirá mensagem de alerta. O Administrador ou Usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 2.1 O sistema verifica se o Administrador ou Usuário informou uma instituição. 2.2 O sistema emite uma mensagem de alerta.

Tabela 5 - Especificação Manter Curso

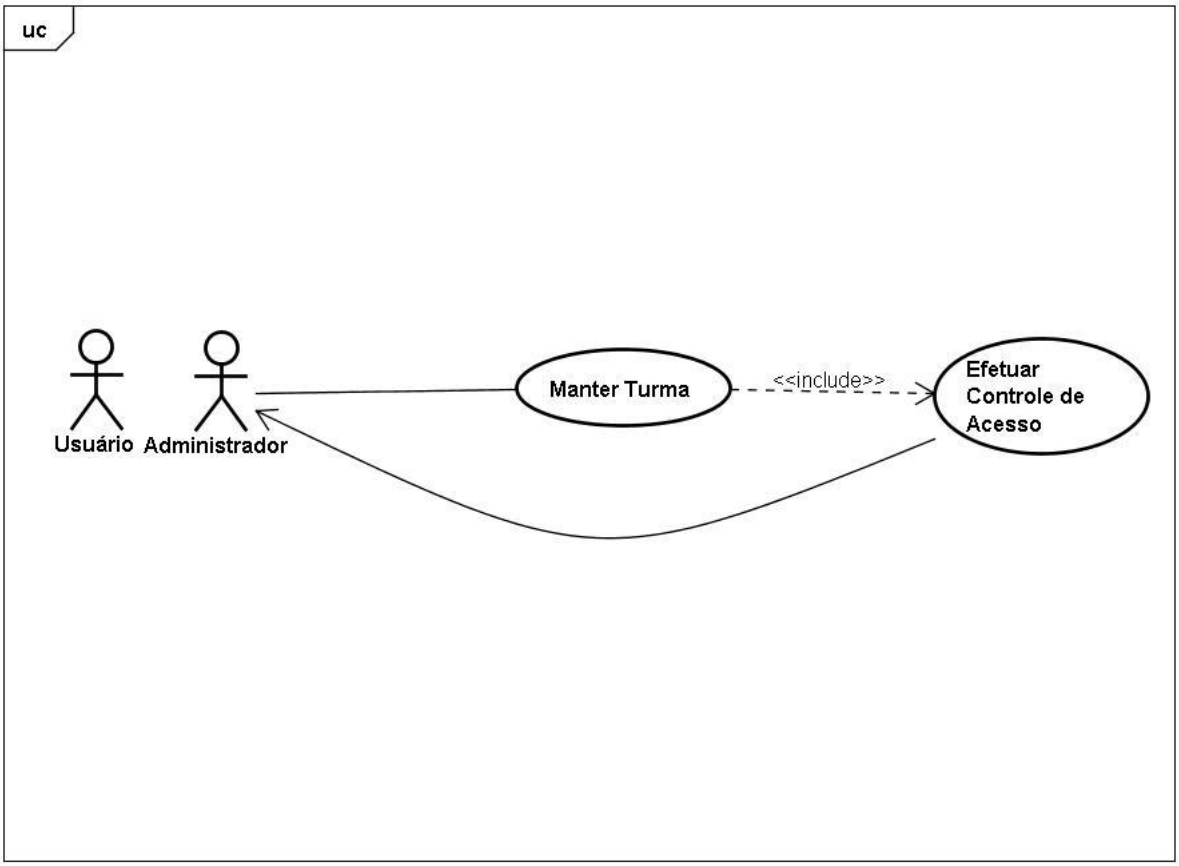

Figura 11 - Diagrama de Caso de Uso Manter Turma

Nome do caso de uso 6	Manter Turma
Atores	Administrador e Usuário.
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1. O Administrador ou Usuário informa os dados da turma. 2. O Administrador ou Usuário informa uma instituição (Caso de Uso Manter Instituição). 3. O Administrador ou Usuário informa um curso (Caso de Uso Manter Curso). 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 curso
Cenários Alternativos	Caso o sistema não valide os dados informados pelo Administrador ou Usuário exibirá mensagem de alerta. O Administrador ou Usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 2.1 O sistema verifica se o Administrador ou Usuário informou uma instituição. 2.2 O sistema emite uma mensagem de alerta. 3.1 O sistema verifica se o Administrador ou Usuário

	informou um curso. 3.2 O sistema emite uma mensagem de alerta.

Tabela 6 - Especificação Manter Turma

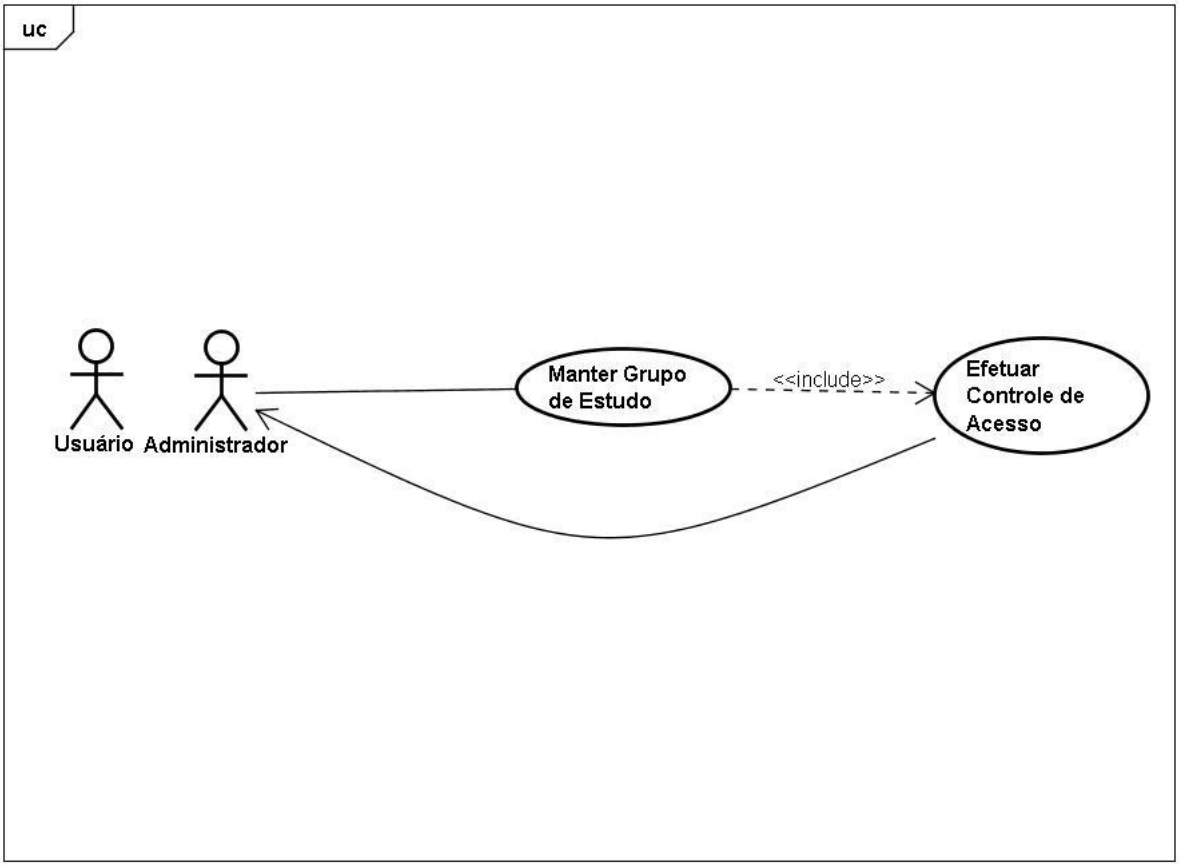

Figura 12 - Diagrama de Caso de Uso Manter Grupo de Estudo

Nome do caso de uso 7	Manter Grupo de Estudo
Atores	Administrador e Usuários.
Pré-condição	Efetuar Controle de Acesso
Cenário principal	1. O Administrador ou Usuário informar o nome do Grupo de Estudo 2. O Administrador ou Usuário informa uma instituição (Caso de Uso Manter Instituição). 3. O Administrador ou Usuário informa um curso (Caso de Uso Manter Curso). 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 curso
Cenários Alternativos	Caso o sistema não valide os dados informados pelo Administrador ou Usuário exibirá mensagem de alerta. O Administrador ou Usuário poderá cancelar o processo durante o cadastro.
Casos de Teste	1.1 O sistema verifica se os campos foram preenchidos corretamente. 1.2 O sistema não confirma o cadastro e emite uma mensagem de erro. 2.1 O sistema verifica se o Administrador, aluno ou professor informou uma instituição. 2.2 O sistema emite uma mensagem de alerta.

	3.1 O sistema verifica se o Administrador ou Usuário informou um curso. 3.2 O sistema emite uma mensagem de alerta.

Tabela 7 - Especificação Manter Grupo de Estudo

3.4 DIAGRAMA DE ATIVIDADES

A Figura 13 ilustra o diagrama de atividade do método Cadastro de Instituição:

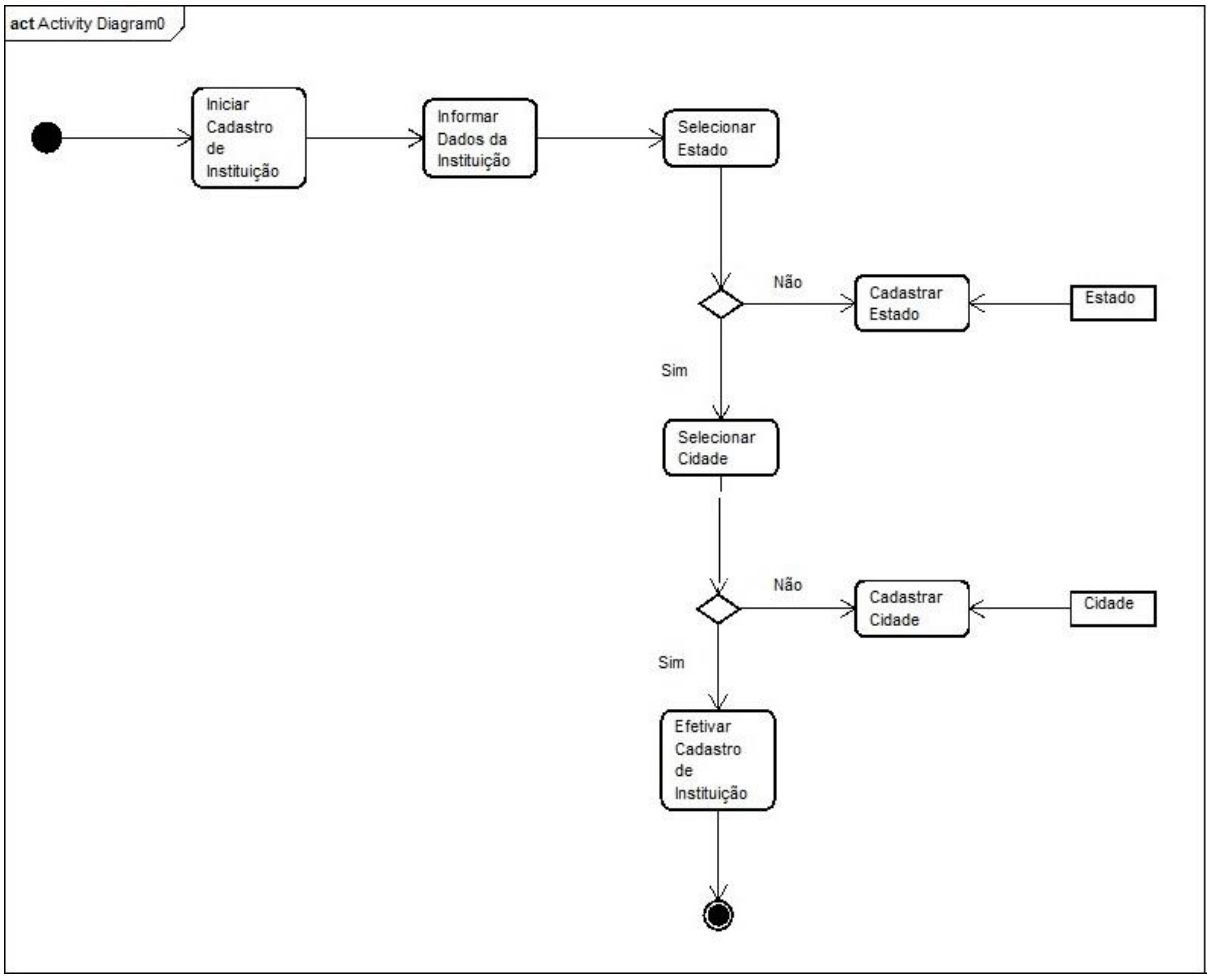

Figura 13 - Diagrama de Atividade Cadastro de Instituição

A Figura 14 ilustra o diagrama de atividade do evento Fazer Doação:

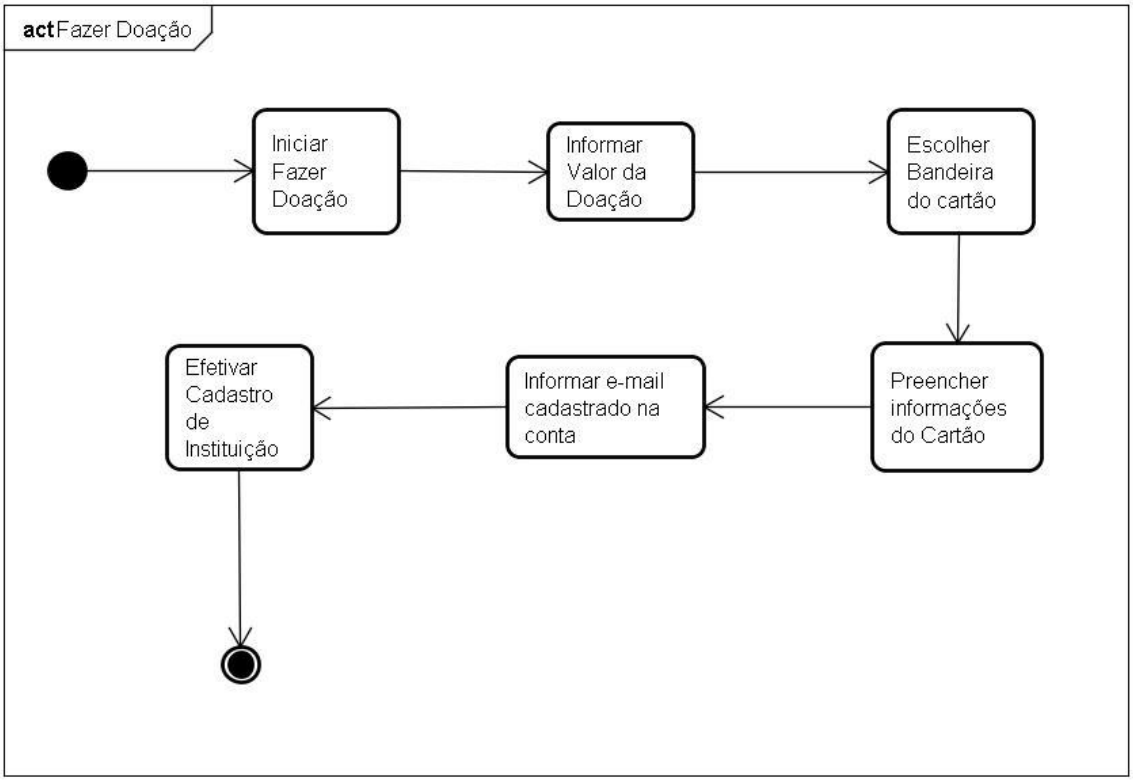

Figura 14 - Diagrama de Atividade Fazer Doação

3.5 DIAGRAMAS DE CLASSES

A Figura 15 ilustra do diagrama de classes do sistema:

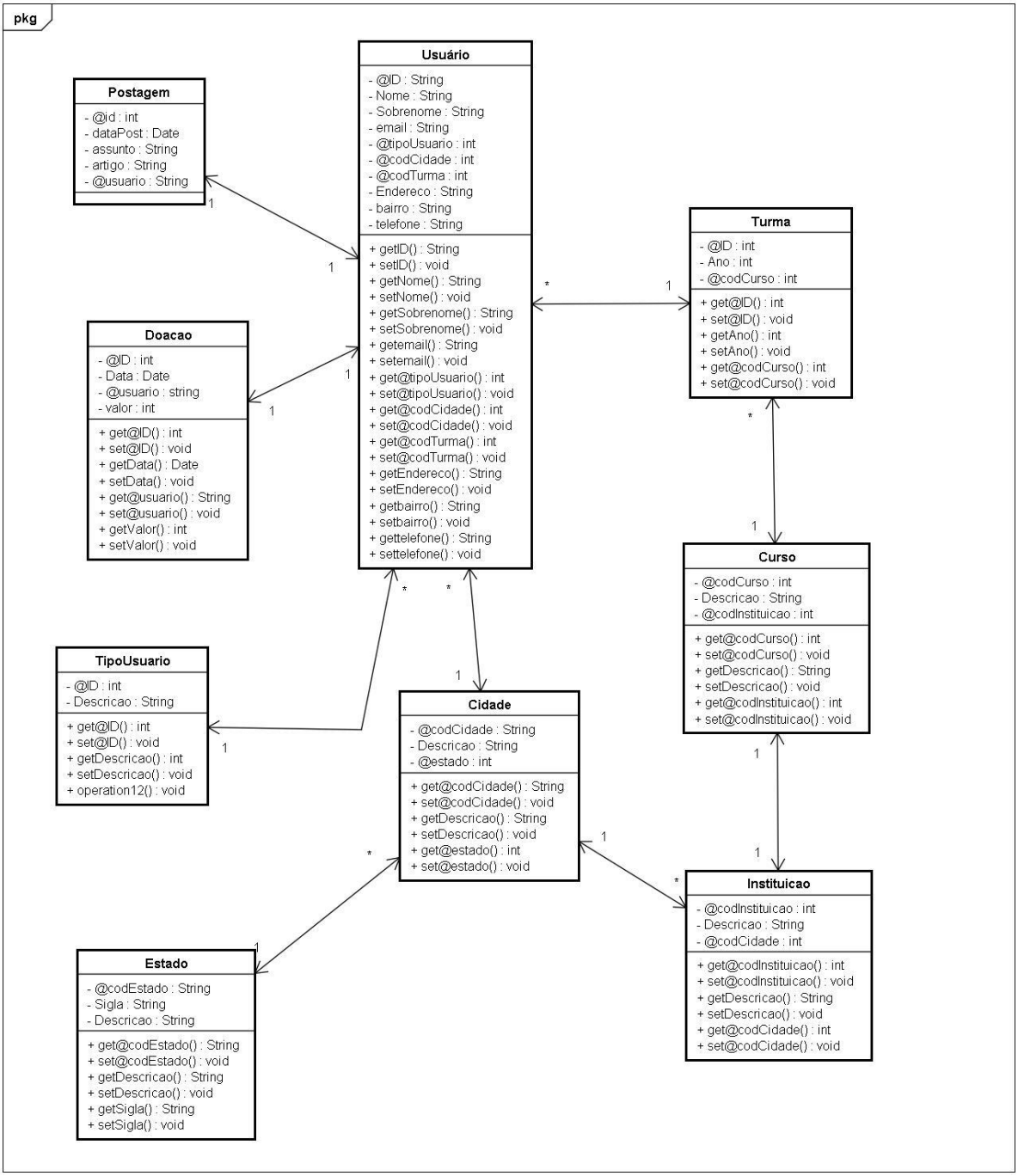

Figura 15 - Diagrama de Classes

3.6 MODELAGEM DE ENTIDADE E RELACIONAMENTO

Abaixo Figura 16 ilustrando a modelagem de entidade e relacionamento:

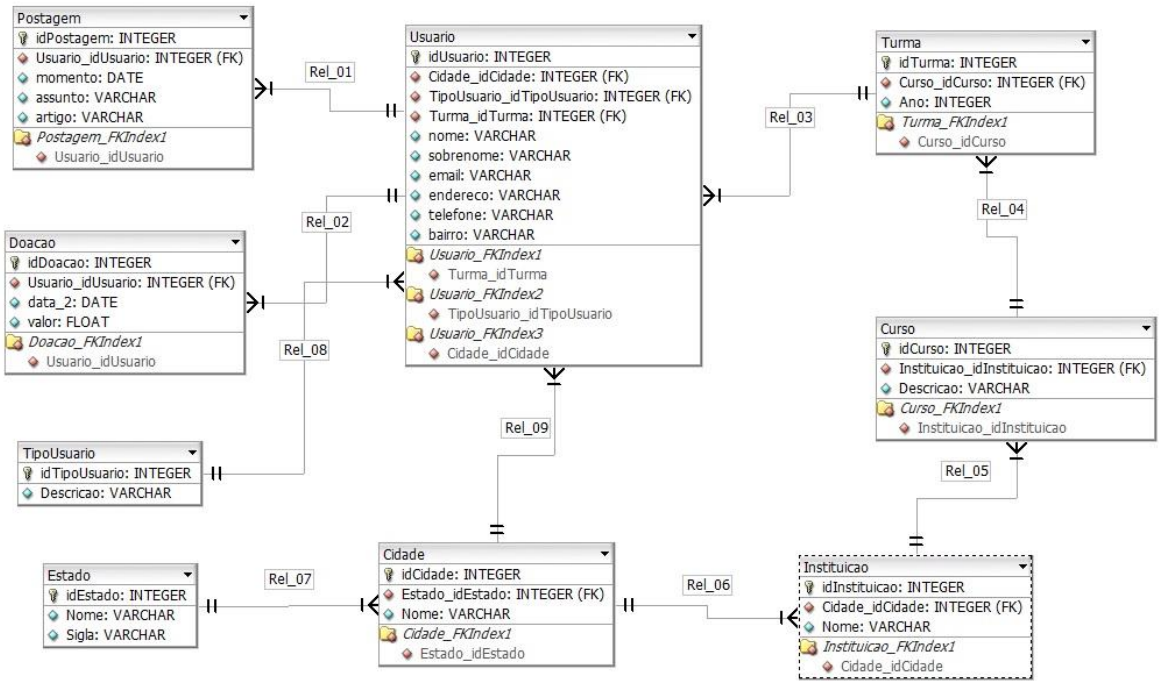

Figura 16 - Modelagem Entidade e Relacionamento

3.7. WORK BREAKDOWN STRUCTURE

Abaixo Figura 17 ilustrando a *Work Breakdown Structure*:

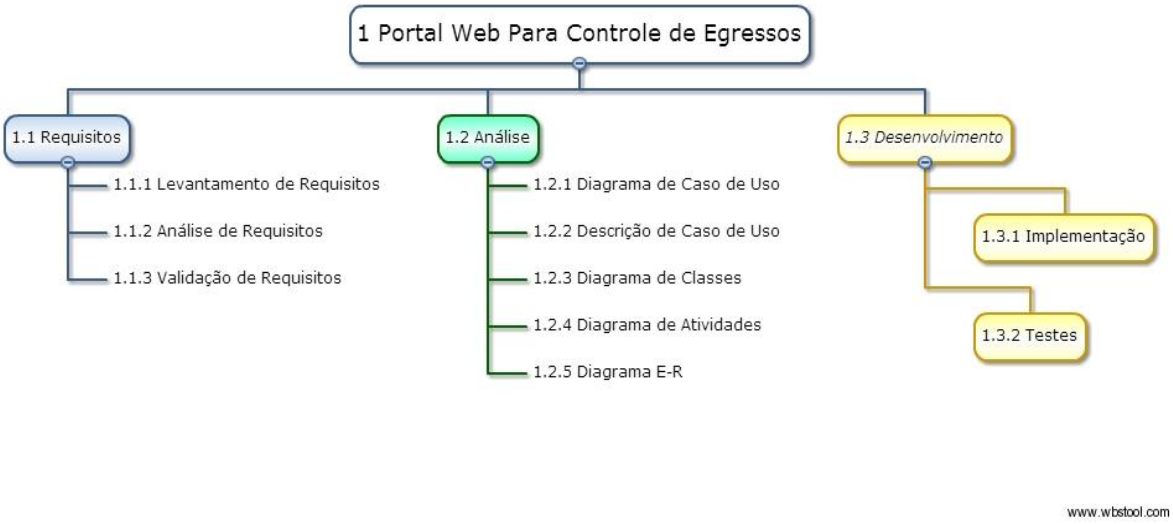

Figura 17 - Work Breakdown Structure

3.8. SEQUENCIAMENTO DE ATIVIDADES

Abaixo ilustração do sequenciamento de atividades:

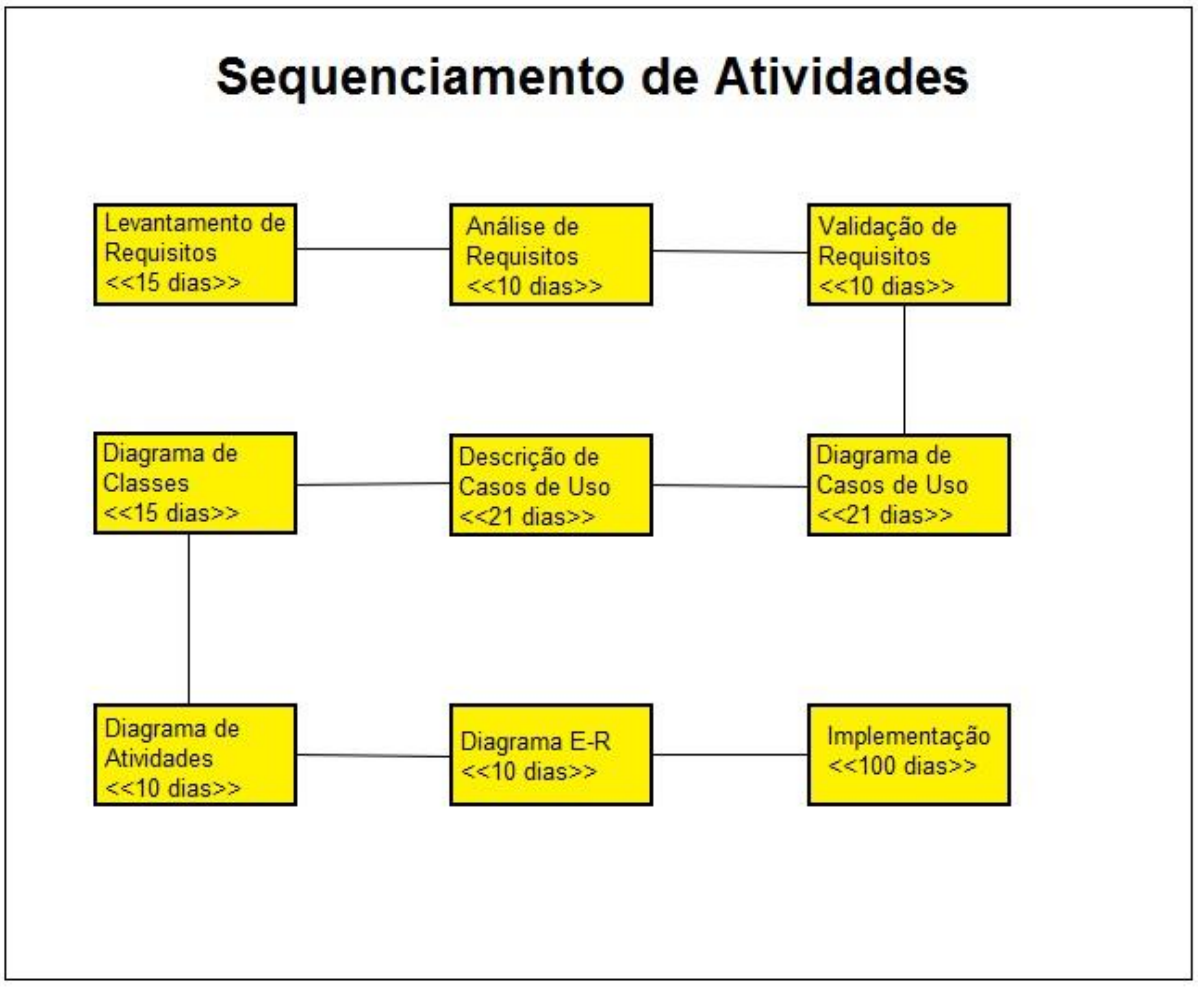

Figura 18 - Sequenciamento de Atividades

3.9 ORÇAMENTO

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

- 01 Analista – Programador
- 01 Impressora Multifuncional;
- 01 Notebook;
- Microsoft Visual Studio 2013;

Analista – Programador			
Analista - Programador	**Quantidade de Horas**	**Valor Hora**	**Total**
Murillo Gonçalves Zampieri	200	R$ 40,00	R$ 8.000,00

Equipamentos			
Equipamento	**Quantidade de Horas**	**Depreciação Diária**	**Total**
Notebook	R$ 2.000,00	R$ 4,17	R$ 104,17
Impressora	R$ 500,00	R$ 1,04	R$ 26,04
Microsoft Visual Studio 2012	R$ 2.391,00	R$ 4,98	R$ 124,53

Total	**R$ 8.254,74**

Tabela 8 Orçamento

4 IMPLEMENTAÇÃO DO SISTEMA

Para a implementação do portal, foi utilizado o ambiente Visual Studio, com a linguagem de programação C#, no qual foi usado o padrão de projetos MVC, com o objetivo de diminuir custos com manutenção e melhor organizar o projeto. A Figura 19 ilustra o projeto projetoTccMvc.

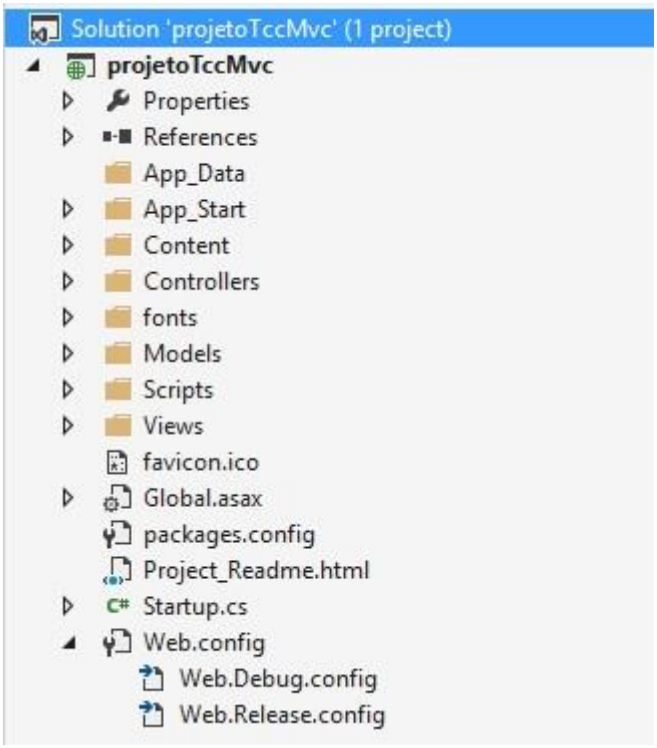

Figura 19 - Projeto ProjetoTccMvc

O projetoTccMvc, é responsável por todas as regras de negócio e acesso a dados, utilizando a linguagem de programação C#.

4.1 ORGANIZAÇÃO DO PROJETO

Para uma melhor organização do projeto, utilizou o padrão de projetos MVC, o projetoTccMvc foi organizado em três camadas: *controllers, models, views*, conforme Figura 20 :

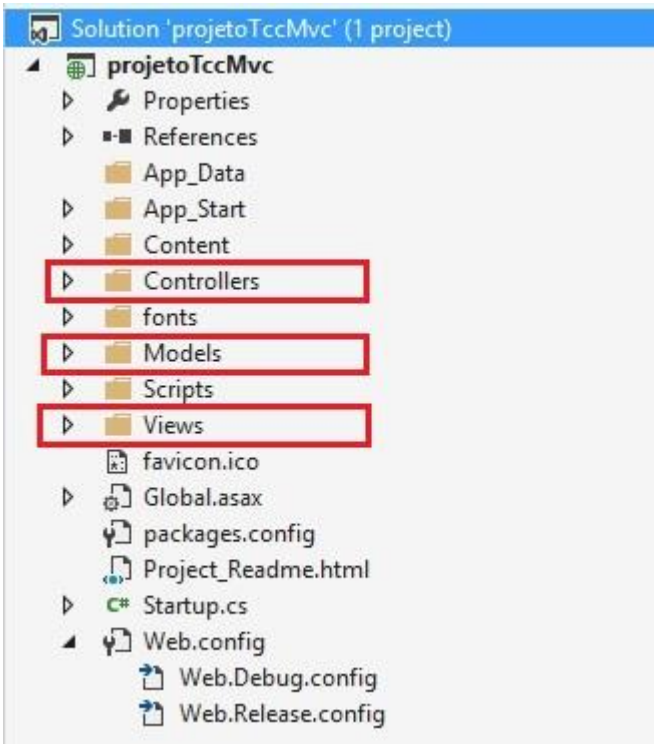

Figura 20 - Camadas ProjetoTccMVC

Abaixo são representadas as camadas que foram destacadas na Figura 20 e suas responsabilidades:

Models: Camada onde se localiza as classes de modelo, além das definições das características e o comportamento das propriedades. O trecho de Código 1 ilustra o mapeamento da classe Instituição.

É possível notar que para a classe instituição foram definidos quatro atributos, sendo eles instituicaoID, instituicaoNome e cidade, responsáveis por representar a *Primary Key*, nome da instituição e a cidade respectivamente. Acima da definição de cada atributo, podem-se visualizar as *annotation,* responsáveis pelas características de cada atributo, elas são apresentadas entre um par de colchetes, por exemplo a *annotation [Key]* está definindo que o atributo instituicaoID, é a *Primary Key*, a *annotation* [DatabaseGenerated(DatabaseGeneratedOption.Identity)], defini que o atributo instituicaoID será preenchido automaticamente em sequencia.

Código 1 - Código C# da Camada Model – Instituições.

```
public class Instituicao
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Display(Name = "Código da Instituição")]
        public int instituicaoID { get; set; }

        [Display(Name = "Nome da Instituição")]
        [StringLength(50)]
        [Required(ErrorMessage="Campo obrigatório")]
        public string instituicaoNome { get; set; }


        [Required(ErrorMessage = "Campo obrigatório")]
        [Display(Name = "Cidade")]
        public int cidadeID { get; set; }
        [ForeignKey("cidadeID")]
        public Cidade cidade { get; set; }

    }
```

Controllers: Camada onde se localiza todas as regras de negócio, nela também está às classes cuja instância é um objeto responsável por acessar os dados. O Código 2 ilustram uma parte do *controller* da classe Instituicao, neste trecho do código nota-se que é criado uma ActionResult, nomeada de *Index*. Na *action* criada, tem duas linhas de código, onde a primeira linha é a responsável por buscar o nome da cidade, que está na tabela Cidades. Já a segunda linha lista todos os dados da tabela Instituicoes, retornando para a *View*, desta *action.*

Código 2 - Código C# da Camada Controller – Instituições.

```
public ActionResult Index()
        {
            var instituicoes = db.Instituicoes.Include(i => i.cidade);
            return View(instituicoes.ToList());

        }
```

No segundo trecho do código *controller* Instituicao, é definido a *action Details*, onde é passado um parâmetro. No Código 3 é feito uma busca na tabela Instituicoes, comparando o parâmetro passado com o id das informações armazenas na tabela, quando encontrado é exibido os dados da instituição buscada, quanto não é dado uma mensagem de item não encontrado.

Código 3 - Código C# da Camada Controller – Instituições.

```
public ActionResult Details(int? id)
      {
         if (id == null)
         {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
         }
         Instituicao instituicao = db.Instituicoes.Find(id);
         if (instituicao == null)
         {
            return HttpNotFound();
         }
         return View(instituicao);

      }
```

No trecho do *controller* mostrado abaixo, se refere a *action Create*, a primeira *action* não tem parâmetro, ela é responsável por mostrar a lista de cidades da *view,* já a segunda *action* é feita a validação dos dados preenchidos na *view*, e salvando se tudo estiver correto, caso contrário é mostrado na *view* o que não estiver correto, para que seja corrigido.

Código 4 - Código C# da Camada Controller – Instituições.

```
public ActionResult Create()
      {
         ViewBag.cidadeID = new SelectList(db.Cidades, "cidadeID", "nomeCidade");
         return View();
      }

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include="instituicaoID,instituicaoNome,cidadeID")] Instituicao
instituicao)
      {
         if (ModelState.IsValid)
         {
            db.Instituicoes.Add(instituicao);
            db.SaveChanges();
            return RedirectToAction("Index");
         }

ViewBag.cidadeID = new SelectList(db.Cidades, "cidadeID", "nomeCidade", instituicao.cidadeID);
         return View(instituicao);
      }
```

No quarto trecho são duas *action Edit*, sendo o primeiro responsável por buscar os dados na tabela em comparação com o parâmetro passado, caso o usuário decida

fazer alguma alteração nos dados exibidos, então é utilizado a segunda *action,* onde é feita a alteração e salvo no banco a alteração. Abaixo Código 5:

Código 5 - Código C# da Camada Controller – Instituições.

```
public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Instituicao instituicao = db.Instituicoes.Find(id);
            if (instituicao == null)
            {
                return HttpNotFound();
            }
            ViewBag.cidadeID = new SelectList(db.Cidades, "cidadeID", "nomeCidade",
instituicao.cidadeID);
            return View(instituicao);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include="instituicaoID,instituicaoNome,cidadeID")] Instituicao
instituicao)
        {
            if (ModelState.IsValid)
            {
                db.Entry(instituicao).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.cidadeID = new SelectList(db.Cidades, "cidadeID", "nomeCidade",
instituicao.cidadeID);
            return View(instituicao);

        }
```

O próximo *controller action Delete,* tem a finalidade de buscar o dado selecionado pelo usuário para exclusão, após encontrado é acionado a *action* DeleteConfirm onde é salvo a remoção, após a confirmação do usuário de que realmente deseja excluir os dados.

Código 6 - Código C# da Camada Controller – Instituições.

```
public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Instituicao instituicao = db.Instituicoes.Find(id);
            if (instituicao == null)
            {
                return HttpNotFound();
            }
            return View(instituicao);
        }
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Instituicao instituicao = db.Instituicoes.Find(id);
            db.Instituicoes.Remove(instituicao);
            db.SaveChanges();
            return RedirectToAction("Index");

        }
```

View: Camada onde se localiza as telas, onde o usuário irá informar e receber informações.

As *view* são tem o nome da *action* na qual se referem e ficam em uma pasta com o nome do *controller,* por exemplo: Vimos acima a *action Index* pertencente ao *controller* Instituicao, então na camada *View* terá uma pasta nomeada de Instituicao e dentro deste diretório um arquivo nomeado de *Index*, desta forma mantendo o projeto muito mais fácil de fazer manutenção. Abaixo veremos um trecho na *View Index* do *controller* Instituicao, onde é possível ver na primeira linha do código que é definido que serão exibidas as informações referentes ao *model* Institucao e abaixo desta linha também nota-se que é definido o *layout* que será utilizado, no caso Adm.cshtml no diretório *Shared*, na camada *View*.

Código 7 - Código C# da Camada View – Instituições.

```
@model IEnumerable<projetoTccMvc.Modelsid.Instituicao>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/Adm.cshtml";

}
```

O Código 8, ilustra a parte do código que se refere ao que será exibido na tela, neste trecho é possível notar que é criado uma tabela e com duas colunas, nome da cidade e nome da instituição, o que aparecerá na tela será o nome definido na *annotation,* que pode ser vista no Código 1acima.

Código 8 – Código C# da Camada View – Instituições

```
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.cidade.nomeCidade)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.instituicaoNome)
        </th>
        <th></th>
    </tr>
```

Após ter sido criada a tabela, é preciso preenchê-la, para isso foi utilizado Código 9, onde ele percorrerá todos os dados referente a classe Instituicao e finalizadas a tabela.

Código 9 – Código C# da Camada View – Instituições

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.cidade.nomeCidade)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.instituicaoNome)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.instituicaoID }) |
            @Html.ActionLink("Details", "Details", new { id=item.instituicaoID }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.instituicaoID })
        </td>
    </tr>

}
</table>
```

4.2. INTERFACE DO PORTAL

Para começar a utilizar o Portal Web para Controle de Egressos é necessário fazer o cadastro, onde o usuário estará preenchendo as informações solicitadas, como está ilustrado na Figura 21 abaixo:

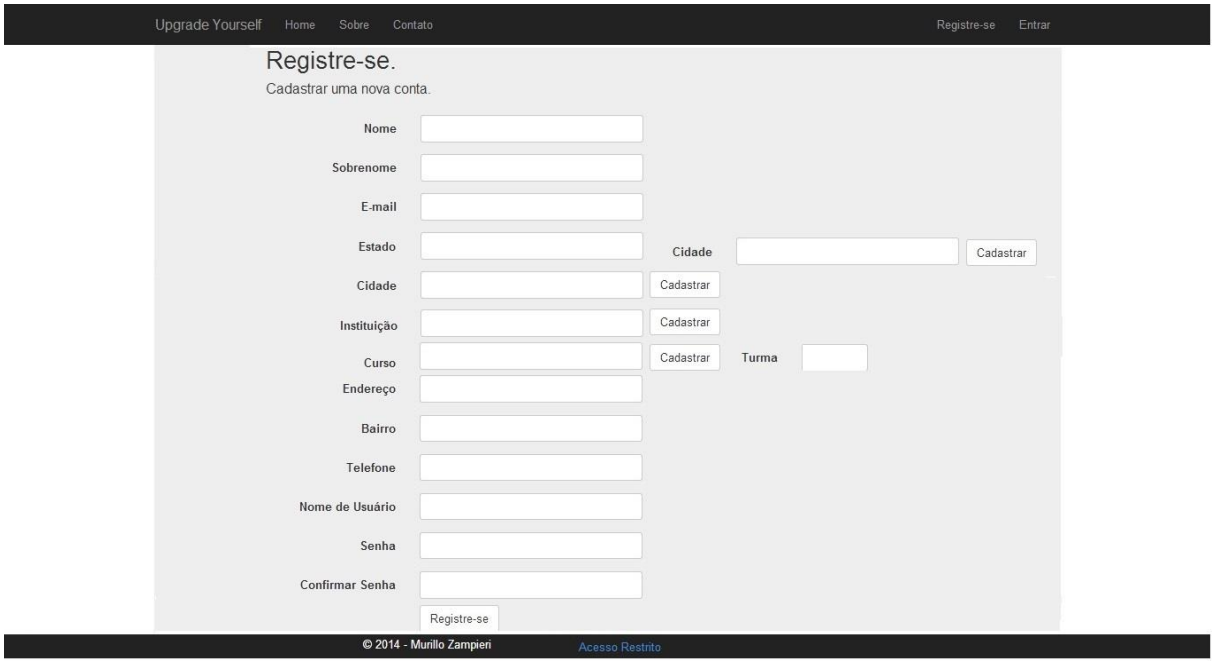

Figura 21 - Tela de Cadastro de Usuário

O Portal Web para Controle de Egressos, tem duas interfaces, uma para os usuários e outra para os administradores, diferenciando o menu de navegação lateral. Abaixo a Figura 22 Menu do Usuário:

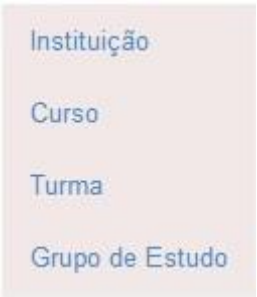

Figura 22 - Menu do Usuário

Diferente do usuário o Administrador terá acesso a todas as classes, podendo ele criar um novo dado, alterar, visualizar e até excluir, abaixo a Figura 23 Menu do Administrador, onde ele escolhe qual a classe deseja trabalhar:

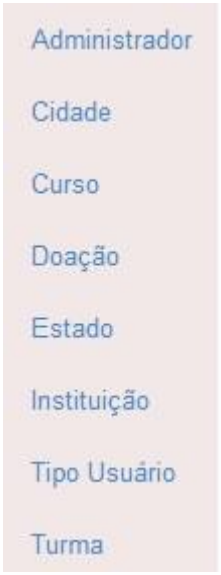

Figura 23 - Menu do Administrador

A principal função do administrador é manter o portal confiável, com dados verdadeiros e informações corretas.

Já o usuário, tem como objetivo maior a troca de informações, onde eles incluem novos artigos e tem acesso aos artigos incluídos pela sua rede de relacionamento.

Quando o usuário selecionar o item Instituição do menu lateral, irá ser feito um filtro de todos os artigos, onde é exibida na tela apenas os artigos postados pelos usuários que pertencem a mesma instituição.

Seguindo o mesmo raciocínio, ao selecionar o item Curso, são exibidos apenas os artigos postados pelos usuários pertencentes ao mesmo curso, e desta mesma forma funciona o item Turma, onde são exibidos apenas os artigos enviados pelos usuários que pertençam à mesma instituição, curso e também o ano de conclusão.

Os Administradores ainda podem consultar as doações feitas pelos usuários, podendo emitir relatórios do período escolhido por ele.

5 CONCLUSÃO E TRABALHOS FUTUROS

A necessidade da constante atualização e busca de informações, tem feito com que as pessoas recorram à internet para encontrar o que precisa. Dessa forma fazendo com que a área de tecnologia da informação tenha um crescimento altíssimo.

O portal web para controle de egressos é desenvolvido para auxiliar no enriquecimento intelectual, tanto para o meio acadêmico quanto para o profissional. Tudo com o auxilio de professores e profissionais da área, que estarão disponibilizando artigos, materiais e as novidades da área de interesse do aluno.

Os diagramas UML, permitem uma visão detalha do funcionamento do sistema, dessa forma, pode-se ter um melhor entendimento das funcionalidades do portal.

Com o uso do conceito do padrão de projetos MVC, foi possível uma melhor organização do projeto, separando a camada das regras de negócio da camada de visualização, desta forma facilitando a manutenção e evolução do sistema.

Durante o desenvolvimento do portal, houve dificuldades, especificamente com a utilização do MVC 5, porém essas dúvidas foram sanadas com o curso de C# (C Sharp) – ASP.NET MVC Avançado[2].

Para o desenvolvimento de trabalhos futuros, pretende-se aplicar uma área para empresas que buscam contratar, onde os usuários estarão disponibilizando seu curriculum e interesses.

[2] http://www.treinaweb.com.br/curso/csharp-avancado-asp-net-mvc

REFERÊNCIAS BIBLIOGRÁFICAS

REFERÊNCIAS DE LIVROS

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.

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

DA SILVA, Alberto Manuel Rodrigues; VIDEIRA, Carlos Alberto Escaleira; **UML, Metodologias e Ferramentas CASE**. Lisboa, Centro Atlântico, 2001.

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

GOES, Wilson Moraes. **Aprenda UML por Meio de Estudos de Caso**. São Paulo, Novatec, 2014.

GREENE, Jennifer; STELLMAN, Andrew. **Use a cabeça C#.** Rio de Janeiro, Altabooks, 2008.

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

SANCHES, Matheus Faria; **Sistema Para Gestão de Obras Civis**. Trabalho de Conclusão de Curso. Assis: 2012.

REFERÊNCIAS DE INTERNET

ALEXANDRE, Claudio; Introduction to the C# Language and the .Net Framework. Disponível em <http://msdn.microsoft.com/library/z1zx9t92> Acesso em 12 de jun 2014.

ARAÚJO, Everton Coimbra de; JAVA e C#.NET – Um breve e introdutório estudo compatativo de duas sintaxes e convenções. Disponível em: <http://www.linhadecodigo.com.br/artigo/1620/java-e-cshartpnetum-breve-.aspx> Acesso em 12 de jun 2014.

AURELIO, Dicionário; **Controle**. Disponível em < http://www.dicionariodoaurelio.com/Controle.html> Acesso em 22 fev 2014.

AURELIO, Dicionário; **Egresso**. Disponível em < http://www.dicionariodoaurelio.com/Egresso.html> Acesso em 22 fev 2014.

BASTOS, Daniel Flores; **O Que é Model-View-Controller (MVC)?**. Disponível em <http://www.oficinadanet.com.br/artigo/desenvolvimento/o_que_e_model-view-controller_mvc>. Acesso em 16 jul 2014.

BUZAN, Tony; **Mapas Mentais**. Sextante. Disponível em <http://www.esextante.com.br/sextante/livros/MapasMentais%20Cap1.pdf> 21 fev 2014.

COLLETTO, Armando Dall; **A Importância do Aperfeiçoamento Profissional**. Disponível em <http://www1.folha.uol.com.br/folha/educacao/ult305u17270.shtml>, 2005.

DE ARAÚJO, Everton Coimbra; **Introdução à Linguagem C#.** Disponível em <http://www.devmedia.com.br/introducao-a-linguagem-c/27711> Acesso em 23 fev 2014.

GODOY, Solange Cervinho Bicalho; GUIMARÃES, Eliane Marina Palhares; Educação Permanente: Uso das Tecnologias de Informação e Comunicação como Ferramenta Para Capacitação Profissional. Disponível em <http://reme.org.br/artigo/detalhes/636>, Acesso em 30 jun 2014.

GUEDES, Gilleanes Thorwald Araújo; **UML 2 Guia Prático**. Novatec. Disponível em <http://www.novateceditora.com.br/livros/uml2/capitulo9788575221457.pdf>. Acesso em 06 mar 2014.

LIMA, Edwin; REIS, Eugênio; **C# E .NET – Guia do Desenvolvedor**. Disponível em < http://www.etelg.com.br/paginaete/downloads/informatica/apostila2.pdf> 5 mar 2014.

LOTAR, Alfredo; **Como Programar com ASP.NET e C#. Novatec**. Disponível em < http://www.novateceditora.com.br/livros/aspnet2/capitulo9788575221211.pdf> Acesso em 5 mar 2014.

LOUREIRO, Henrique; **C# 5.0 Com Visual Studio 2012 – Curso Completo**, Lidel. Disponível em <https://www.fca.pt/cgi-bin/fca_main.cgi/?op=2&isbn=978-072-722-752-5> Acesso em 1 mar 2014

LUCCI, Elian Alabi; **A Era Pós-Industrial, a Sociedade do Conhecimento e a Educação Para o Pensar**. Disponível em <http://www.hottopos.com/vidlib7/e2.htm> Acesso em 23 out 2013.

MACORATTI, José Carlos; **UML – Diagrama de Classes e objetos**. Disponível em <http://www.macoratti.net/net_uml1.htm> Acesso em 14 mar 2014.

MEDEIROS, Higor; **Introdução ao Padrão MVC**. Disponível em <http://www.devmedia.com.br/introducao-ao-padrao-mvc/29308> acesso em 16 julho 2014.

MICROSOFT; **Introdução à Plataforma .NET da Microsoft**. Disponível em <http://msdn.microsoft.com/pt-br/aa702903.aspx> Acesso em 19 fev 2014.

MICROSOFT; **Introdução à Linguagem C# e ao .NET Framework**. Disponível em <http://msdn.microsoft.com/library/z1zx9t92> Acesso em 23 fev 2014.

MICROSOFT; **Visão Geral do SQL Server**. Disponível em < https://www.microsoft.com/sqlserver/pt/br/product-info/overview-capabilities.aspx> Acesso em 6 mar 2014.

MOURA, José; **Entendendo o Padrão de Projeto MVC**. Disponível em <http://www.blogomoura.com/2011/07/entendendo-o-padrao-de-projeto-mvc/> Acesso em 11 Ago 2014.

MUNIZ, Marcos de Paula; **Desenvolvimento WEB – Uma área que não para de crescer**. Disponível em <http://www.ituiutaba.uemg.br/sistemas/index.php?option=com_content&view=article&id=35:desenvolvimento-web-uma-area-que-nao-para-de-crescer> 22 fev 2014.

PACIEVITCH, Yuri; **SQL Server**. Disponível em <http://www.infoescola.com/informatica/sql-server/> 24 fev 2014.

RIBEIRO, Leandro; **O que é UML e Diagramas de Caso de Uso: Introdução Prática à UML**. Disponível em < http://www.devmedia.com.br/o-que-e-uml-e-

diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408> Acesso em 15 mar 2014.

SANT'ANNA, Mauro; **C#: A Nova linguagem de arquitetura .NET**. Disponível em <http://www.linhadecodigo.com.br/artigo/15/csharp-a-nova-linguagem-da-arquitetura-net.aspx> Acesso em 28 fev.2014.

VAMBERTO, Carlos; **Curso de C# Introdução ao .NET com C#.** Disponível em <http://www.etelg.com.br/paginaete/downloads/informatica/apostila.pdf> 23 fev 2014.

ANEXO I

CRONOGRAMA DE DESENVOLVIMENTO DO TRABALHO

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

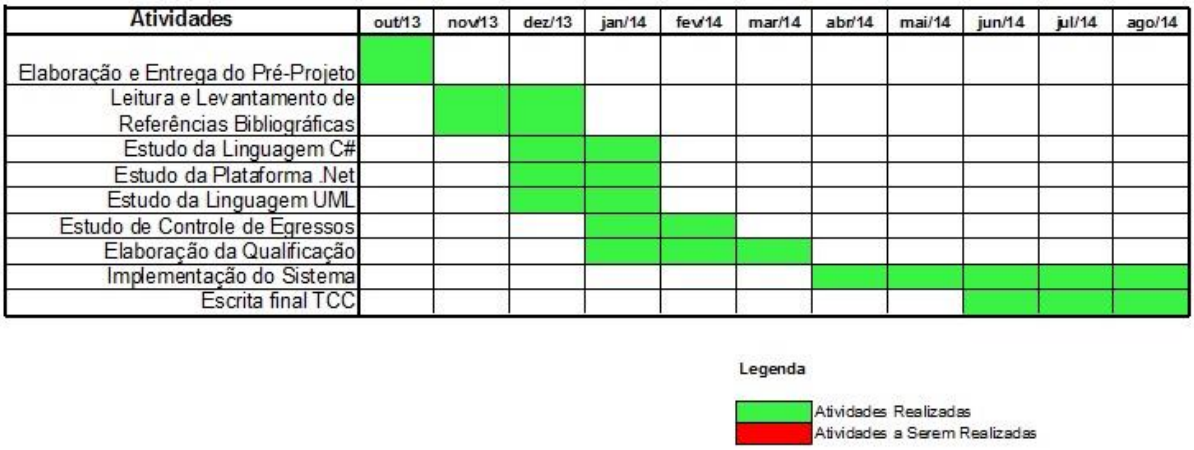

Figura 24 - Cronograma