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

DANIEL DE SOUZA CAMPOS

SISTEMA DE INFORMATIZAÇÃO PARA REDE PRIVADA DE SAÚDE

Assis

2013

DANIEL DE SOUZA CAMPOS

SISTEMA DE INFORMATIZAÇÃO PARA REDE PRIVADA DE SAÚDE

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: Prof. Dr. Alex Sandro Romeo de Souza Poletto

Área de Concentração: Informática

Assis

2013

FICHA CATALOGRÁFICA

CAMPOS, Daniel de Souza.

Sistema de Informatização para Rede Privada de Saúde / Daniel de Souza Campos. Fundação Educacional do Município de Assis – FEMA –Assis, 2013.

104 p.

Orientador: Alex Sandro Romeo de Souza Poletto.

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

1.Saúde. 2.Consultas. 3.Histórico. 4.CID. 5.Rede Privada de Saúde.

CDD: 001.61

Biblioteca da FEMA

SISTEMA DE INFORMATIZAÇÃO PARA REDE PRIVADA DE SAÚDE

DANIEL DE SOUZA CAMPOS

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: Alex Sandro Romeo de Souza Poletto

Analisador: Guilherme de Cleva Farto

Assis

2013

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus por me dar a sabedoria, a paciência e a força de vontade para concluí-lo da melhor maneira possível.

AGRADECIMENTOS

Primeiramente a Deus, por todas as oportunidades de aprendizado que tem me proporcionado.

Ao meu Orientador Prof. Dr. Alex Sandro Romeu de Souza Poletto, pela dedicação, paciência e orientação para a realização deste projeto.

Aos meus colegas de curso e de trabalho que também contribuíram para a realização do mesmo.

A todos os professores, que não medem esforços para ajudar e nos fazer crescer como profissionais e como seres humanos, em especial ao Prof. Osmar, motivando-me a não desistir do projeto e por todos seus conselhos.

E, obviamente, a toda minha família, que me apoiam a cada dia em minhas decisões.

“A essência do conhecimento consiste em aplicá-lo, uma vez possuído.”

Confúcio
(551 a.C. - 479 a.C.)

RESUMO

Este trabalho descreve toda análise, documentação, criação e implementação de um *software* para informatização do sistema privado de saúde, idealizado para a auxiliar a tomada de decisão de médicos em clínicas particulares, geralmente de médio a pequeno porte, de maneira segura, rápida e eficaz. Este *software* contará com um histórico, que reúne informações precisas, para que as mesmas sejam consultadas e atualizadas pelo usuário, visto que, estas informações, são essenciais para diagnósticos corretos, bem como o ideal tratamento dos mesmos. A ideia de criação do *software* surgiu após experiências próprias em consultórios médicos, onde o médico, em suas consultas, se via obrigado a perguntar ao paciente se este possuía algum tipo de restrição medicamentosa ou quais foram seus tratamentos anteriores. Tratando-se de ferramentas para a criação, o *software* foi construído com o *Visual Studio 2012*, que oferece uma ampla gama de opções de criação, utilizando-se também, do banco de dados da própria ferramenta, o *Microsoft SQL Server 2012*. Foi utilizada a plataforma *Web ASP.Net* juntamente com a linguagem de programação C# para a criação das páginas e formulários, não exigindo altos custos de implantação e de manutenção, tanto para o usuário quanto para o programador.

Palavras-chave: Informatização; Histórico; Consulta; Médico.

ABSTRACT

This work describes all the analysis, documentation, development and implementation of a software to the computerization of the private health care system, designed to assist doctors in their work decisions into private clinics, usually at small or medium size clinics, in a safe, quick and effective way. This software will have a history which gathers accurate information that will be consulted and updated by the user, since it is essential to provide right diagnosis, as well as their ideal treatment. The idea of creating the software came after my own experiences in doctors' offices, where the doctor, in his consultations, was obliged to ask to a patient if he or she had some sort of pharmacological restriction or what treatments he or she had undergone previously.

In relation to the development tools, the software was built with Visual Studio 2012, offering a wide range of creation options, using the database from the own tool, Microsoft SQL Server 2012. It was used the ASP.Net Web platform along with the C # programming language in order to create pages and forms, not requiring high costs of deployment and maintenance, neither for the user nor for the programmer.

Keywords: Computerization; History; Consult; Doctor.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

WBS	*Work Breakdown Structure*
UML	*Unified Modeling Language*
SQL	*Structure Query Language*
C#	*C Sharp*
ASP	*Active Server Pages*
HTML	*Hypertext Markup Language*
CID	Classificação Internacional de Doenças
SGDB	Sistema de Gerenciamento de Banco de Dados
CRUD	*Create, Read, Update, Delete*
DAL	*Data Access Layer*
BLL	*Business Logic Layer*
MVC	*Model-View-Controller*
RAM	*Random Access Memory*
IDE	*Integrated Development Environment*

SUMÁRIO

1. INTRODUÇÃO

O Sistema de Informatização para Rede Privada de Saúde tem como principal objetivo interligar consultórios médicos particulares, que fazem parte de algum tipo de cooperativa médica ou convênio, para que seus pacientes não tenham a preocupação de ter que lembrar de medicamentos ou tratamentos que foram ou estão sendo ministrados no ato da atual consulta.

Com a implantação desse sistema, o médico que atender seu paciente saberá informações das quais são relevantes para seu diagnóstico, como por exemplo, quais remédios o paciente está tomando, a posologia de seu tratamento, o CID (Classificação Internacional de Doenças) dos tratamentos anteriores, bem como a correta medicação sobre a queixa atual do paciente.

É comum o paciente não lembrar dos medicamentos que está fazendo uso quando perguntado, além de muitos medicamentos interagirem entre si, como por exemplo, um medicamento comumente indicado para tratamento de males do estômago, a Cimetidina. Essas interações podem afetar drasticamente a solução de sua atual queixa, bem como trazer riscos à saúde do mesmo.

Vale ressaltar que o projeto teve sua estrutura adaptada para estar de acordo com a Lei nº 10.241 de 17 de março de 1999, Artigo 2º, §IV onde se lê: *“ter resguardado o direito sobre seus dados pessoais, através da manutenção do sigilo profissional, desde que não acarrete riscos a terceiros ou a saúde pública.”*

1.1. OBJETIVOS

O sistema servirá de auxiliador na tomada de decisões de possíveis tratamentos, bem como restrições medicamentosas e informações

importantes para as clínicas e seus médicos, sendo estes, os usuários mais visados neste projeto.

Os dados pessoais do paciente, bem como os tratamentos anteriores serão guardados em um banco de dados, e poderão ser cadastrados pelo próprio usuário (médico ou outra pessoa de escolha do médico, como por exemplo, secretárias e recepcionistas), agilizando assim todo o processo desde a consulta até o diagnóstico final.

A implantação deste módulo viabilizará mais controle, segurança e eficiência para aplicação de tratamentos e maior agilidade na tomada de decisão por parte dos usuários, sendo o mesmo otimizado para clínicas médicas particulares.

1.2. JUSTIFICATIVAS

A necessidade de desenvolvimento e implantação deste *software* se dá pela falta de controle medicamentoso, bem como de diagnósticos, por parte de sistemas de gestão médica de clínicas, que em sua maioria, registram apenas a movimentação do paciente, como data e horário de consultas e exames.

1.3. PÚBLICO ALVO

O *software* desenvolvido atingirá diretamente os pacientes da rede privada de saúde, dando-lhes segurança em relação a possíveis diagnósticos e tratamentos, uma vez que estas informações serão armazenadas de forma sigilosa e será somente analisada por médico competente.

Médicos também serão beneficiados, dando-lhes auxílio para um correto diagnóstico e prescrição medicamentosa em seus pacientes, diminuindo

riscos de problemas que possam surgir, além de um diagnóstico preciso e garantia de bom atendimento e saúde para seus pacientes.

1.4. LEVANTAMENTO DOS REQUISITOS

Foram levantados os seguintes requisitos:

- Cadastro
  - Usuário/Médico/Login
  - Paciente
  - Clínica
  - CID
  - Histórico
- Atualização
  - Usuário/Médico
  - Paciente
  - Clínica
  - CID
- Consulta
  - Usuário/Médico
  - Paciente
  - Clínica
  - CID
  - Histórico

1.4.1. DESENVOLVIMENTO

Para o desenvolvimento do sistema, foram levados em conta experiências pessoais de pacientes, bem como conversas informais com médicos no decorrer de consultas. Todas as informações foram colhidas com o passar do tempo, sendo a ideia principal do Sistema, a de mostrar um histórico médico de cada paciente, extraída de uma outra ideia, de maior

amplitude, que por consequência de leis que garantem a privacidade das informações entre Paciente-Médico, não pode ser trabalhada. Assim, para chegar ao resultado final, foi consultado uma pessoa com conhecimentos Jurídicos, para esclarecer o que poderia e o que não poderia estar presente no sistema.

2. ANÁLISE DE REQUISITOS

Neste capítulo, será levantado os tipos de requisitos técnicos que se fazem necessários para a correta utilização e construção do sistema, bem como os conflitos encontrados em relação a estes requisitos e suas possíveis soluções.

2.1. CLASSIFICAÇÃO DOS REQUISITOS

Os requisitos para a implantação do sistema, em suma, não exigem altos investimentos em *hardware* e *software*. Por ser um sistema *web*, um dos requisitos mínimos é uma boa conexão com a *internet*, e a utilização de um navegador de *internet*, disponível gratuitamente na *web*.
Para facilitar a análise, os requisitos foram classificados em Requisitos Computacionais e Requisitos Humanos.

2.1.1. Requisitos Computacionais

Um computador básico, com processador *dual core*, dos fabricantes AMD ou Intel, 1Gb de memória RAM (*Radom Access Memory*) no mínimo, placa de rede e conexão com a *internet*.
Por ser um sistema para a *web*, não se exige espaço de instalação no computador do cliente, podendo o sistema ser acessado também por dispositivos móveis que tenham acesso à *internet*.
Para hospedagem do sistema, um servidor Intel Xeon mais simples, com 8Gb ou 16Gb de memória RAM, e um HD (*Hard Disk*) de 500Gb para armazenamento de dados. Uma conexão com a *internet* acima de 2mb também seria o ideal, para poder transmitir os dados sem lentidão, pois o processamento das requisições e das páginas ocorrem todas no servidor *web*.

2.1.2. Requisitos Humanos

Um profissional habilitado que entenda o mínimo de informática, e que saiba utilizar as ferramentas de navegação na *internet*.

2.2. CONFLITOS ENCONTRADOS

Possíveis conflitos encontrados são:

- Não disponibilidade de acesso à *internet*, tanto para o usuário, como para o servidor da aplicação;
- Ocorrências que tenham a ver com fatores climáticos, tais como falta de energia, chuva muito forte, que irão de alguma maneira impossibilitar o correto funcionamento do servidor;
- Falta de habilidade do usuário para manipulação do sistema;
- Manutenção mal feita nos recursos computacionais;
- Espaço físico para armazenamento do servidor de aplicação que não atenda requisitos básicos de conservação.

2.3. PROPOSTAS DE SOLUÇÃO

Para solucionar estes possíveis conflitos, deverá ser adotado as seguintes especificações:

- Conexão com a *internet* estável.
- Usar *nobreaks* para evitar possíveis falhas de energia, além de o servidor estar localizado em local apropriado, com ventilação e com pouco poeira.
- Usuário que tenha ao menos noções básicas de informática, sendo que, especificamente para o sistema, será disponibilizado treinamento para o usuário.

- Correta manutenção no servidor quando necessário, geralmente a cada 30 dias.

2.4. LISTA DE EVENTOS

Número	Evento	Caso de uso
01	Usuário faz Login	Login
02	Usuário cadastra Médico	Cadastrar Usuário/Médico
03	Usuário entra na Home	Acessar Homepage
04	Usuário cadastrar Clínica	Cadastrar Clínica
05	Usuário cadastra Paciente	Cadastrar Paciente
06	Usuário cadastra Histórico	Cadastrar Histórico
07	Administrador cadastra CID	Cadastrar CID
08	Usuário consulta Médico	Consultar Médico
09	Usuário consulta Clínica	Consultar Clínica
10	Usuário consulta Paciente	Consultar Paciente
11	Usuário consulta Histórico	Consultar Histórico
12	Usuário consulta CID	Consultar CID
13	Administrador atualiza CID	Atualizar CID
14	Usuário atualiza Paciente	Atualizar Paciente
15	Administrador atualiza Usuário	Atualizar Usuário/Médico
16	Usuário atualiza Clínica	Atualizar Clínica

Tabela 1 – Lista de Eventos

Conforme a lista de eventos descrita acima, bem como as propostas de soluções de possíveis conflitos, mostra-se de suma importância a análise e classificação dos requisitos técnicos, proporcionando ao projeto que se encaminhe da forma mais clara possível para a fase do planejamento. Esta análise visa um entendimento do que será feito no sistema pelos usuários, além de identificar seu passo a passo na manipulação de telas e dados.

3. PLANEJAMENTO DO PROJETO

Neste capítulo será apresentada as ferramentas de análise, modelagem e desenvolvimento do sistema, visando garantir uma abordagem simples, porém completa, de conceitos e descrições das referidas ferramentas, tornando o planejamento o mais ágil e completo possível.

3.1. DESCRIÇÃO DA METODOLOGIA DE ANÁLISE

Para a modelagem do sistema, foi utilizado a metodologia de Análise Orientada a Objeto, constituída pela linguagem de análise UML (*Unify Modeling Language*), sendo ela a mais utilizada, por ser uma linguagem universal, adaptativa e que pode ser usada para dar base a sistemas construídos em qualquer linguagem de programação orientada a objetos disponíveis, tais como o C#, Java, F# entre outras.

3.2. DESCRIÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Após concluída a modelagem, o sistema foi desenvolvido utilizando a ferramenta *Visual Studio Professional 2012*, da empresa *Microsoft*, utilizando a plataforma *ASP.Net*. Para armazenar todos os dados necessários, utilizou-se o banco de dados *Microsoft SQL Server 2012*, o qual está incluído na ferramenta Visual Studio e atende satisfatoriamente os requisitos do sistema. Utilizou-se também o IIS (*Internet Information Service*), *Web Service* próprio da ferramenta *Visual Studio*, evitando configurações mais complexas e possíveis incompatibilidades.

3.3. FERRAMENTAS DE DESENVOLVIMNETO

Para a construção do sistema, foram usadas duas ferramentas indispensáveis, que são de fácil manipulação e não exigem alto

conhecimento específico. Se o projeto exigir algum estudo mais aplicado e específico, as ferramentas de desenvolvimento usadas possuem extensa documentação e vários títulos bibliográficos disponíveis para possíveis consultas. São elas o *Microsoft Visual Studio 2012* e o *Microsoft SQL Server 2012*.

Para a Modelagem e Análise do projeto foram usadas as ferramentas *Astah Professional*, especializada em UML, *DB Designer Fork* para a modelagem do banco de dados e *Gantt Project*, para a gestão do tempo do projeto.

3.3.1. Microsoft Visual Studio 2012 Professional

Segundo Marcos (2012), o *Visual Studio* é uma ferramenta IDE (*Integrated Development Environment*), ou seja, um *software* com editor de texto que possibilita a criação de vários programas, tanto para *desktop* quanto para *web*. O *Visual Studio* possui linguagens próprias de programação: o C++, C# e o *VB.Net*. Possui diversas ferramentas auxiliares para depuração (detecção de erros de sintaxe, de código e de lógica de programação) e é plenamente integrado com a plataforma *.NET*, facilitando sua interação com banco de dados e com os compiladores de suas linguagens.

Um dos componentes mais marcantes do *Visual Studio* é a possibilidade de usar pouca digitação de linha de código na programação, tornando a atividade mais visual. Isso proporciona a facilidade de clicar em um componente na caixa de ferramentas e arrastá-lo para o formulário de programação, além de deixar a conexão com banco de dados mais simples, sem escrever uma linha de código.

3.4. FERRAMENTAS DE ANÁLISE

Para que o desenvolvimento do projeto seja executado da melhor maneira possível, sem grandes ocorrências de atrasos que podem influenciar no

resultado final, bem como de erros, sejam técnicos ou de programação, se faz necessária a análise de toda uma documentação do mesmo. Nesta documentação, os envolvidos no projeto tem uma visão geral, porém detalhada, do que se pretende desenvolver, por meio de diagramas, tabelas, fluxogramas e outros. Para esta análise, usa-se ferramentas que auxiliam na tomada de decisão, na gerência de tempo, na estrutura do que se irá desenvolver, facilitando o desenvolvimento do projeto e tornando-o mais fiel ao objetivo especificado em sua fase inicial.

Para a confecção dessa documentação, foram usadas ferramentas de análise atuais, que suprem a necessidade de entendimento do projeto, por meio de uma extensa gama de diagramas, sendo estas o *Astah Professional* para documentação de processos e atividades, o *DB Designer Fork* para documentação em relação ao banco de dados e o *Gantt Project* para gerenciamento de tempo do projeto.

3.4.1. Astah Professional

O *Astah* é uma ferramenta de modelagem em UML, que permite a construção de vários diagramas essenciais para a análise de *software*. Usada por grande parte dos engenheiros de sistemas, o *Astah* permite, de maneira simples, intuitiva e rápida, a construção de diagramas, tais como o de Caso de Uso, de Atividade, de Objetos, CRUD (*Create, Read, Update, Delete*), Entidade-Relacionamento, Classes e etc. Projetado pela empresa *Change Vision*, utiliza a linguagem de UML mais segura e estável, em sua versão 2.0. Disponível nas versões *Community* e *Professional*, o *Astah* conta ainda com opções como exportar diagramas para imagens, fazer a engenharia reversa de código pronto em Java para os diagramas, e fazer a conversão dos diagramas em linguagem Java.

3.4.2. DB Designer Fork

O *DB Designer Fork* é uma ferramenta para modelagem de dados, trabalhando com modelos lógicos e desenvolvida pela *fabFORCE*. Possui licença livre, sendo multiplataforma e implementado em *Delphi/Kylix*. O *DB Designer* permite a modelagem, criação e manutenção de banco de dados, bem como fazer a engenharia reversa, ou seja, gerando o modelo de dados a partir de um banco existente, sincronizando-os.
Originalmente construída para oferecer suporte ao *MySQL*, também permite sincronização com outros SGBDs (Sistema de Gerenciamento de Banco de Dados), como *Oracle*, *SQL Server*, *SQLite* e outros que permitem acesso via OBDC (*Open Database Connectvity*).

3.4.3. Gantt Project

O *Gantt Project* é uma ferramenta livre, que possibilita a criação de Diagramas de *Gantt* para gerenciamento de projetos.
O Diagrama de *Gantt* é um gráfico usado par ilustrar diferentes etapas de um projeto. Os intervalos de tempo que representam o início e o fim de cada atividades, aparecem como barras coloridas sobre o eixo horizontal do gráfico.
Desenvolvido em 1917 pelo engenheiro Henry Gantt, tem como principal função visualizar as tarefas de cada membro de uma equipe, analisando seu empenho e seu tempo de execução.
O programa também permite a exportação dos gráficos para imagem .JPG, para arquivo da ferramenta *Microsoft Project*, para arquivo .PDF e .HTML.

3.5. BANCO DE DADOS

Para o correto funcionamento do sistema, bem como a elaboração do histórico pelo médico, têm se a necessidade de guardar e tratar dados de todos os atores, ou usuários, sejam eles pacientes, médicos e clínicas onde o sistema estará implantado.

O *Visual Studio* oferece, integrada a sua IDE, um banco de dados seguro, robusto e compatível com a grande maioria de sistemas, sejam eles desktop ou web, tornando a programação do banco de dados do sistema mais fácil e totalmente integrada com outras etapas de programação.

3.5.1. Microsoft SQL Server 2012

Segundo Mistry (et al, 2012), o *SQL Server* é um gerenciador de banco de dados relacional criado pela *Microsoft* para uso em sistemas corporativos ou de outros portes.

É um banco de dados robusto, seguro e de fácil manipulação, tornando as rotinas no banco de dados mais fáceis de serem criadas, atualizadas e gerenciadas, além das mesmas serem compatíveis com as linguagens da plataforma *.NET*.

O SQL Server trabalha somente sobre a plataforma *Windows*, diferente de outros SGBD's, como o *Oracle* e o *MySql*. Atualmente encontra-se na versão 2012, com várias melhorias implementadas ao longo de suas versões.

3.6. LINGUAGENS UTILIZADAS

Para a implementação, foram utilizadas linguagens de programação presentes na construção de sistemas no nosso dia a dia, utilizadas principalmente na construção de sites web.

A linguagem HTML oferece uma mobilidade e adaptabilidade ao sistema, deixando-o leve, fácil de manipular e adaptável as condições de navegação que o usuário do sistema terá no momento de sua utilização.

Utilizou-se também do C#, uma linguagem de programação orientada a objetos, que ficará responsável pelas ações do sistema, ou seja, todos os eventos e processos que ocorrerão ao clicar em algum botão, por exemplo.

3.6.1. HTML (Hyper Text Mark-up Language)

Idealizada em 1980 pelo cientista Tim Berners-Lee, a linguagem HTML tinha como principal objetivo o acesso e a troca de informações entre cientistas de diferentes universidades, tornando-se um sucesso ao longo dos anos, possibilitando as fundações da *internet* como conhecemos hoje em dia.

É uma linguagem de marcação de texto, ou seja, utiliza-se *tags*, "<>", que funcionam como comando ou marcações para a construção de sites e formulários para *web*. Os *browsers*, ou seja, os navegadores, identificam estas estruturas e mostram as páginas como elas realmente são, com elementos textuais, gráficos e outros.

3.6.2. ASP.Net

Segundo LOTAR (2010, p. 115):

> *O ASP.Net é um modelo de desenvolvimento web que inclui os recursos necessários para o desenvolvimento de websites dinâmicos, sendo parte do .Net Framework.*
> *[..]*
> *O ASP.Net utiliza arquivos de configuração para controlar o comportamento da aplicação. Esses arquivos alteram o modo como funciona o servidor web, o websystem ou uma aplicação web, além de utilizarem o formato XML, o que facilita a manutenção da aplicação.*

Também segundo Lotar (2010), *ASP.Net* é sucessora da plataforma *ASP (Active Server Pages)*, sendo uma plataforma da *Microsoft* para desenvolvimento *web*, fazendo parte do ambiente *.Net Framework*. Herdando todas as suas características, possibilita assim, uma programação mais estável, segura e escalonável para ser compatível com

qualquer navegador *web* ou dispositivo. Também possui compatibilidade com várias linguagens de programação, como o C# e o C++. Com o *ASP.Net*, pode se criar formulários *web*, páginas mais complexas, além da plataforma trabalhar integrada ao *Visual Studio*, o que possibilita a programação ser mais visual e amigável.

3.6.3. C#

Lotar (2010), descreve o C#, leia-se *c sharp*, como uma linguagem de programação criada para o desenvolvimento de uma infinidade de aplicações que são executadas sobre a plataforma *.NET Framework*. Simples, poderosa, com tipagem segura e orientada a objetos, o C# permite o desenvolvimento rápido de aplicações, mantendo o estilo das linguagens *C-style*.

Primeiramente desenvolvida com o nome de *Cool*, e renomeada para C# no lançamento da plataforma *.Net*, ajudou muito no seu desenvolvimento, fazendo com que esta plataforma não fosse obrigada a se adaptar a outras linguagens já existentes, criando assim, uma linguagem própria para ela. O C# também possui compatibilidade com várias outras linguagens, tais como o C++, VB.NET e J#.

3.6.4. A Plataforma .Net

Segundo LIMA (2002, p. 3):

> *.NET é a nova plataforma de desenvolvimento da Microsoft que tem como foco principal o desenvolvimento de Serviços WEB XML. Um serviço Web XML, ou simplesmente Web Service como o chamaremos de aqui em diante por simplicidade e coerência com a linguagem da indústria de software, transcende ao que nós conhecemos como páginas dinâmicas, as quais podem ser acessadas a partir de um browser. A ideia*

> *central de um Web Service consiste em permitir que as aplicações, sejam elas da Web ou Desktop, ou ainda middleware, se comuniquem e troquem dados de forma simples e transparente, independente do sistema operacional ou da linguagem de programação.*

3.6.5. .Net Framework

Segundo LOTAR (2010, p. 27):

> *O .Net Framework é um componente integrado ao Windows que dá suporte à execução e ao desenvolvimento de uma nova geração de aplicações e XML web services.*
> *Segundo a documentação, o .Net Framework foi projetado como os seguintes objetivos:*
>
> - *Oferecer um ambiente consciente de programação orientado a objetos, de modo que o código do objeto seja armazenado e executado localmente, mas com a possibilidade de ser armazenado na internet e executado de forma remota.*
> - *Oferecer um ambiente de execução de código que minimiza o desenvolvimento de software e conflito de versões.*
>
> *[...]*
>
> - *Construir toda a comunicação em padrões reconhecido pela indústria para que o .Net Framework possa se integrar com qualquer tipo de código.*

3.6.6. CSS (*Cascading Style Sheets*)

O CSS, ou uma “folha de estilos”, é composta por camadas e utilizada para definir a aparência em páginas de *internet* que utilizam, em sua construção, linguagens de marcação, como o *HTML*. Sua maior vantagem é separar o estilo de formatação do conteúdo da página, deixando-a mais organizada, sendo esse, o principal motivo de sua criação.

3.6.7. jQuery

Segundo Silva (2008, p. 16), jQuery é uma biblioteca disponibilizada como software livre e aberto, criada por John Resig, utilizando como base o *JavaScript*. Sua principal função é tornar a programação em *JavaScript* mais simples e fácil, tanto para programadores experientes como para iniciantes.

Ainda segundo Silva (2008, p. 18):

> *jQuery se destina a adicionar interatividade e dinamismo às páginas web, incrementando de forma progressiva e não obstrutiva a usabilidade, a acessibilidade e o design, enriquecendo a experiência do usuário.*
>
> *Use jQuery em sua página para:*
>
> - *Adicionar efeitos visuais e animações;*
> - *Acessar e manipular o DOM;*
> - *Buscar informações no servidor sem necessidade de recarregar a página;*
> - *Prover interatividade;*
> - *Alterar conteúdos;*
> - *Modificar apresentação e estilização;*
> - *Simplificar tarefas específicas de JavaScript;*
> - *Realizar outras tarefas relacionadas às descritas*

3.7. WBS – WORK BREAKDOWN STRUCTURE

Segundo o Guia PMI de Gerenciamento de Projetos (2004, p. 112):

> *A WBS, ou Estrutura Analítica de Projeto, é uma decomposição hierárquica orientada à entrega do trabalho a ser executado pela equipe do projeto, para atingir os objetivos do projeto e criar as entregas necessárias. A EAP organiza e define o escopo total do projeto. A EAP subdivide o trabalho do projeto em partes menores e mais facilmente gerenciáveis, em que cada nível descendente da EAP representa uma definição cada vez mais detalhada do trabalho do projeto. É possível agendar, estimar custos, monitorar e controlar o trabalho planejado contido nos componentes de nível mais baixo da EAP, denominados pacotes de trabalho.*

A *Work Breakdown Structure* foi definida hierarquicamente de acordo com cada parte do projeto e suas partes terminais, como especificado no diagrama abaixo.

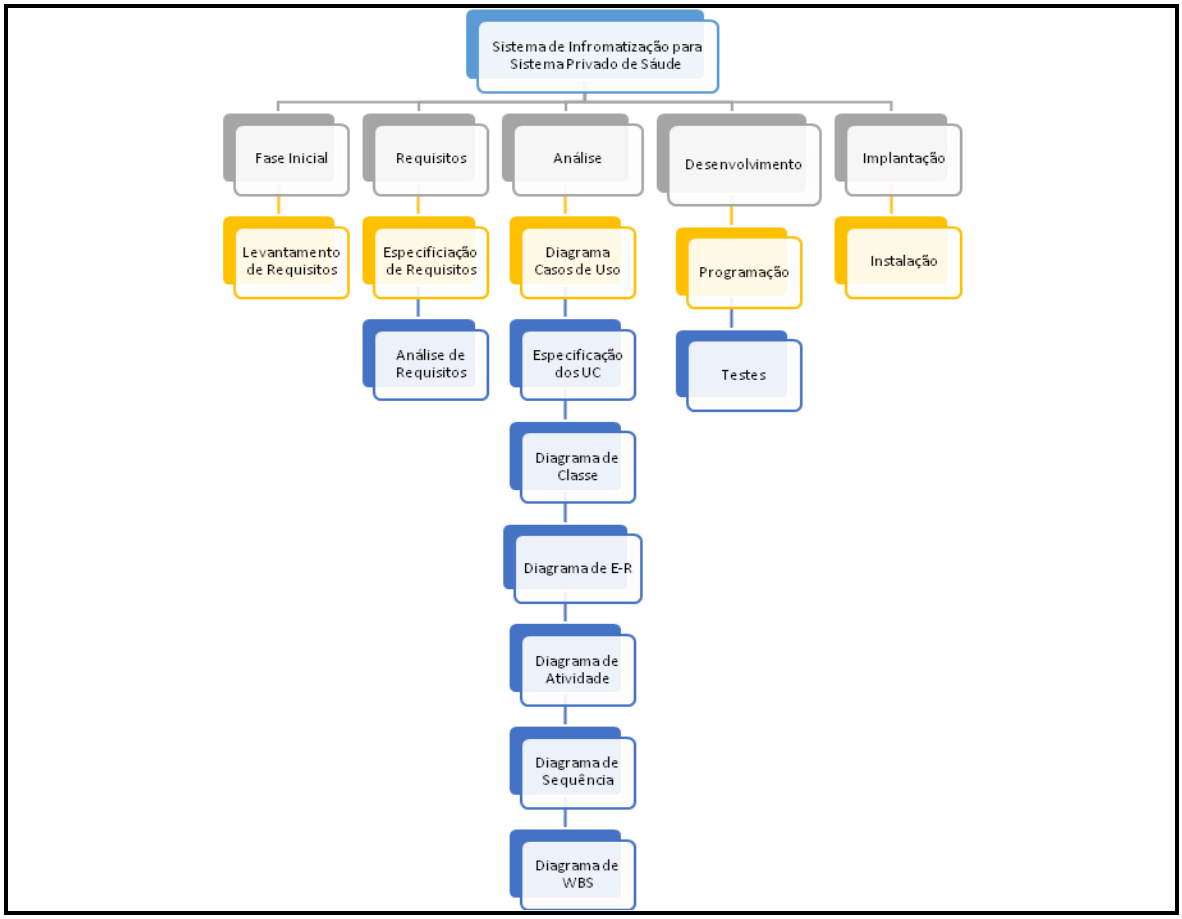


Figura 1 – Diagrama de WBS

3.8. SEQUENCIAMENTO DAS ATIVIDADES DEFINIDAS

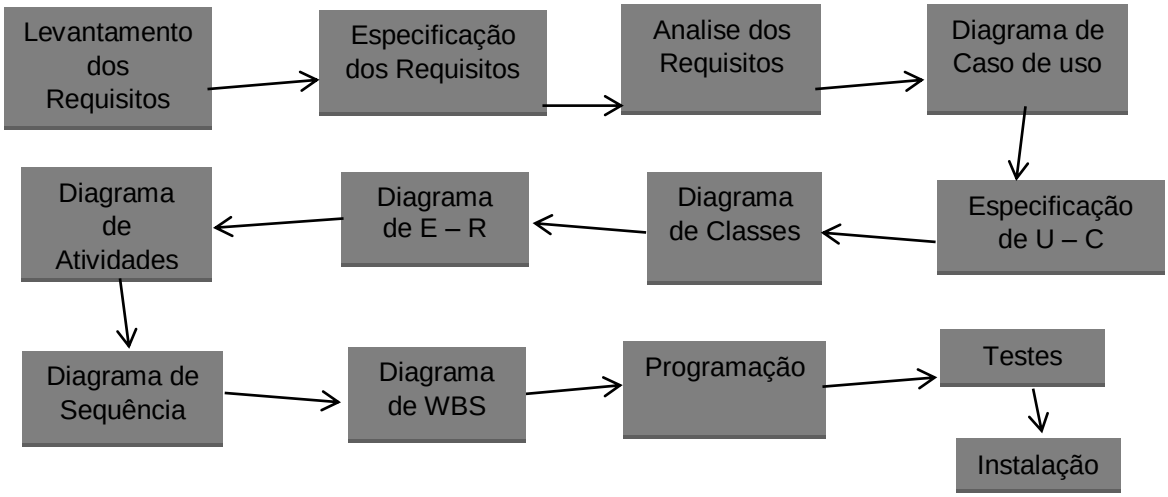


Figura 2 – Sequenciamento de Atividades

3.9. ESTIMATIVA DE DURAÇÃO DAS ATIVIDADES DEFINIDAS

ATIVIDADE	TOTAL DE DIAS	HORAS (DIA)	TOTAL DE HORAS
Levantamento de Requisitos	10	4 h	80 h
Especificação de Requisitos	10	2 h	20 h
Analise de Requisitos	10	1 h	10 h
Diagrama de Caso de Uso	10	1 h	10 h
Especificação de U-C	10	2 h	20 h
Diagrama de Classes	2	1 h	2 h
Diagrama de E-R	2	2 h	4 h
Diagrama de Atividades	3	2 h	6 h

Diagrama de Sequência	3	1 h	3 h
Diagrama de WBS	2	1 h	2 h
Programação	100	2 h	200 h
Testes	2	1 h	2 h
Instalação	3	1 h	3 h

Tabela 2 – Duração das Atividades (horas)

3.10. CRONOGRAMA DE REALIZAÇÃO DAS ATIVIDADES

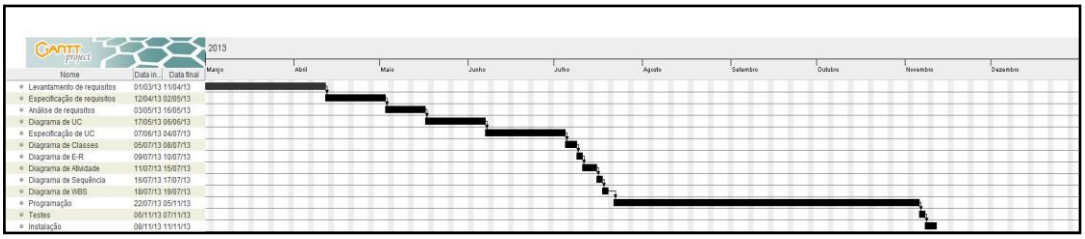


Figura 3 – Cronograma das Atividades (dias)

3.11. RECURSOS NECESSÁRIOS PARA O DESENVOLVIMENTO DO PROJETO

Para a elaboração, teste e implementação do projeto, será demonstrado a seguir os custos para todas essas atividades, desde recursos físicos, como computadores, impressoras, até recursos humanos, como contratação de pessoal, salário e outros. Levando em consideração ser um projeto de pequeno porte, a contratação de pessoal não será necessária, sendo que o próprio graduando executará todas as tarefas cabíveis a cada componente da equipe, reduzindo assim, o custo final do Projeto. A duração prevista para o projeto é de 8 meses, não sendo possível eventuais atrasos.

3.11.1. Recursos Computacionais

- *Windows 7 Home Premium*;
- Mi*crosoft Visual Studio 2012 Professional*;
- Banco de Dados *Microsoft SQL Server 2012*;
- *Astah Professional Academic*;
- *DB Designer Fork*;
- *Microsoft Office Home & Student 2013*.

3.11.2. Recursos Físicos

- 1 (um) *notebook*;
- 1 (uma) impressora;

3.11.3. Recursos Humanos

- 1 (um) Analista de Sistema

3.12. ESTIMATIVA DE CUSTOS

Recursos	**Valor**
1. Computacionais	
Windows 7 Home Premium (Vitalício)	R$ 309,00
Microsoft Visual Studio 2012 Professional (Vitalício)	R$ 2.414,67
Microsoft SQL Server 2012 (Vitalício)	(Incluído no preço do Visual Studio)
Astah Professional V6.7 Academic Annual License	$ 40,00 USD – R$ 89,80
DB Designer Fork	Freeware
Microsoft Office Home & Student 2013 (Vitalício)	R$ 239,00
2. Físicos	
Notebook LG A530	R$ 2.500,00
Impressora *Wireless* HP	R$ 299,00
Humanos	Valor
Analista de Sistema (em média R$ 17,23 por hora, trabalhando em regime CLT de 44 horas	R$ 30.324,00

semanais e 220 horas mensais, sendo o projeto para 8 meses)	

Tabela 3 – Estimativas de Custo

3.13. ORÇAMENTO DO PROJETO

O orçamento total do projeto está estimado em R$ 36.175,57. Levando em conta o projeto ser concluído no prazo determinado e não havendo atrasos, terá um custo de R$ 4.521,94 por mês.

4. ANÁLISE ORIENTADA A OBJETOS

Para que a análise do projeto seja realizada de maneira mais fácil, transparente e, de modo geral, mais correta possível, prevendo situações e problemas que poderiam ocorrer no decorrer do projeto, a análise orientada a objeto foi adotada como forma principal de documentação.

4.1. DEFINIÇÃO DE UML

Segundo Guedes (2010), UML, ou em português, Linguagem de Modelagem Unificada, é uma linguagem visual para modelagem de *softwares* baseada no paradigma da orientação a objetos, sendo possível aplicá-la, juntamente com seu propósito geral, a todos os domínios de aplicação, tornando-se nos últimos anos, a linguagem padrão de modelagem, sendo adotada internacionalmente pela indústria de engenharia de *software*.

Ainda segundo o autor, ela não é uma linguagem de programação, e sim uma linguagem de modelagem, cujo objetivo é auxiliar os engenheiros de *software* a definirem as características do sistema, tais como seus requisitos, comportamento, estrutura lógica, a dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento sobre o qual o sistema será implantado.

Segundo Lima (2006), em 1996, surgiu como resultado da união de três métodos de modelagem: o método de *Booch*, o método *OMT (Object Modeling Technique)* de Jacobson, e o método *OOSE (Object-Oriented Software Engineering)* de Rumbaugh, sendo eles os métodos de modelagem orientada a objetos mais populares entre os profissionais da área de desenvolvimento de software até meados da década de 1990. A união desses métodos contou com amplo apoio da *Rational Software*, incentivando e financiando-a. Esta união ficou conhecida popularmente como “Os Três Amigos”.

No início do projeto, existia somente a união do método de Booch ao OMT de Jacobson, o que resultou no lançamento do Método Unificado, no final de 1995. Logo em seguida, Rumbaugh juntou-se a Booch e Jacobson na *Rational Software*, sendo o seu método OOSE incorporado à nova metodologia.

A partir da primeira versão, muitas empresas atuantes na área de modelagem e desenvolvimento de *software* passaram a contribuir para o projeto, fornecendo sugestões para melhorar e ampliar a linguagem. Finalmente, em 1997, foi adotada pela *OMG (Object Management Group* ou Grupo de Gerenciamento de Objetos), como uma linguagem padrão de modelagem.

Em julho de 2005, foi lançada oficialmente a versão 2.0 da linguagem, encontrando-se atualmente na versão 2.2, existindo ainda a versão 2.3 em fase beta (GUEDES, 2010, p.19-20).

Segundo Larman (2007, p. 39):

> *A palavra visual na definição é um ponto chave – a UML é a notação diagramática padrão, de fato, para desenhar ou apresentar figuras (com algum texto) relacionadas a software – principalmente software Orientado a Objetos.*

4.2. DIAGRAMA DE CASOS DE USO (USE-CASE)

Segundo Guedes (2010, p. 31):

> *O diagrama de casos de uso é o diagrama mais geral e informal da UML, utilizado normalmente nas fases de levantamento e análise de requisitos do sistema, embora venha a ser consultado durante todo o processo de modelagem e possa servir de base para outros diagramas. [...] Procura*

identificar os atores (usuários, outros sistemas ou até mesmo algum hardware especial) que utilizarão de alguma forma o software, bem como o serviço, ou seja, as funcionalidades que o sistema disponibilizará aos atores, conhecidas nesse diagrama como casos de uso.

4.2.1. Caso de Uso Geral

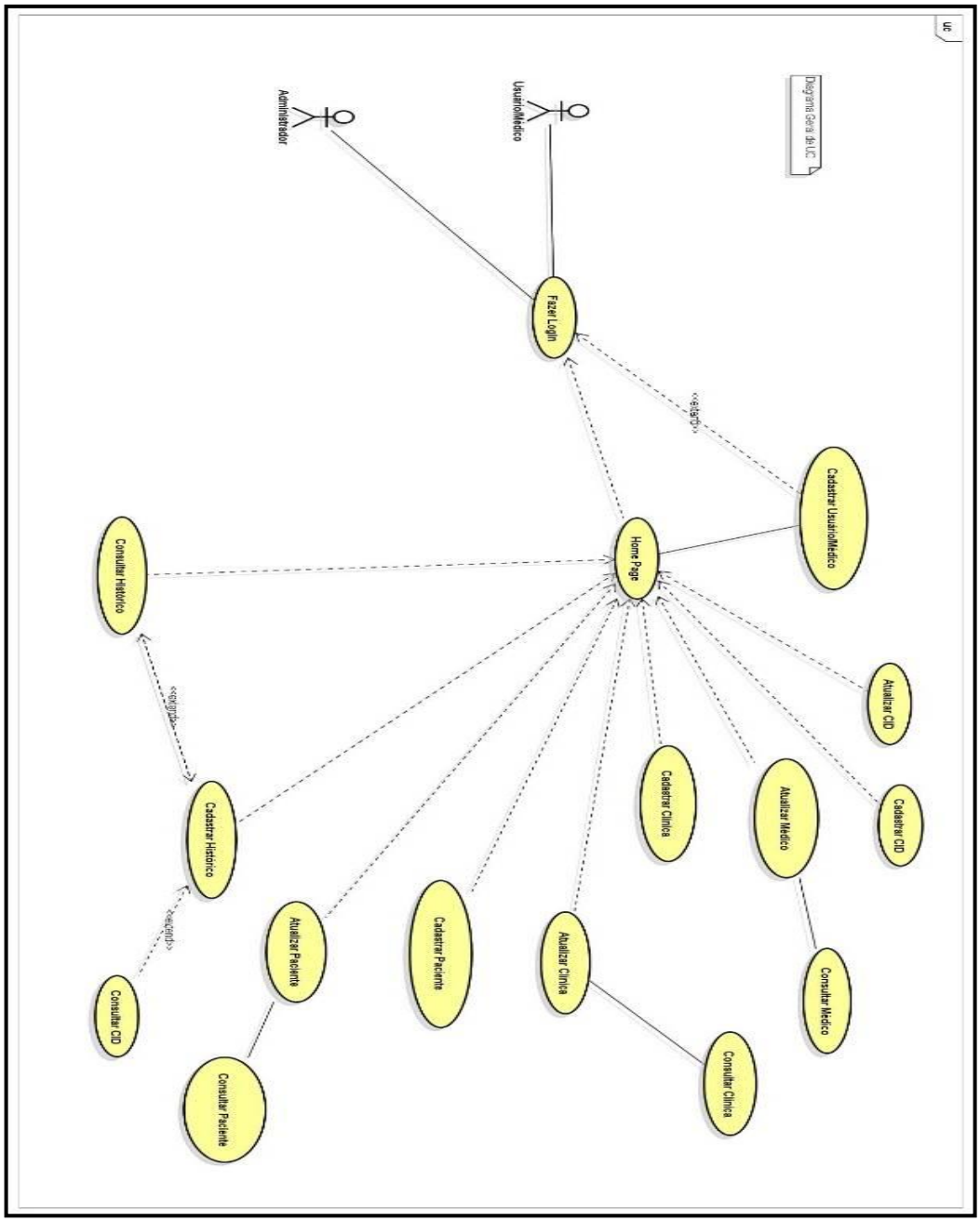


Figura 4 – Caso de Uso geral

4.2.2. Caso de uso Fazer Login

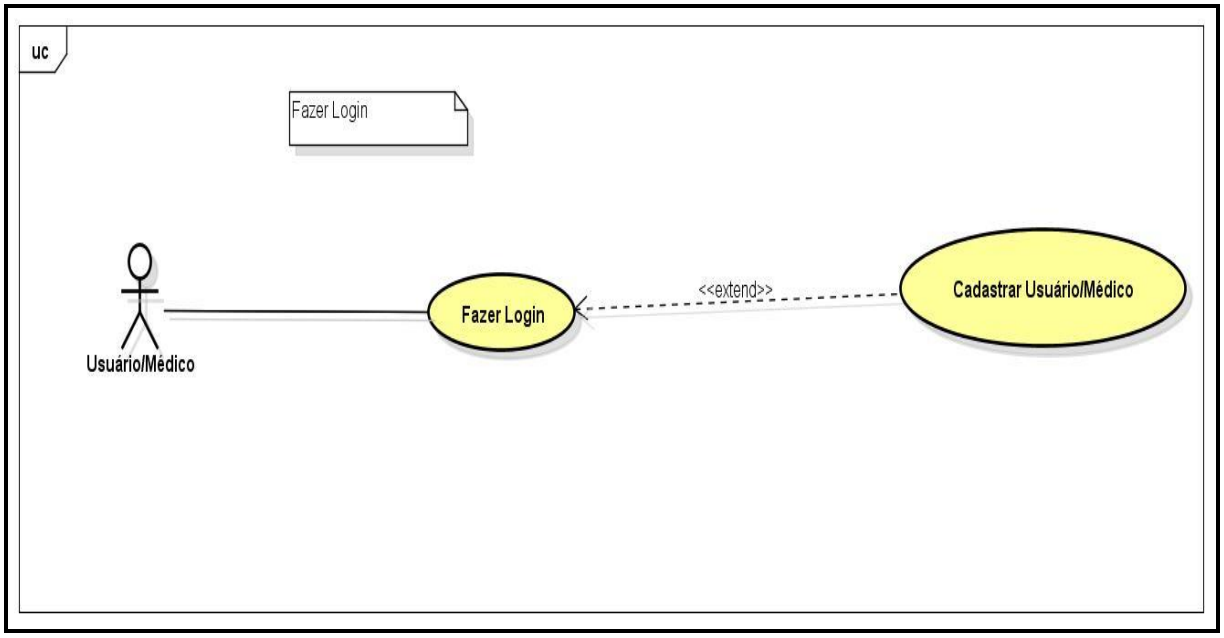


Figura 5 – Caso de Uso Fazer Login

Nome do Caso de Uso	Fazer Login
Caso de Uso Geral	Cadastrar Medico
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso descreve as etapas percorridas para fazer login no sistema
Pré-Condições	O médico deverá ter efetuado seu cadastro no sistema antes de efetuar o login
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O usuário Informa o nome de Usuário e Senha e clica no botão Login	
	2. O sistema entre na página Principal
Restrições/Validações	1. O nome de usuário e senha precisam estar cadastrados no Banco de Dados do Sistema
Fluxo Alternativo – Cadastrar Usuário/Médico	
Ações do Ator	Ações do Sistema
	1. Se usuário do Sistema não estiver cadastrado, executar Caso de Uso Cadastrar Médico, para que o mesmo possa ser cadastrado
Fluxo de Exceção I – nome de Usuário Incorreto	

Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que o nome de Usuário está incorreto
Fluxo de Exceção II – senha incorreta	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando ao usuário que a senha está incorreta
Fluxo de Exceção III – usuário não cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem avisando que não existe este usuário cadastrado

Tabela 4 – Descrição do Caso de Uso Fazer Login

4.2.3. Entrar na Home

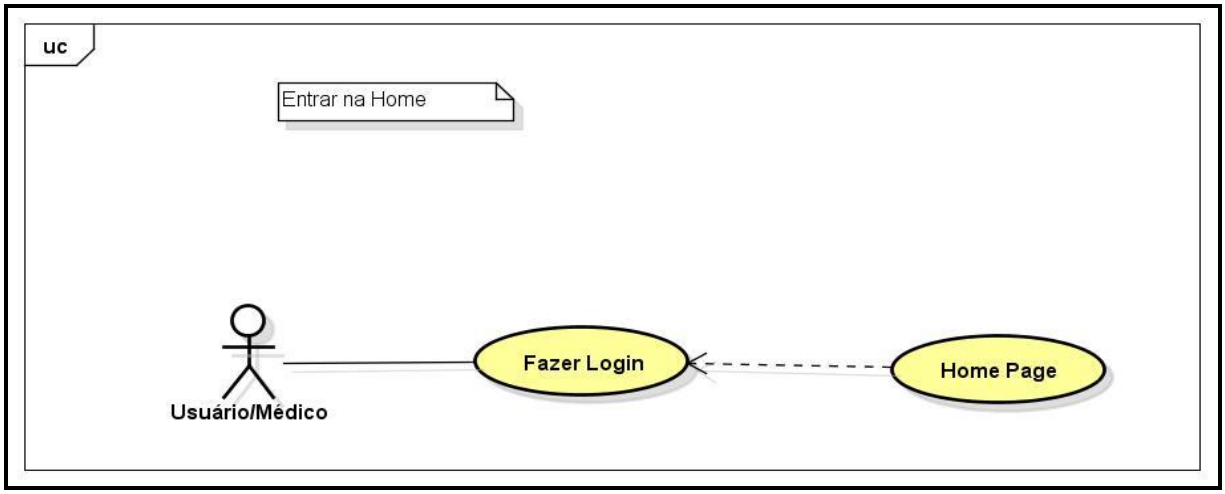


Figura 6 – Caso de Uso Entrar na Home

Nome do Caso de Uso	Entrar na Homepage
Caso de Uso Geral	Fazer Login
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso serve para especificar a Homepage do Sistema
Pré-Condições	O Médico deverá estar logado no Sistema
Pós-Condições	Dar acesso aos casos de uso existente
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Médico/Usuário escolhe qual Caso de	

Uso executar através de botões no Menu Principal	
	2. O sistema executa o Caso de Uso de acordo com a escolha do Médico no Menu
Restrições/Validações	1. O Usuário/Médico deve estar logado no Sistema.
Fluxo Alternativo	
Ações do Ator	Ações do Sistema
Fluxo de Exceção	
Ações do Ator	Ações do Sistema

Tabela 5 – Descrição do Caso de Uso Entrar na Home

4.2.4. Cadastrar Médico

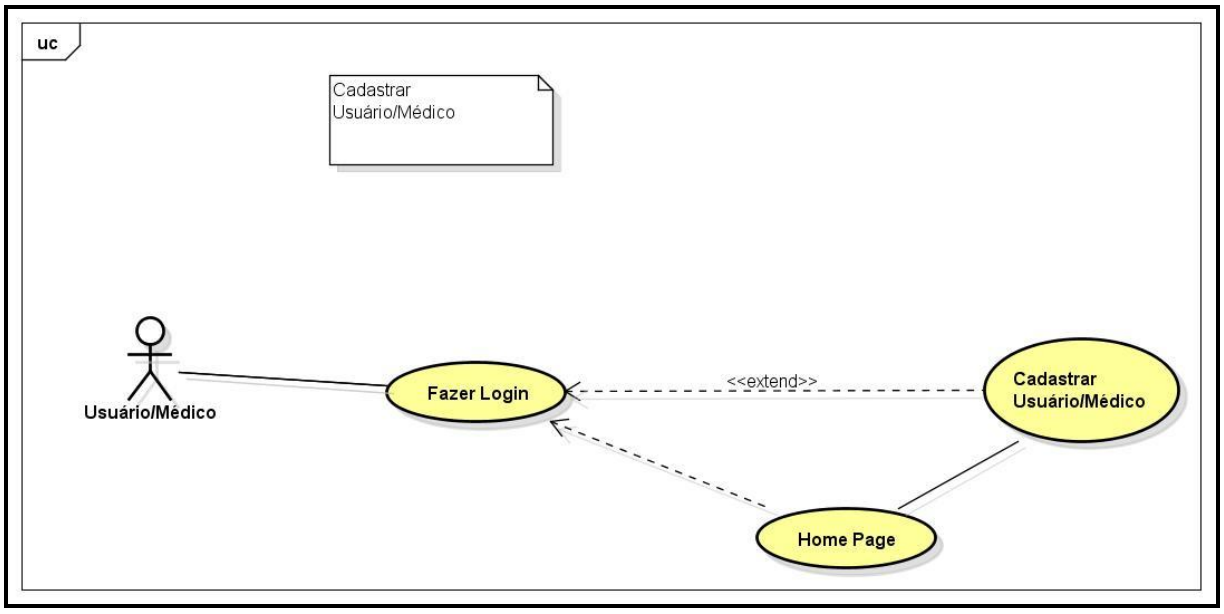


Figura 7 – Caso de Uso Cadastrar Usuário-Médico

Nome do Caso de Uso	Cadastrar Médico
Caso de Uso Geral	Acessar Homepage
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso é para realizar o cadastramento do Usuário/Médico no Sistema
Pré-Condições	Usuário/Médico deverá conter alguns documentos importantes em

	mãos, tais como CRM, documentos pessoais e etc.
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Usuário/Médico informa os dados necessários, bem como o nome de Usuário e senha. Após o formulário preenchido, clica no botão cadastrar	
	2. O sistema entra na página Home e envia um e-mail para o e-mail cadastrado informando o nome de Usuário e senha
Restrições/Validações	1. Todos os dados devem estar preenchidos corretamente, bem como o nome de usuário e a senha conterem 10 caracteres cada
Fluxo de Exceção I – campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário
Fluxo de Exceção II – usuário e senha fora dos padrões	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando ao usuário que os campos usuário e senha não estão corretos
Fluxo de Exceção III – usuário/médico já cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem avisando que o usuário já está cadastrado

Tabela 6 – Descrição do Caso de Uso Cadastrar Médico

4.2.5. Cadastrar CID

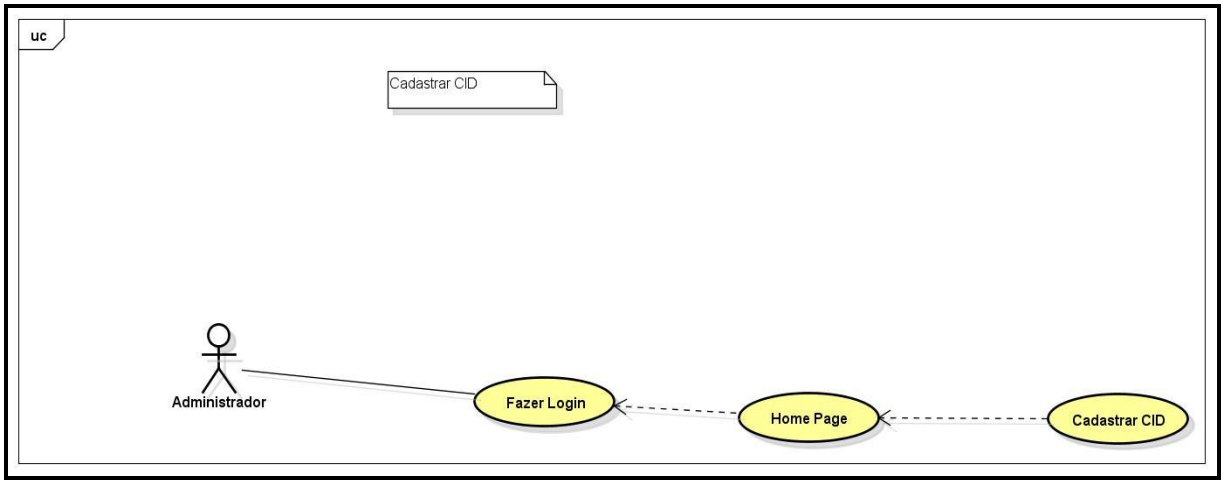


Figura 8 – Caso de Uso Cadastrar CID

Nome do Caso de Uso	Cadastrar CID
Caso de Uso Geral	
Ator Principal	Administrador
Atores Secundários	
Resumo	Este caso de uso descreve as etapas do cadastramento do CID (Código Internacional de Doenças)
Pré-Condições	O Administrador deverá estar logado no Sistema
Pós-Condições	Somente o Administrador consegue executar este caso de uso
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Administrador entre com as informações do CID	
	2. O sistema gera lista de CID cadastrados
Restrições/Validações	1. Não podem existir duplicidade de CID no Sistema
Fluxo Alternativo	
Ações do Ator	Ações do Sistema
Fluxo de Exceção	
Ações do Ator	Ações do Sistema
Fluxo de Exceção	
Ações do Ator	Ações do Sistema

Tabela 7 – Descrição do Caso de Uso Cadastrar CID

4.2.6. Cadastrar Clínica

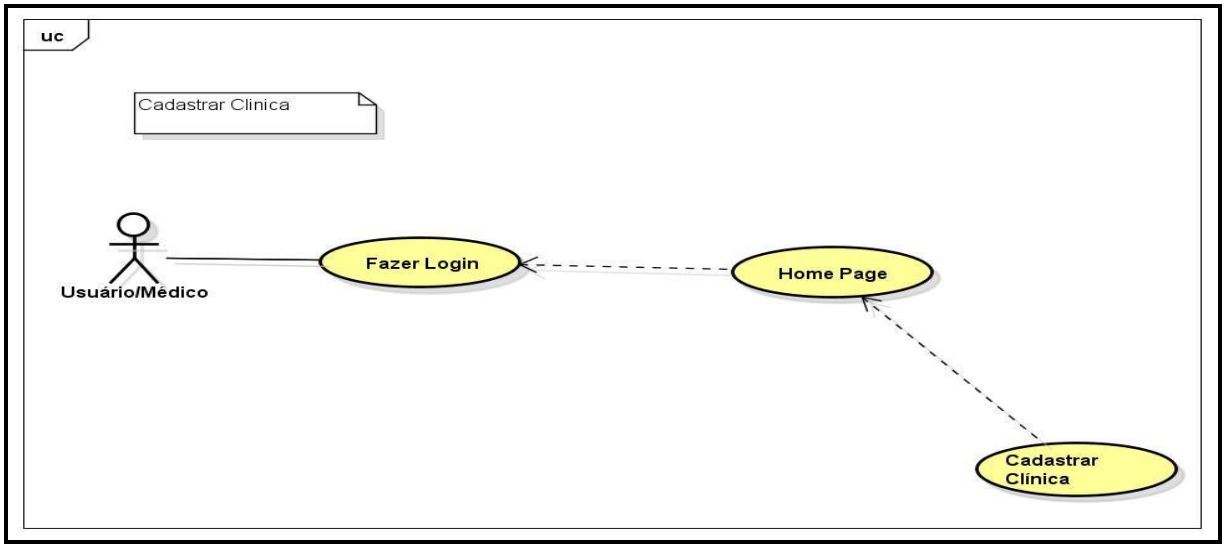


Figura 9 – Caso de Uso Cadastrar Clínica

Nome do Caso de Uso	Cadastrar Clínica
Caso de Uso Geral	Acessar a Homepage
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso é para realizar o cadastramento de Clínicas Médicas no Sistema
Pré-Condições	Usuário/Médico deverá conter alguns documentos importantes em mãos, tais como CNPJ da clínica, informações do proprietário da mesma e etc
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Usuário/Médico informa os dados necessários da clínica em que está atendendo. Após o formulário preenchido, clica no botão cadastrar	
	2. O sistema entra na página Home e envia um e-mail para o e-mail cadastrado da clínica informando que a mesma foi cadastrada com sucesso
Restrições/Validações	1. Todos os dados devem estar preenchidos corretamente, bem como o nome do proprietário da clínica, o CNPJ e a inscrição municipal da mesma

Fluxo de Exceção I – campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário
Fluxo de Exceção II – clínica já cadastrada	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem avisando que a clínica já está cadastrada

Tabela 8 – Descrição do Caso de Uso Cadastrar Clínica

4.2.7. Cadastrar Paciente

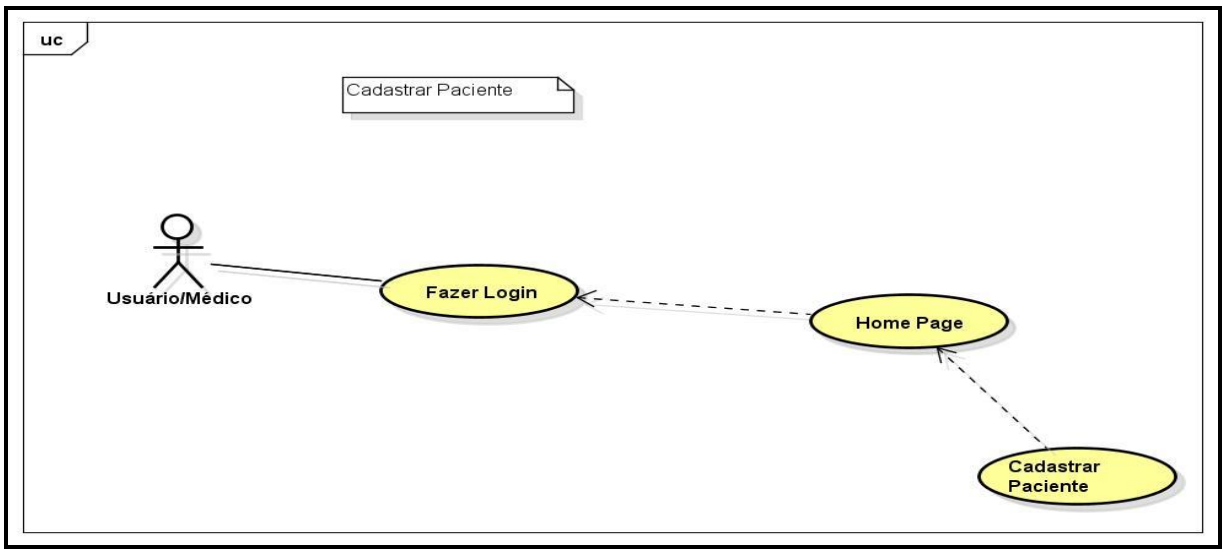


Figura 10 – Caso de Uso Cadastrar Paciente

Nome do Caso de Uso	Cadastrar Paciente
Caso de Uso Geral	Acessar a Homepage
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso é para realizar o cadastramento do paciente no sistema
Pré-Condições	O paciente deverá apresentar os documentos necessários para o cadastro, bem como receitas de medicamentos que estão em uso ou que foram usados
Pós-Condições	Somente após este cadastro, será

	chamado o Caso de Uso Cadastrar Histórico
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O paciente informa ao Usuário/Médico os dados necessários, bem como dados de medicamentos e consultas anteriores. Após o formulário preenchido, clica-se no botão cadastrar	
	2. O sistema entra na página Home e envia um e-mail para o e-mail do usuário informando que o cadastro do mesmo na clínica específica foi efetuado com sucesso
Restrições/Validações	1. Todos os dados devem estar preenchidos corretamente, bem como, no primeiro cadastro, o nome da clínica e o nome do médico
Fluxo de Exceção I – campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário
Fluxo de Exceção II – paciente já cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando ao usuário que o paciente já está cadastrado

Tabela 9 – Descrição do Caso de Uso Cadastrar Paciente

4.2.8. Cadastrar Histórico

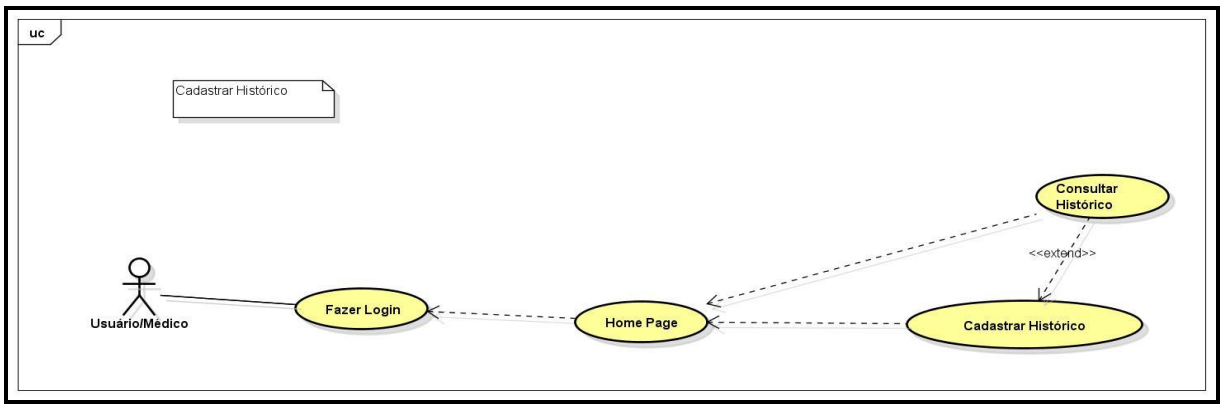


Figura 11 – Caso de Uso Cadastrar Histórico

Nome do Caso de Uso	Cadastrar Histórico
Caso de Uso Geral	Consultar Histórico
Ator Principal	Médico
Resumo	Este caso de uso é para realizar o cadastramento do Histórico do paciente no sistema
Pré-Condições	O paciente deverá apresentar as receitas de medicamentos que foram ministrados anteriormente, bem como tempo de tratamento e CID
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O médico recebe a documentação do paciente e insere no cadastro de histórico	
	2. O Médico cadastra o Histórico do paciente, informando os dados necessários, bem como o CID, remédios prescritos e etc.
Restrições/Validações	1. Somente o médico poderá executar este caso de uso
	2. Cada Histórico equivale a uma Consulta, não podendo alterá-lo
Fluxo Alternativo I – Paciente não Cadastrado	
Ações do Ator	Ações do Sistema
1. O Médico busca o código do Paciente para continuar o cadastro do Histórico e o mesmo não é encontrado. O mesmo clica no botão Cadastrar Paciente	
	2. O Sistema executa o caso de uso Cadastrar Paciente
Fluxo Alternativo II – Clínica não Cadastrada	
Ações do Ator	Ações do Sistema
1. O Médico busca o código da Clínica e a mesma não é encontrada. O Médico clina no botão Cadastrar Clínica.	
	2. O Sistema executa o caso de uso Cadastrar Clínica
Fluxo de Exceção I – Campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário

Tabela 10 – Descrição do Caso de Uso Cadastrar Histórico

4.2.9. Consultar CID

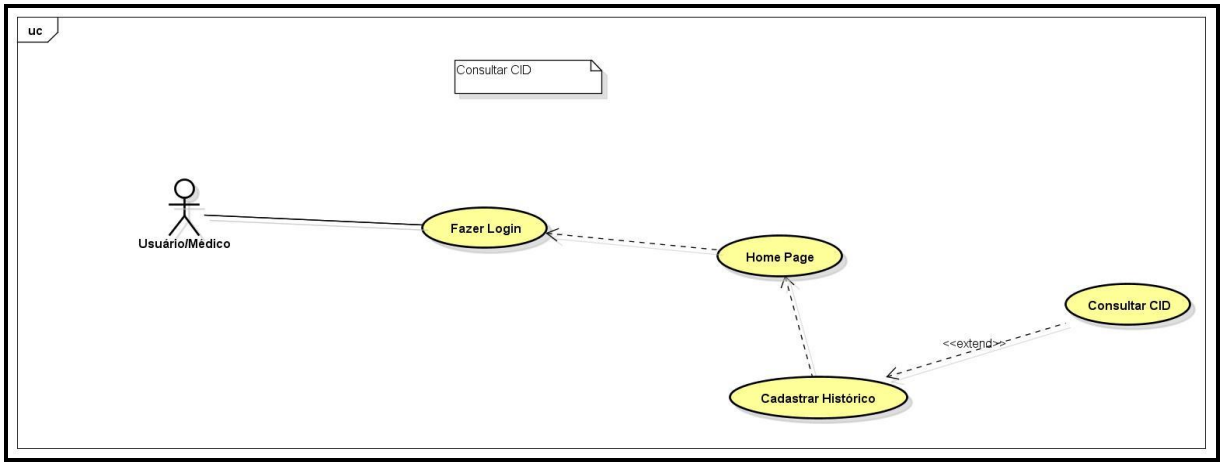


Figura 12 – Caso de Uso Consultar CID

Nome do Caso de Uso	Consultar CID
Caso de Uso Geral	Cadastrar Histórico
Ator Principal	Usuário/Médico
Atores Secundários	Secretária
Resumo	Este caso de uso descreve as etapas do consulta do CID (Código Internacional de Doenças)
Pré-Condições	O Usuário/Médico deve estar logado no Sistema.
Pós-Condições	O Usuário/Médico deverá selecionar outros campos para dar continuidade ao caso de uso Cadastrar Histórico
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Usuário/Médico seleciona o Código do CID	
	2. O sistema puxa a descrição do CID
Restrições/Validações	1. Não podem existir duplicidade de CID no Sistema
Fluxo Alternativo	
Ações do Ator	Ações do Sistema
Fluxo de Exceção	
Ações do Ator	Ações do Sistema
Fluxo de Exceção	
Ações do Ator	Ações do Sistema

Tabela 11 – Descrição do Caso de Uso Consultar CID

4.2.10. Consultar Clínica

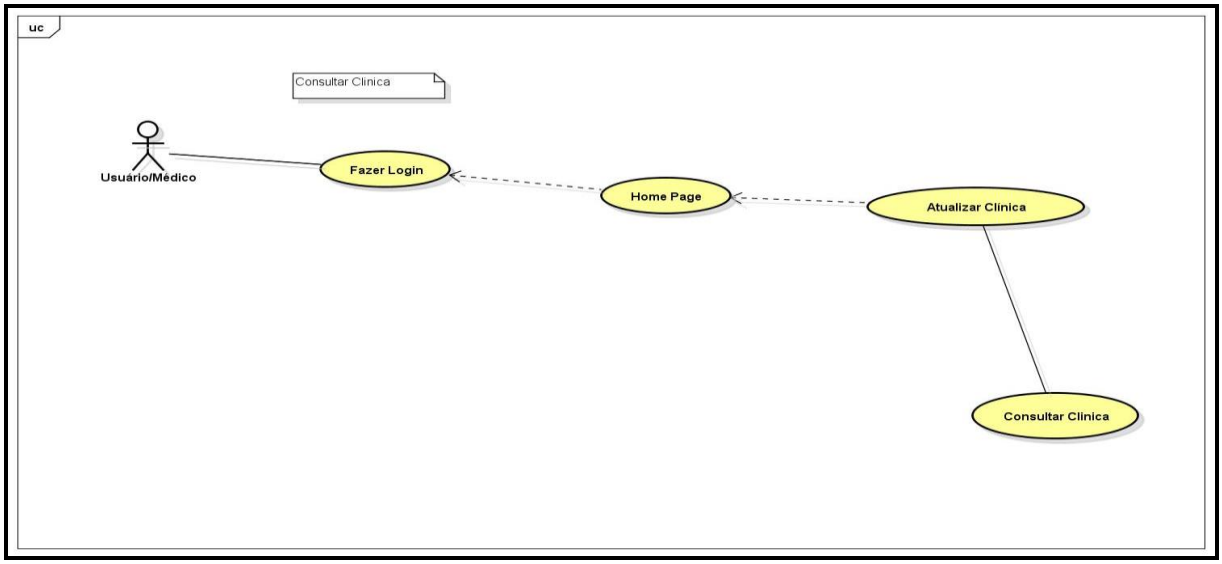


Figura 13 – Caso de Uso Consultar Clínica

Nome do Caso de Uso	Consultar Clínica
Caso de Uso Geral	Cadastrar Histórico
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso descreve as etapas percorridas para consultar as Clínica cadastradas no Sistema
Pré-Condições	O Médico deverá estar logado no Sistema
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Médico seleciona o nome da clínica	
	2. O sistema busca o código e a especialidade da clínica selecionada
Restrições/Validações	
Fluxo Alternativo I – Cadastrar Clínica	
Ações do Ator	Ações do Sistema
1. Se a Clínica não estiver cadastrada no Sistema, o Médico deverá executar o caso de uso Cadastrar Clínica	

Tabela 12 – Descrição do Caso de Uso Consultar Clínica

4.2.11. Consultar Histórico

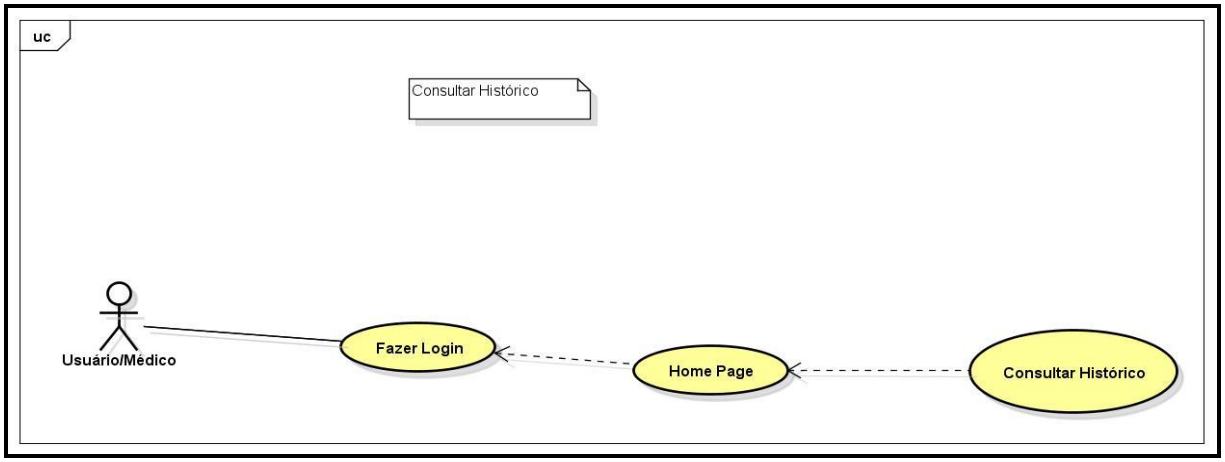


Figura 14 – Caso de Uso Consultar Histórico

Nome do Caso de Uso	Consultar Histórico
Caso de Uso Geral	Acessar a Homepage
Ator Principal	Médico
Atores Secundários	
Resumo	Este caso de uso é para realizar uma prévia consulta do Histórico do Paciente
Pré-Condições	O Médico deverá saber o código ou CPF do Paciente
Pós-Condições	O Sistema executa o caso de uso Cadastrar Histórico
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O médico realiza a consulta do Histórico do Paciente	
	2. O Sistema retorna uma lista de Históricos do Paciente consultado.
Restrições/Validações	1. Somente o médico poderá executar este caso de uso.
Fluxo Alternativo I – Paciente não Cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema executa o caso de uso Cadastrar Paciente.
Fluxo Alternativo II – Clínica não Cadastrada	
Ações do Ator	Ações do Sistema
	1. O Sistema executa o caso de uso Cadastrar Clínica.
Fluxo Alternativo III – Histórico não cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema executa o caso de uso Cadastrar

	Histórico.
Fluxo de Exceção I – Campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário.

Tabela 13 – Descrição do Caso de Uso Consultar Histórico

4.2.12. Consultar Paciente

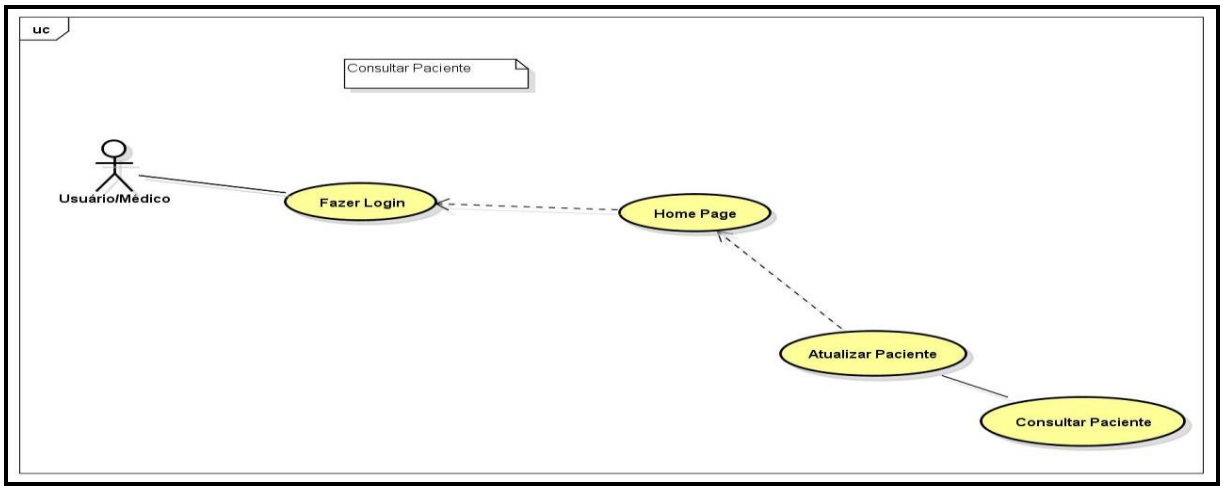


Figura 15 – Caso de Uso Consultar Paciente

Nome do Caso de Uso	Consultar Paciente
Caso de Uso Geral	Cadastrar Histórico
Ator Principal	Médico
Atores Secundários	
Resumo	Este caso de uso descreve as etapas percorridas para consultar o cadastro do Paciente
Pré-Condições	O Médico deverá estar logado no Sistema
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Médico seleciona o nome do paciente	
	2. O Sistema carrega as informações necessárias para o preenchimento correto do Histórico.
Restrições/Validações	
Fluxo Alternativo – Cadastrar Usuário/Médico	
Ações do Ator	Ações do Sistema
Fluxo de Exceção I – Datas em Branco	
Ações do Ator	Ações do Sistema

	1. O Sistema retorna mensagem de erro, informando que os campos relacionados as datas não podem estar em branco

Tabela 14 – Descrição do Caso de Uso Consultar Paciente

4.2.13. Consultar Médico

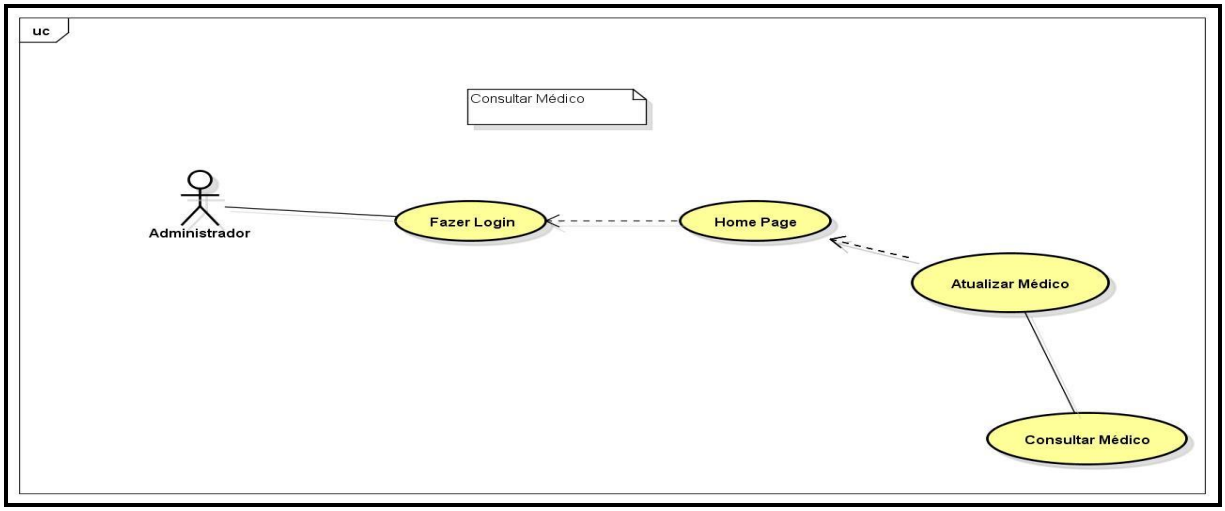


Figura 16 – Caso de Uso Consultar Médico

Nome do Caso de Uso	Consultar Medico
Caso de Uso Geral	Cadastrar Histórico
Ator Principal	Administrador
Atores Secundários	Médico, Secretária
Resumo	Este caso de uso descreve as etapas percorridas para Consultar o Médico
Pré-Condições	O Administrador/Médico deverá estar logado no Sistema
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Médico/Administrador seleciona o nome do Médico	
	2. O sistema busca as informações do médico no Banco de Dados.
Restrições/Validações	1. Somente o Administrador terá acesso ao caso de uso Cadastrar Médico
Fluxo Alternativo – Cadastrar Usuário/Médico	
Ações do Ator	Ações do Sistema
1. Se o Usuário/Médico não estiver cadastrado no Sistema, o Administrador	

deverá cadastra-lo	

Tabela 15 – Descrição do Caso de Uso Consultar Médico

4.2.14. Atualizar CID

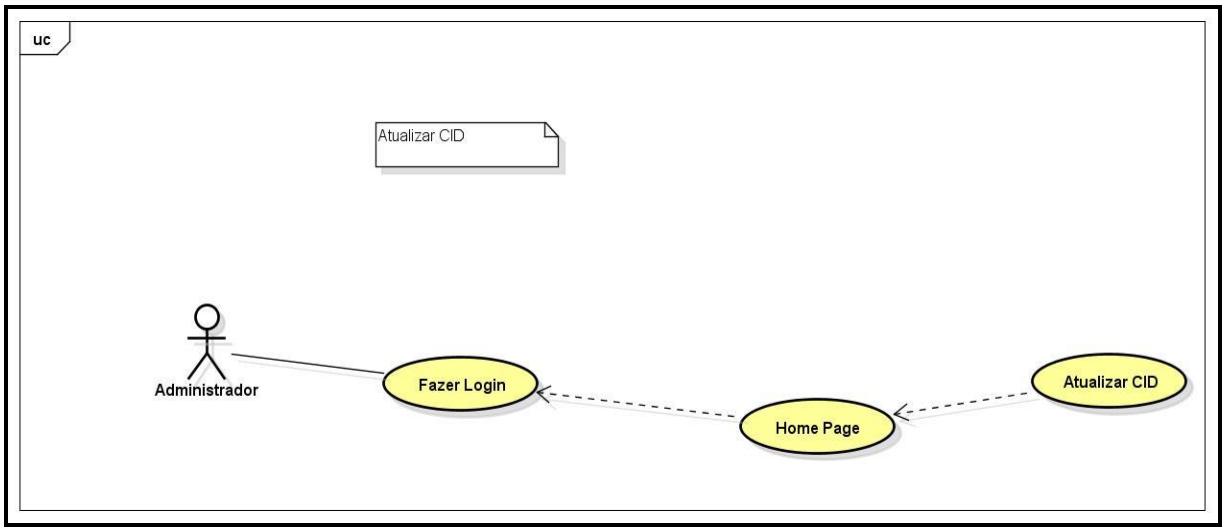


Figura 17 – Caso de Uso Atualizar CID

Nome do Caso de Uso	Atualizar CID
Caso de Uso Geral	
Ator Principal	Administrador
Atores Secundários	
Resumo	Este caso de uso descreve as etapas do atualização do CID (Código Internacional de Doenças)
Pré-Condições	O Administrador deverá estar logado no Sistema
Pós-Condições	Somente o Administrador consegue executar este caso de uso
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Administrador seleciona o código do CID a ser atualizado	
	2. O sistema retorna os dados do CID
3. O Administrador corrige os dados necessários	
	4. O Sistema salva os dados do CID
Restrições/Validações	1. Não podem existir duplicidade de CID no Sistema
Fluxo Alternativo	
Ações do Ator	Ações do Sistema

Fluxo de Exceção	
Ações do Ator	Ações do Sistema
Fluxo de Exceção	
Ações do Ator	Ações do Sistema

Tabela 16 – Descrição do Caso de Uso Atualizar CID

4.2.15. Atualizar Clínica

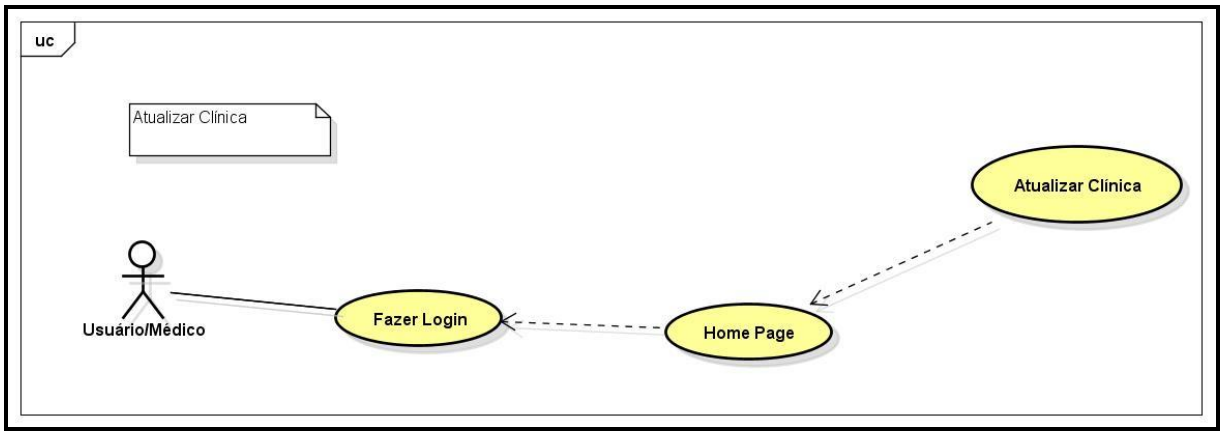


Figura 18 – Caso de Uso Atualizar Clínica

Nome do Caso de Uso	Atualizar Clínica
Caso de Uso Geral	Acessar a Homepage
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso é para realizar uma atualização dos dados do Paciente
Pré-Condições	O Médico deverá estar logado no Sistema
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Médico busca qual Paciente deseja atualizar os dados	
	2. O Sistema retorna os dados do Paciente selecionado
3. O Médico informa os dados que serão atualizados e clica em Atualizar	
	4. O Sistema atualiza os dados modificados
Restrições/Validações	1. Somente o Médico poderá executar este caso de uso.
Fluxo Alternativo I – Clínica não cadastrada	
Ações do Ator	Ações do Sistema

	1. O Sistema executa o caso de uso Cadastrar Clinica
Fluxo de Exceção I – Campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário.

Tabela 17 – Descrição do Caso de Uso Atualizar Clínica

4.2.16. Atualizar Médico

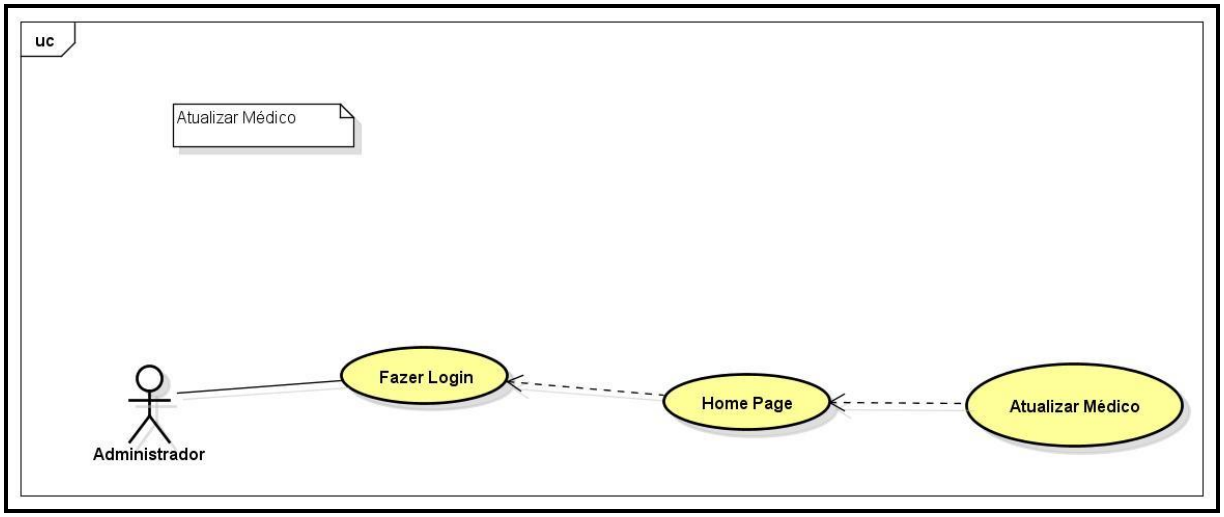


Figura 19 – Caso de Uso Atualizar Médico

Nome do Caso de Uso	Atualizar Médico
Caso de Uso Geral	Acessar a Homepage
Ator Principal	Administrador
Atores Secundários	
Resumo	Este caso de uso é para realizar uma atualização dos dados do Médico
Pré-Condições	O Administrador deverá estar logado no Sistema
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Administrador busca qual médico deseja atualizar os dados	
	2. O Sistema retorna os dados do Médico selecionado
3. O Administrador informa os dados que serão atualizados e clica em Atualizar	

	4. O Sistema atualiza os dados modificados
Restrições/Validações	1. Somente o Administrador poderá executar este Caso de Uso
Fluxo Alternativo I – Médico não Cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema executa o caso de uso Cadastrar Médico
Fluxo de Exceção I – Campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário.

Tabela 18 – Descrição do Caso de Uso Atualizar Médico

4.2.17. Atualizar Paciente

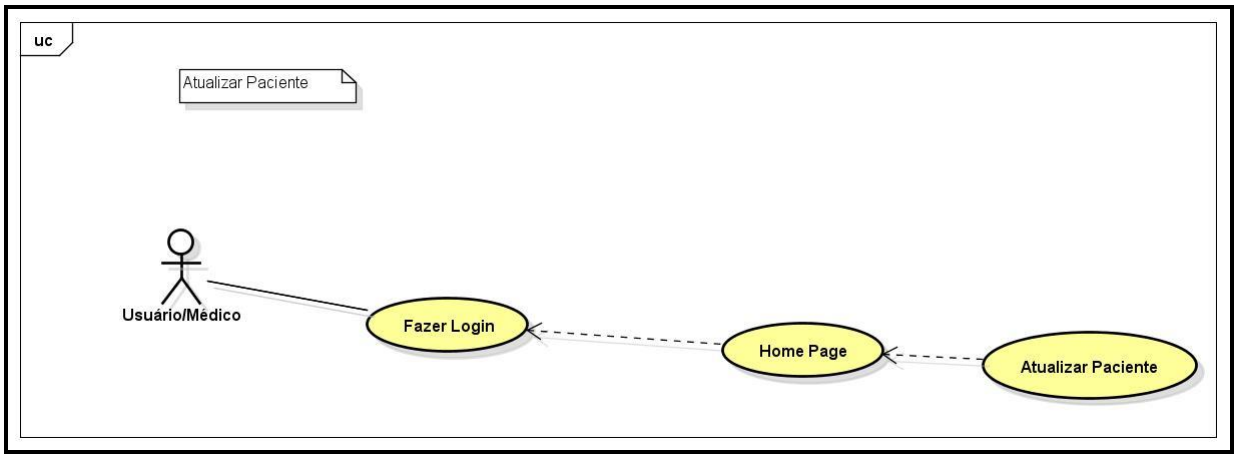


Figura 20 – Caso de Uso Atualizar Paciente

Nome do Caso de Uso	Atualizar Paciente
Caso de Uso Geral	Acessar a Homepage
Ator Principal	Médico
Atores Secundários	Secretária
Resumo	Este caso de uso é para realizar uma atualização dos dados do Paciente
Pré-Condições	O Médico deverá estar logado no Sistema
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. O Médico busca qual Paciente deseja atualizar os dados	
	2. O Sistema retorna os dados do Paciente

	selecionado
3. O Médico informa os dados que serão atualizados e clica em Atualizar	
	4. O Sistema atualiza os dados modificados
Restrições/Validações	1. Somente o Médico poderá executar este caso de uso.
Fluxo Alternativo I – Paciente não cadastrado	
Ações do Ator	Ações do Sistema
	1. O Sistema executa o caso de uso Cadastrar Paciente
Fluxo de Exceção I – Campos em branco	
Ações do Ator	Ações do Sistema
	1. O Sistema retorna mensagem de erro, informando que não podem existir campos em branco no formulário.

Tabela 19 – Descrição do Caso de Uso Atualizar Paciente

4.3. DIAGRAMA DE CLASSE

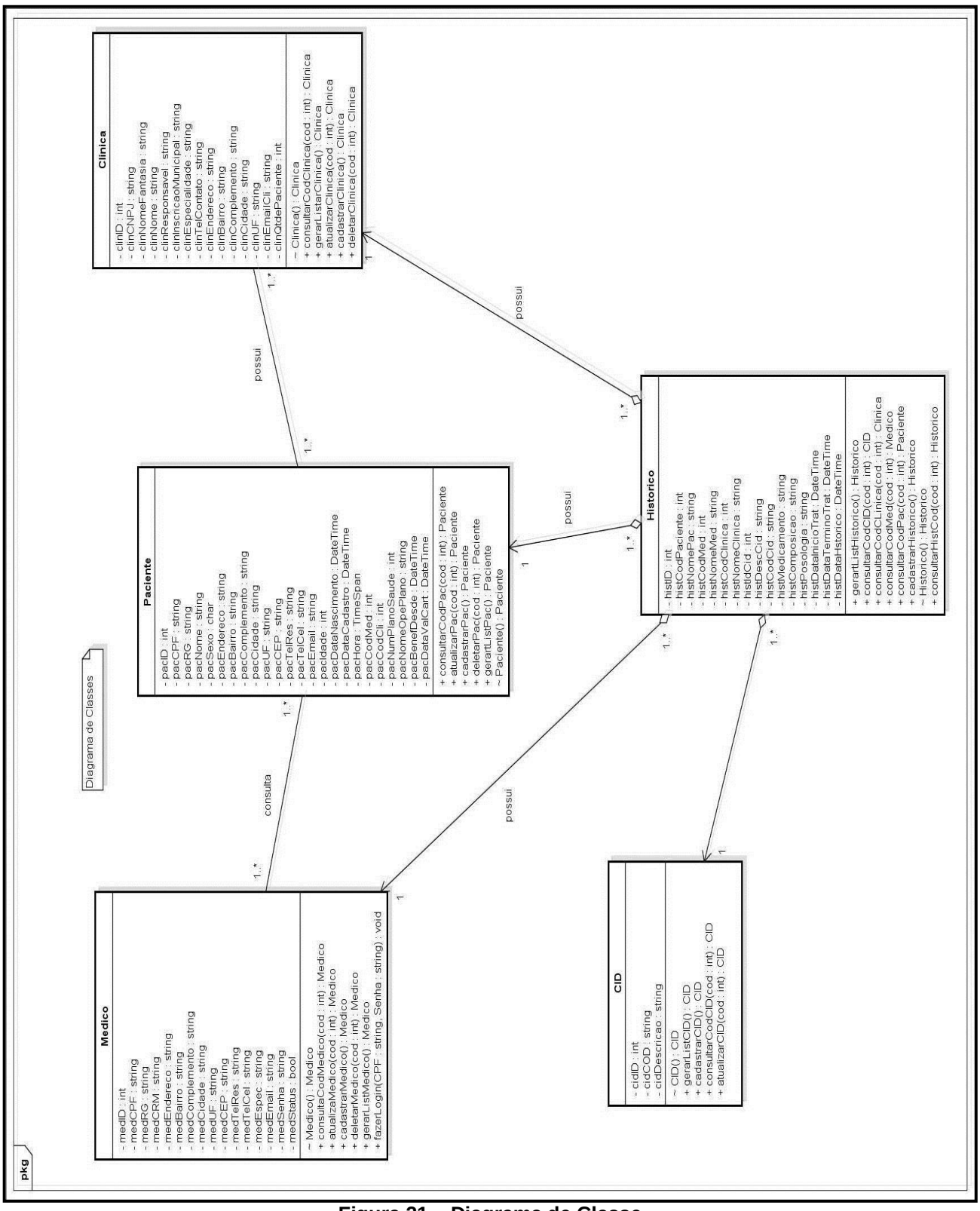


Figura 21 – Diagrama de Classe

4.4. DIAGRAMA DE ENTIDADE RELACIONAMENTO

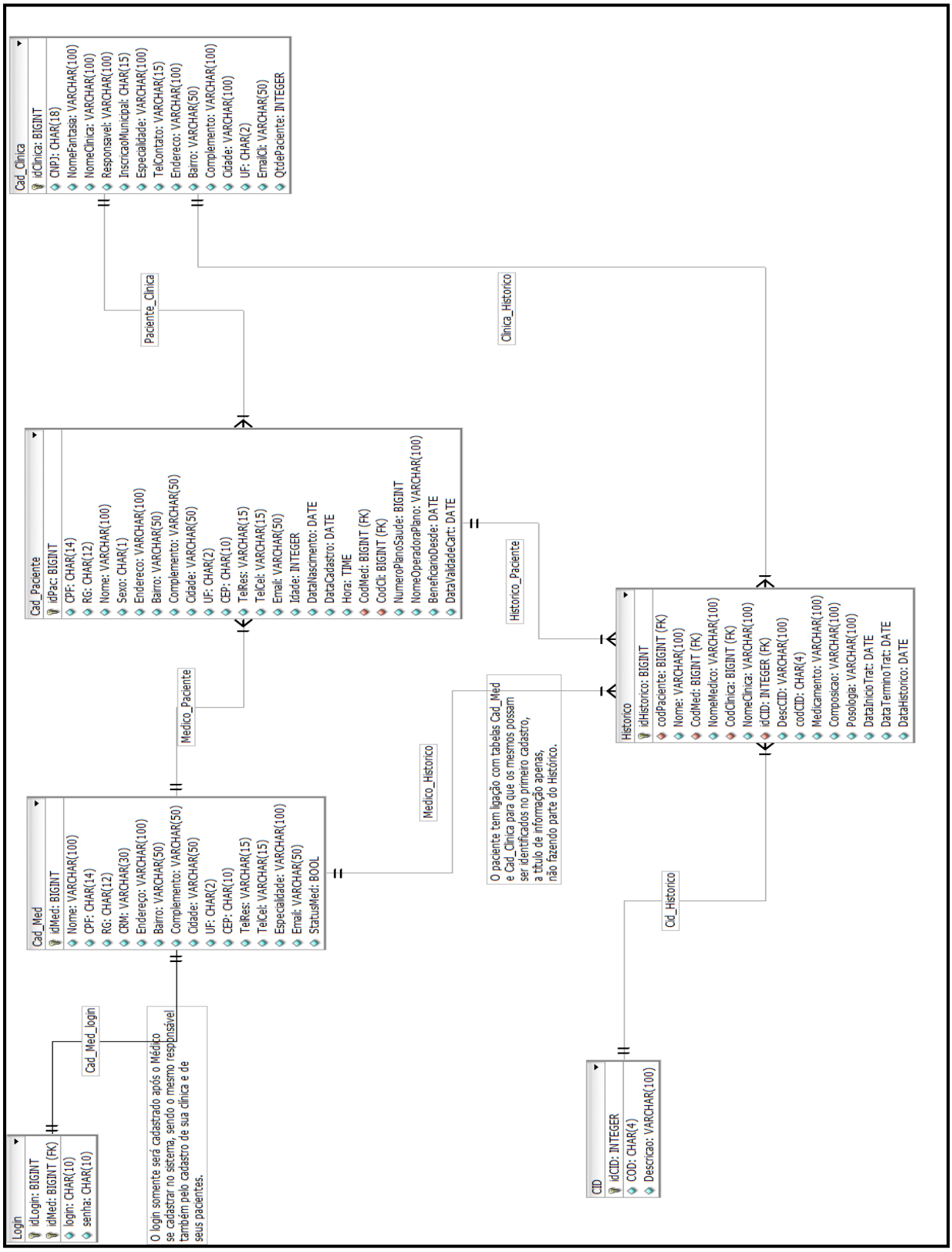


Figura 22 – Diagrama de Entidade-Relacionamento

4.5. DIAGRAMA DE SEQUÊNCIA

Segundo Guedes (2010), o diagrama de sequência é um diagrama comportamental, cuja principal função é determinar a sequência de eventos que ocorrem em um determinado processo, identificando assim, quais mensagens devem ser disparadas entre os elementos envolvidos e em qual sequência.

“Assim, determinar a ordem em que os eventos ocorrem, as mensagens que são enviadas, os métodos que serão chamados e como os objetos interagem dentro de um determinado processo é o objetivo principal desse diagrama” (GUEDES, 2010, p. 200).

4.5.1. Atualizar CID

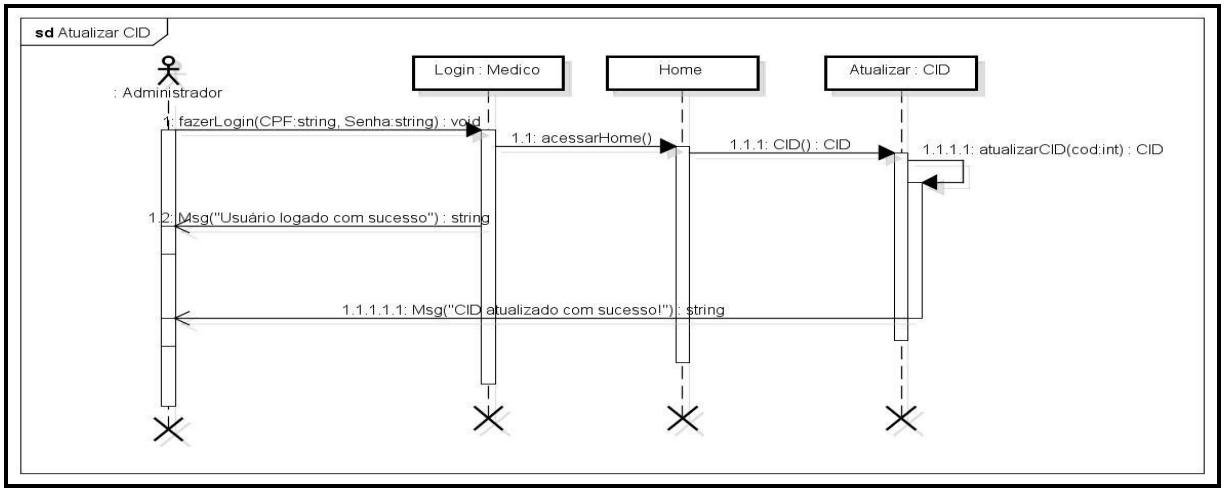


Figura 23 – Atualizar CID

4.5.2. Atualizar Clínica

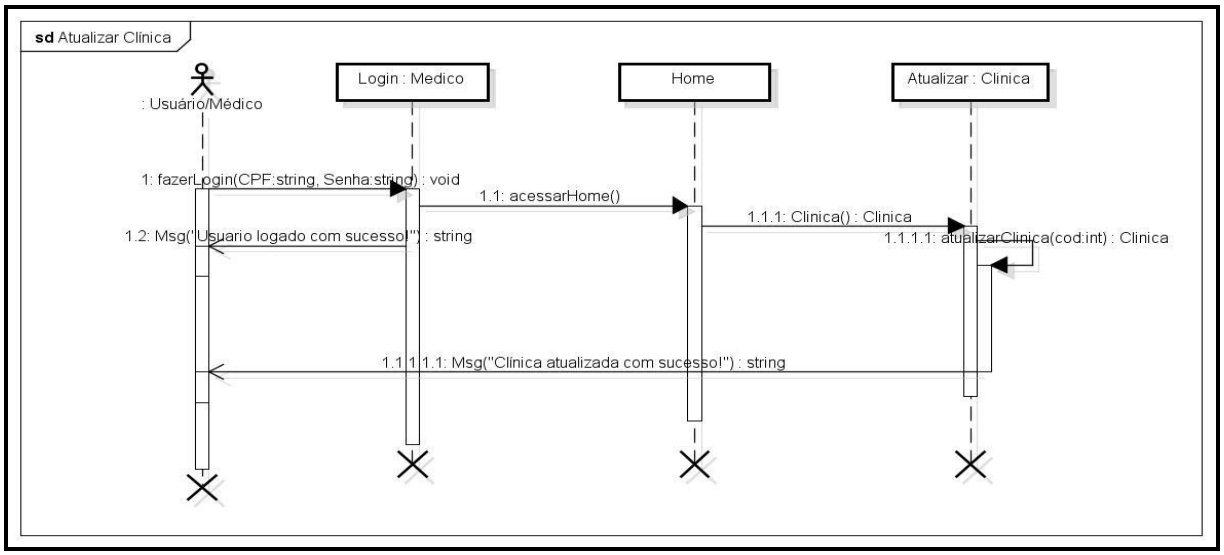


Figura 24 – Atualizar Clínica

4.5.3. Consultar Médico

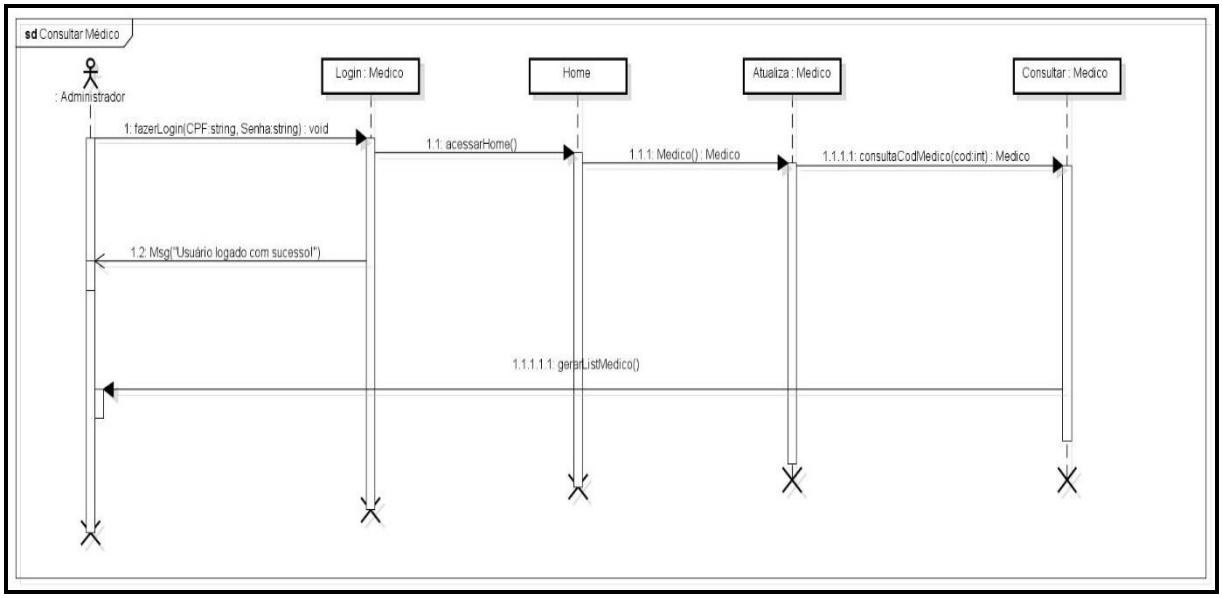


Figura 25 – Consultar Médico

4.5.4. Consultar Paciente

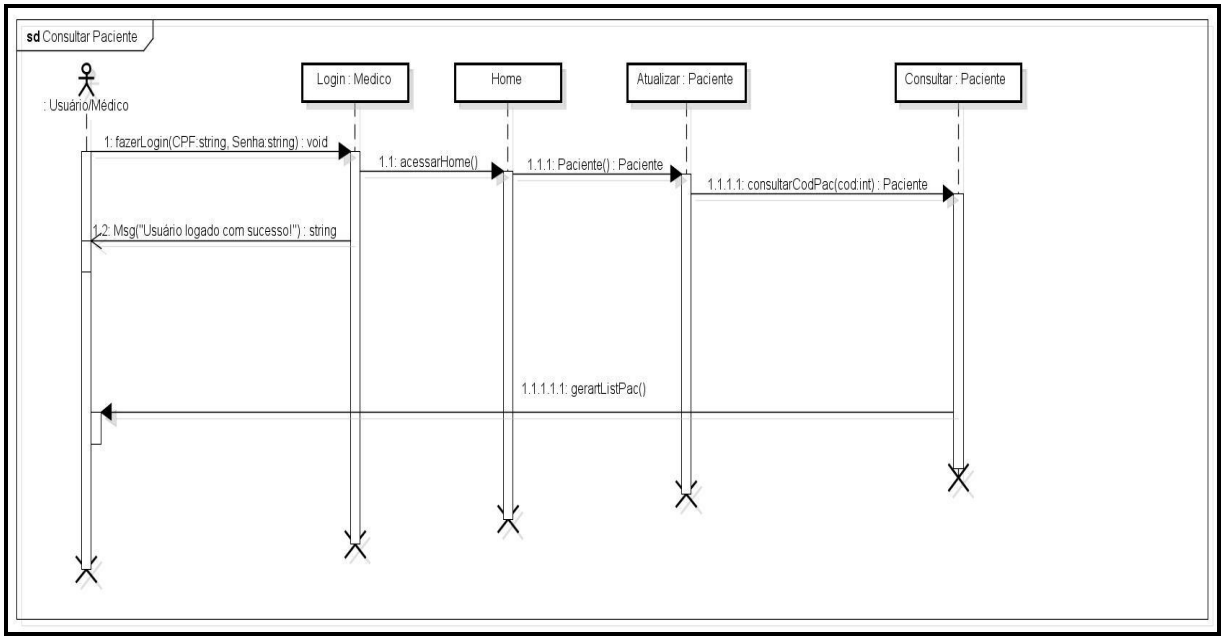


Figura 26 – Consultar Paciente

4.5.5. Entrar Homepage

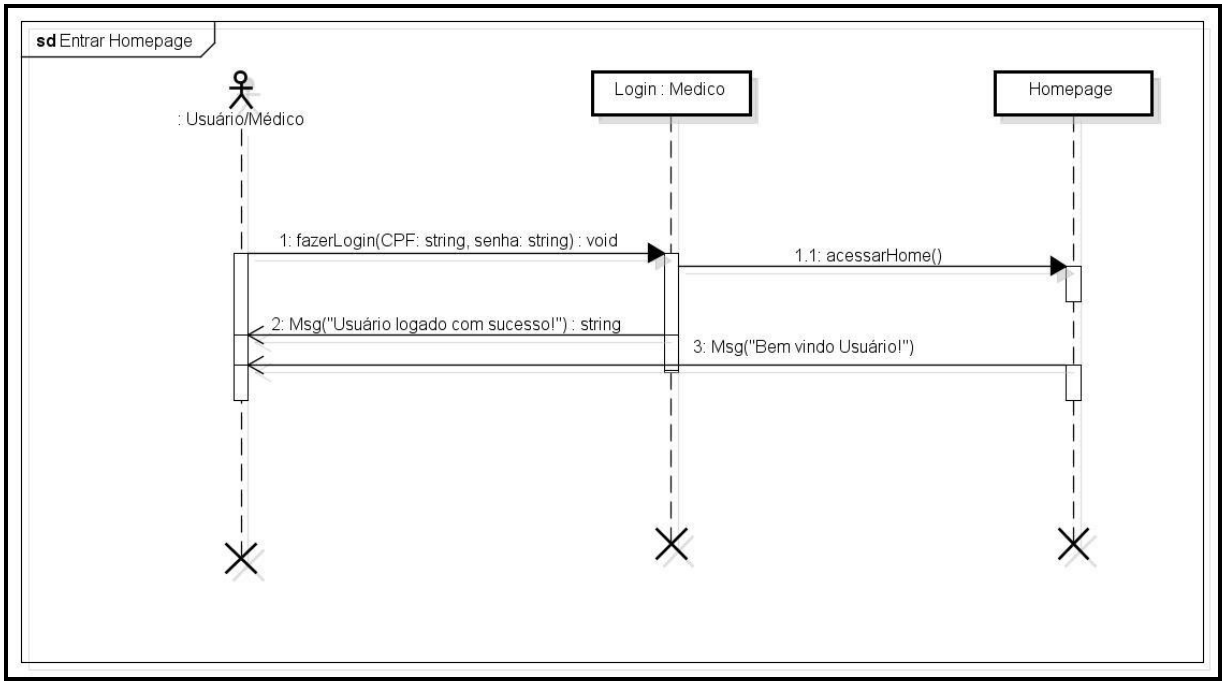


Figura 27 – Entrar Homepage

4.5.6. Fazer Login

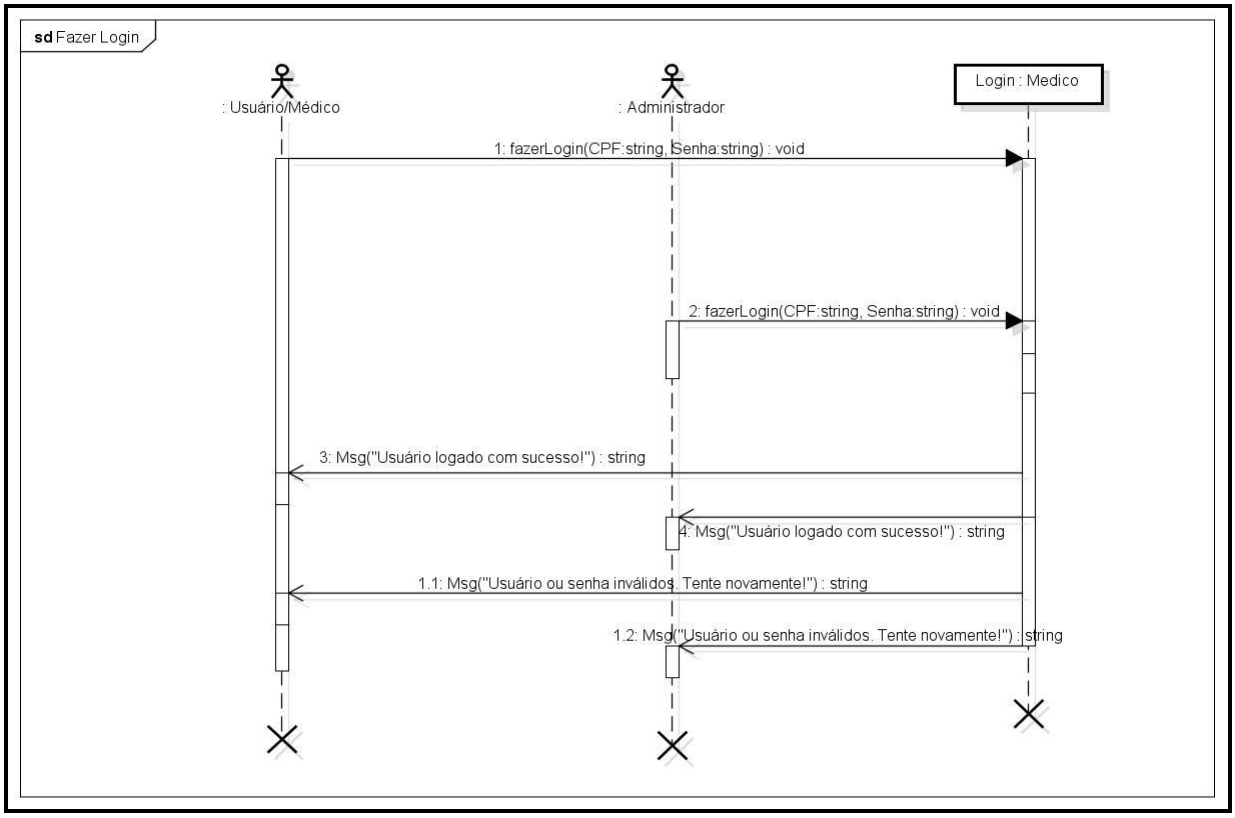


Figura 28 – Fazer Login

4.5.7. Atualizar Médico

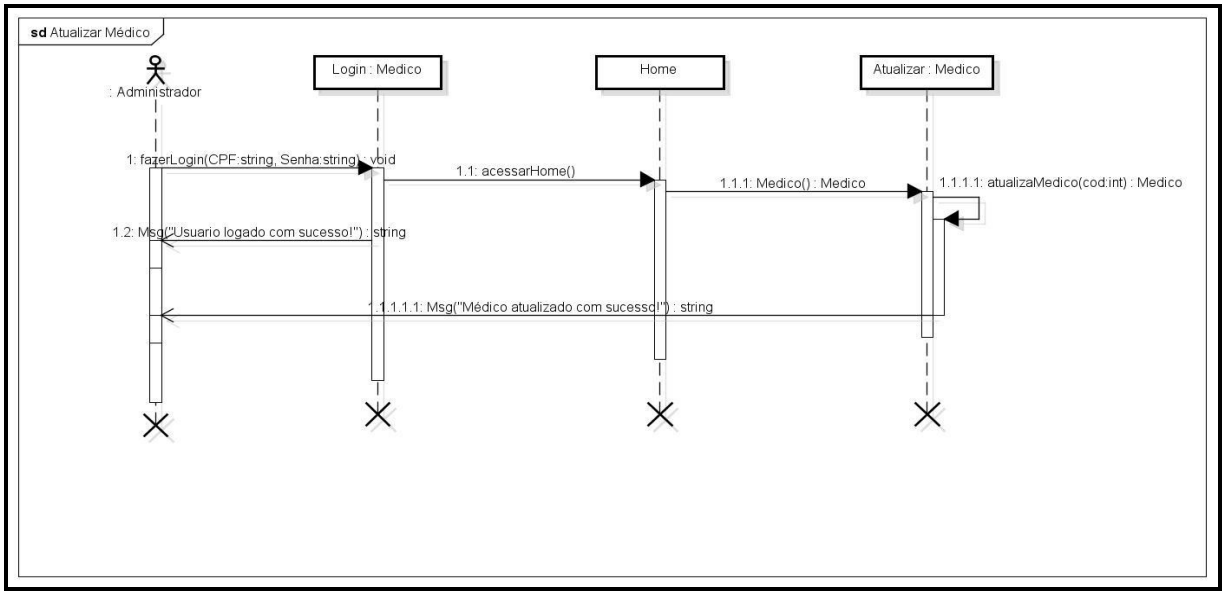


Figura 29 – Atualizar Médico

4.5.8. Atualizar Paciente

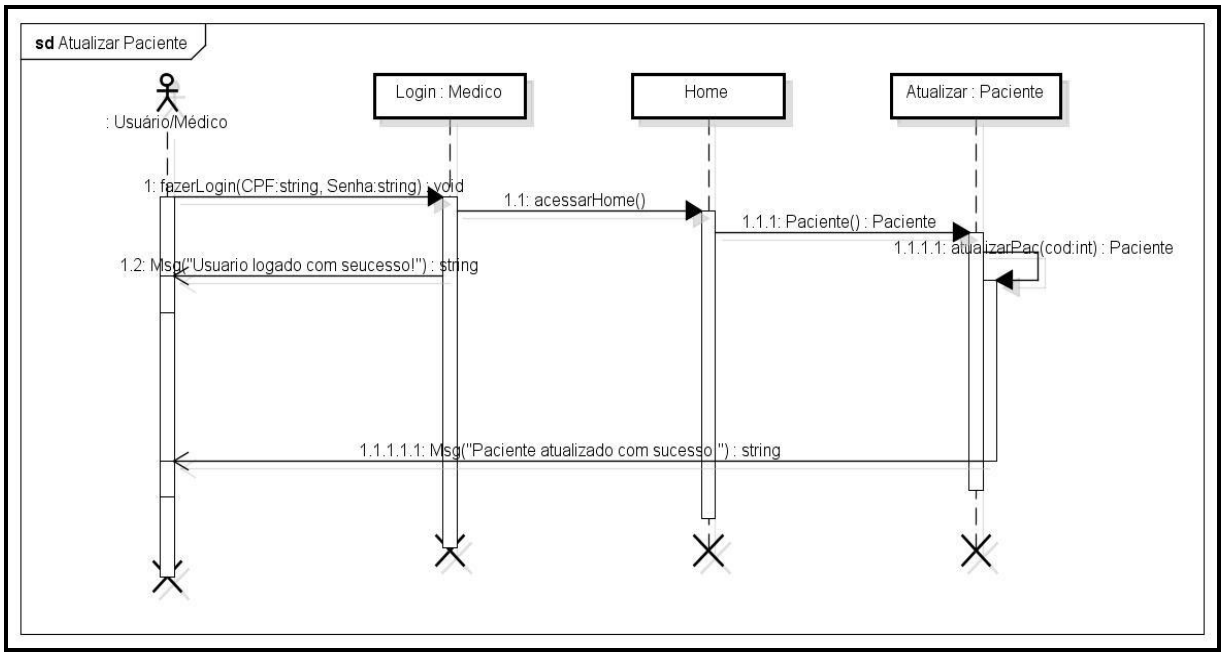


Figura 30 – Atualizar Paciente

4.5.9. Cadastrar CID

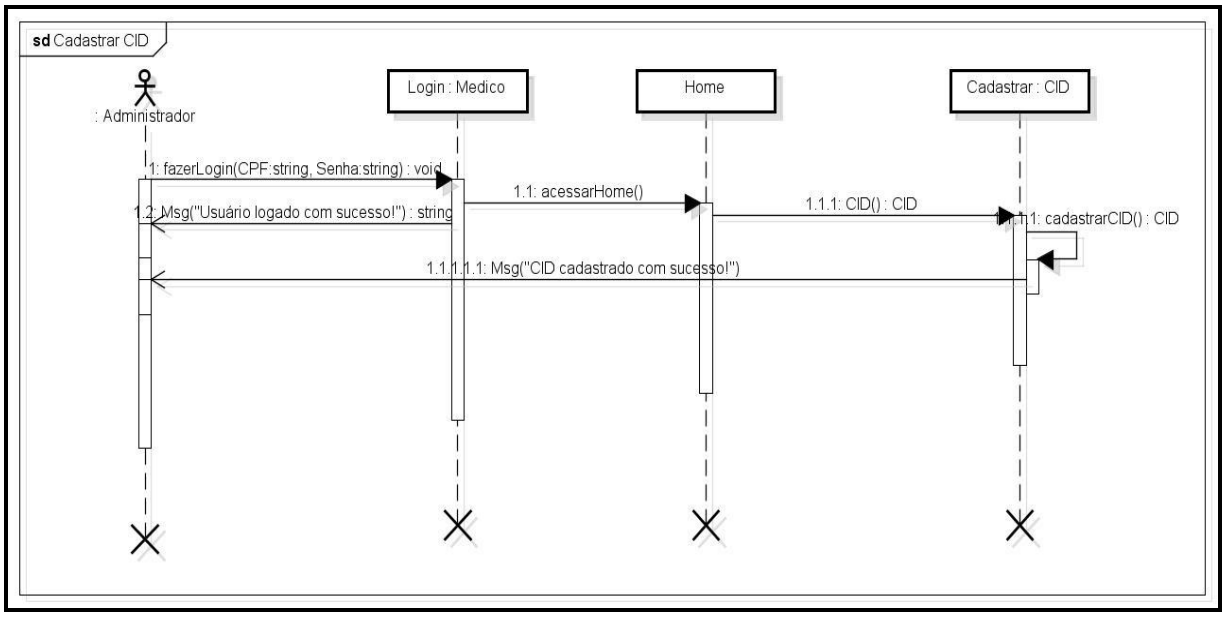


Figura 31 – Cadastrar CID

4.5.10. Cadastrar Clínica

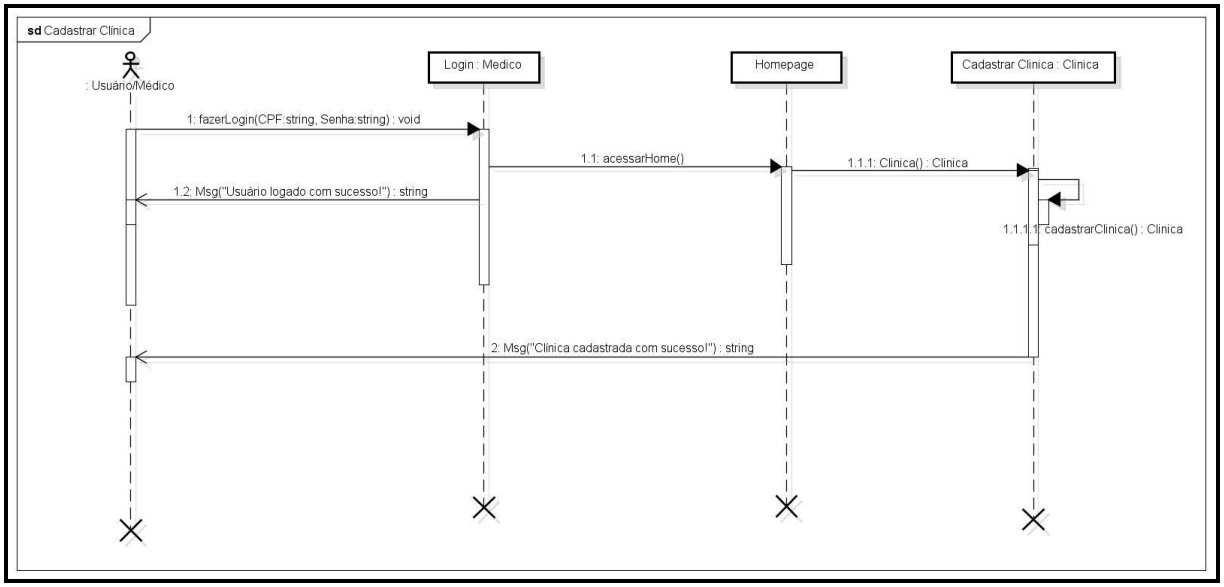


Figura 32 – Cadastrar Clínica

4.5.11. Cadastrar Histórico

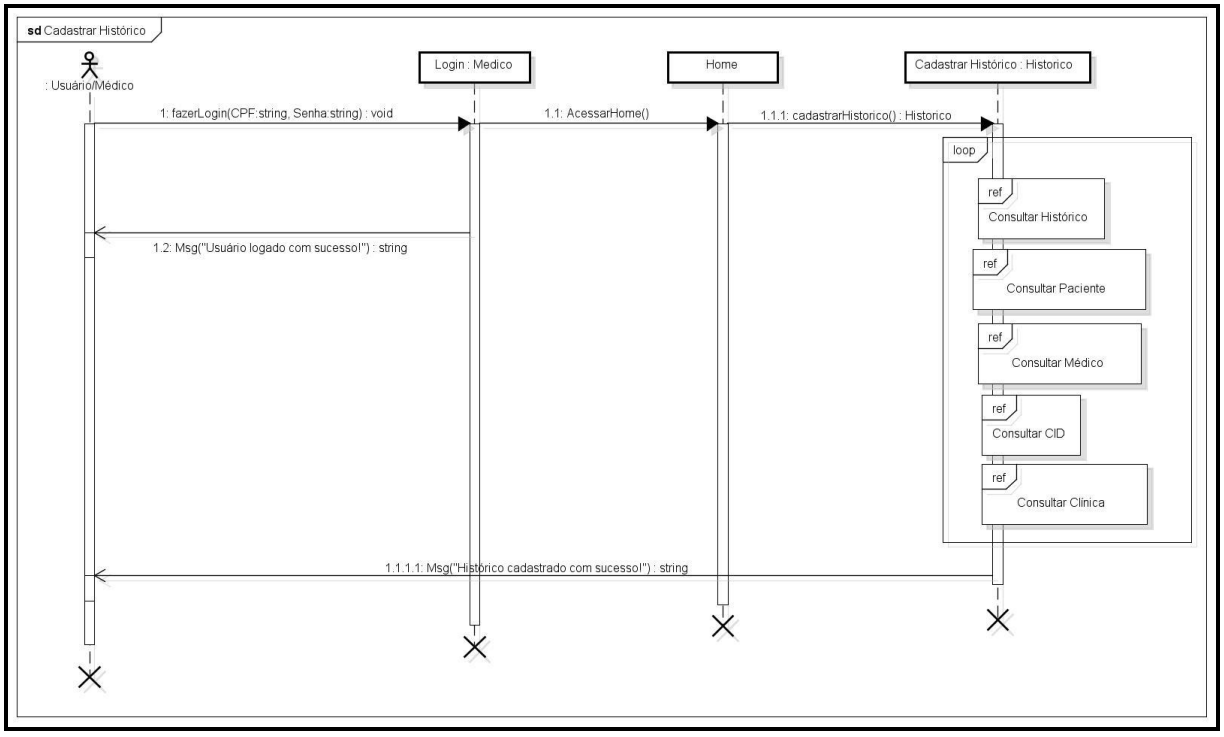


Figura 33 – Cadastrar Histórico

4.5.12. Cadastrar Paciente

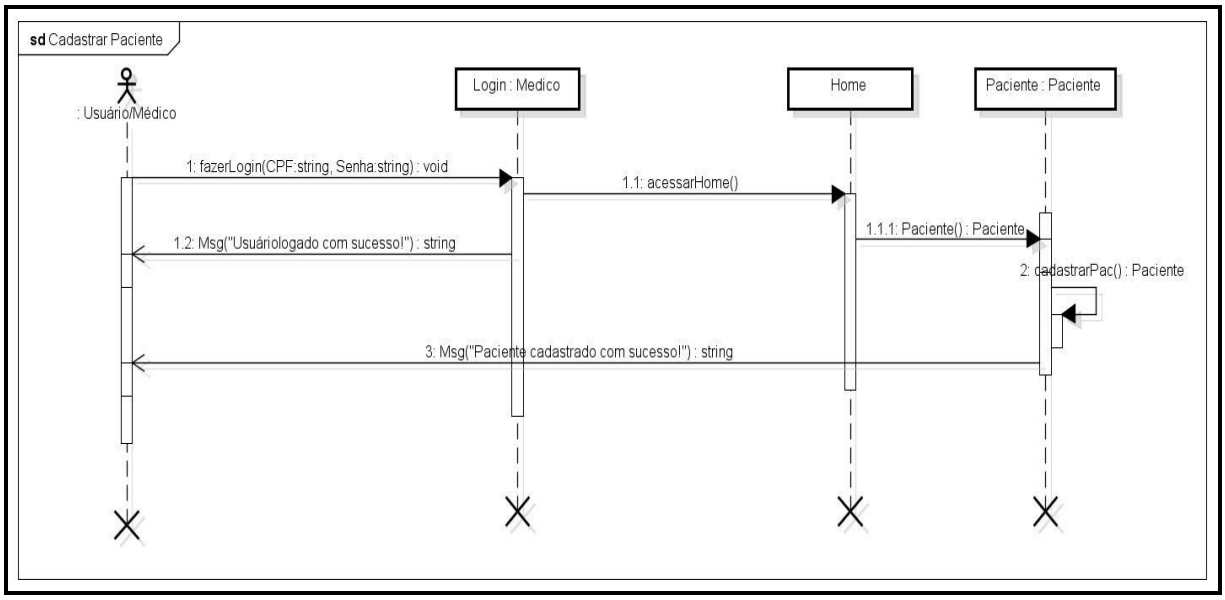


Figura 34 – Cadastrar Paciente

4.5.13. Cadastrar Usuário/Médico

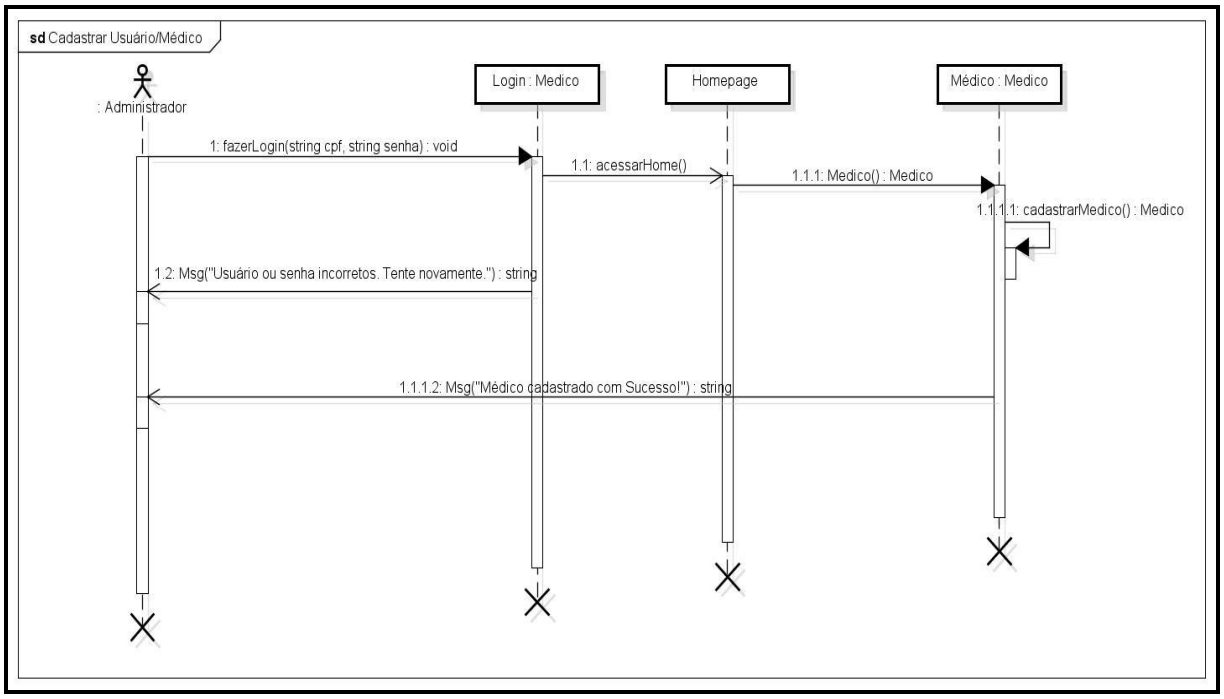


Figura 35 – Cadastrar Usuário/Médico

4.5.14. Consultar CID

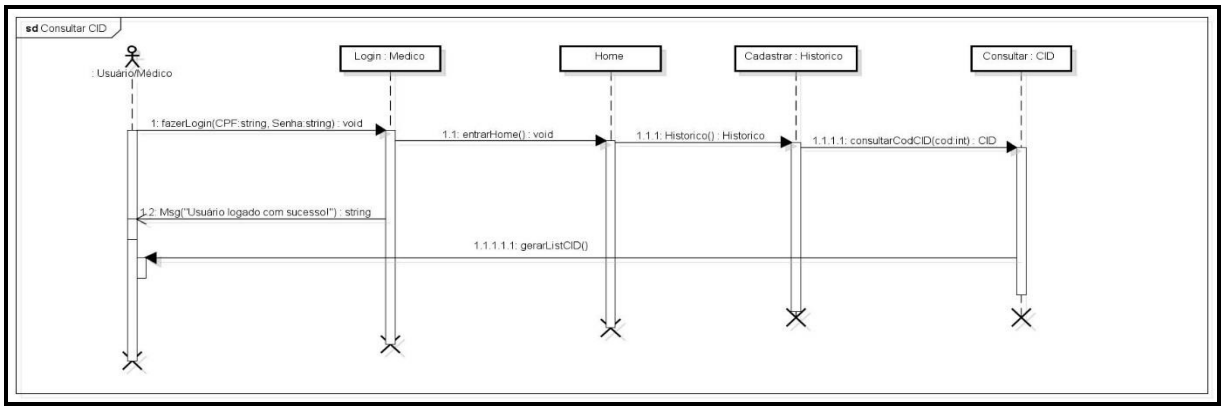


Figura 36 – Consultar CID

4.5.15. Consultar Clínica

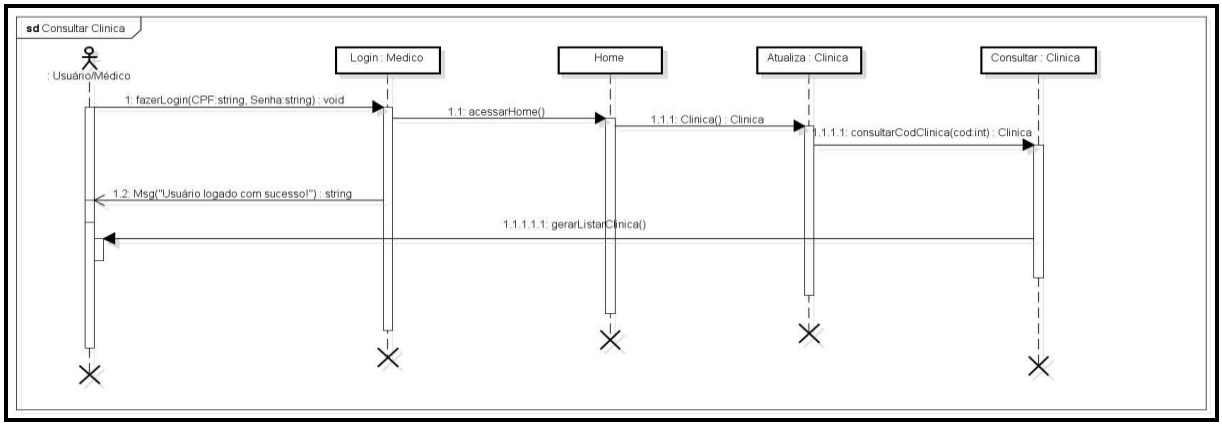


Figura 37 – Consultar Clínica

4.5.16. Consultar Histórico

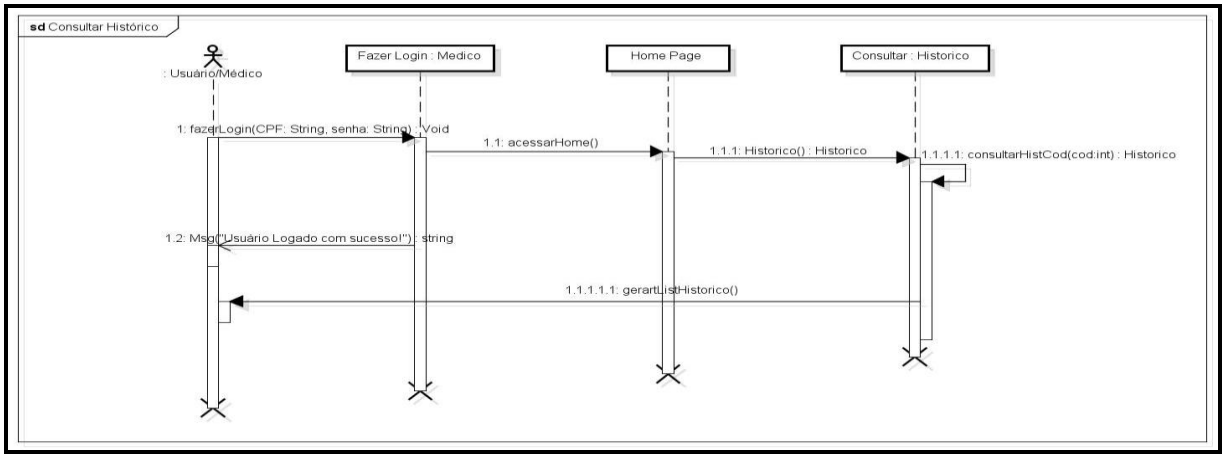


Figura 38 – Consultar Histórico

4.6. DIAGRAMA DE ATIVIDADE

Segundo Guedes (2010):

> Este diagrama é utilizado, como o próprio nome diz, para modelar atividades, que podem ser um método ou um algoritmo, ou mesmo um processo completo. [...] Neste contexto, elas são os métodos correspondentes às operações sobre classes.

4.6.1. Consultar Histórico

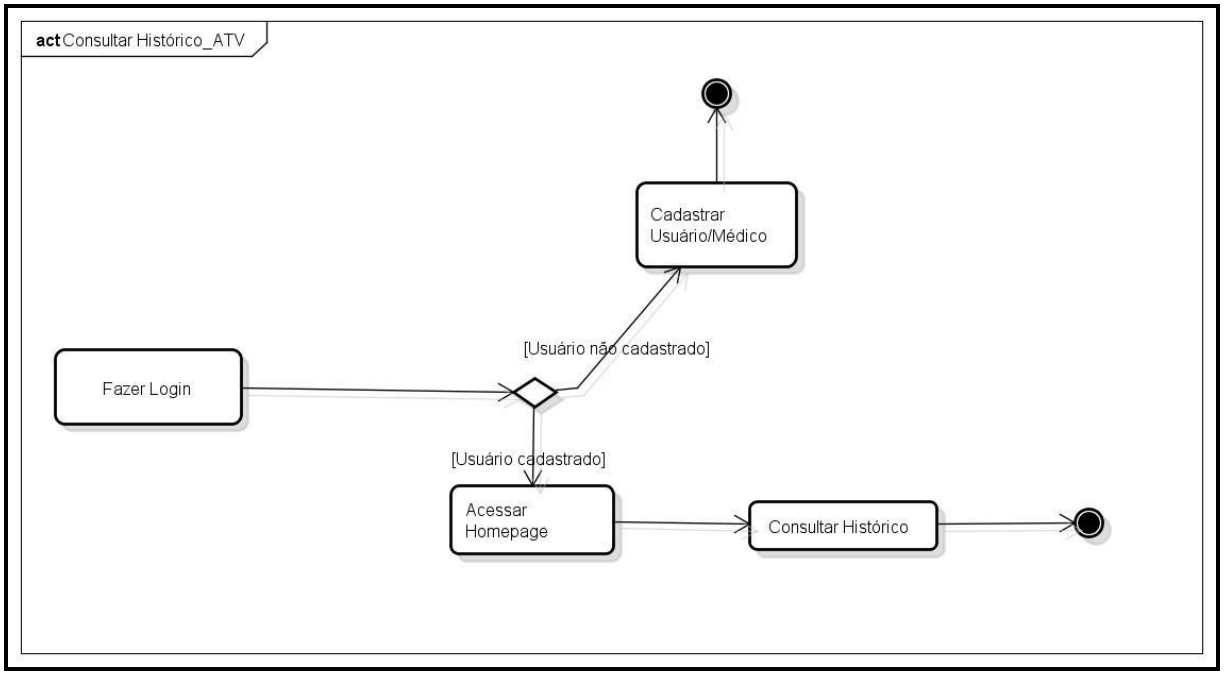


Figura 39 – Atividade Consultar Histórico

4.6.2. Cadastrar Histórico

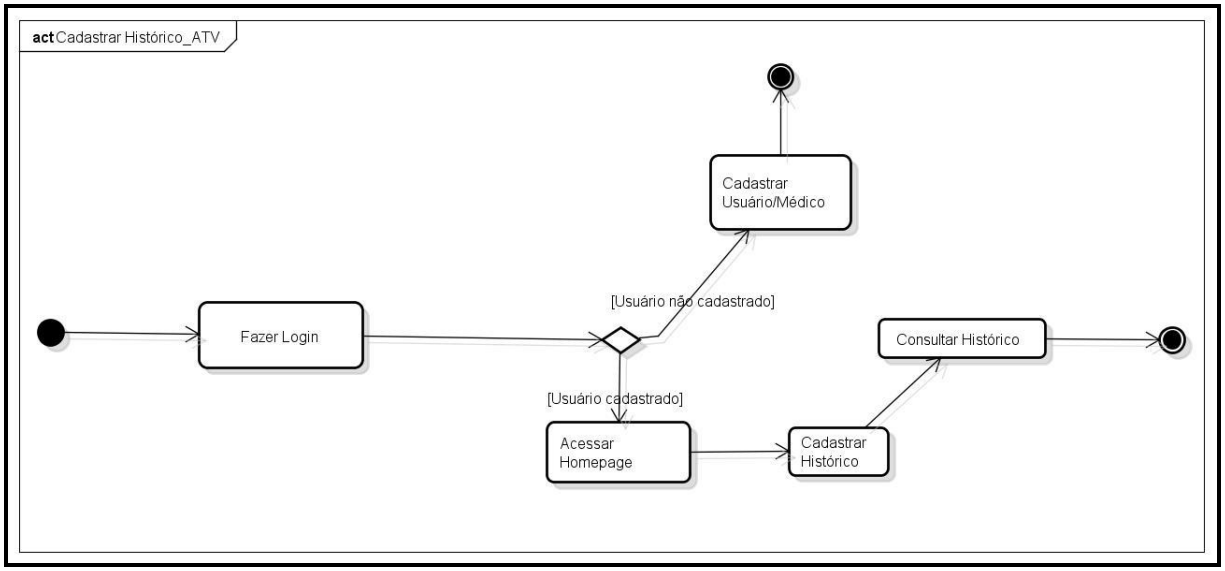


Figura 40 – Atividade Cadastrar Histórico

5. IMPLEMENTAÇÃO

A implementação do sistema será feita em três partes, utilizando o modelo de dados *Entity Data Model*, disponível no modelo *Entity Framework*, além de utilizar Programação em Camadas, onde são definidos separadamente o CRUD e as Regras de Negócio do Sistema.

5.1. IMPLEMENTAÇÃO EM BANCO DE DADOS

Com o auxílio do diagrama ER, será feita a construção do banco de dados relacional usando o *SQL Express do Visual Studio*.

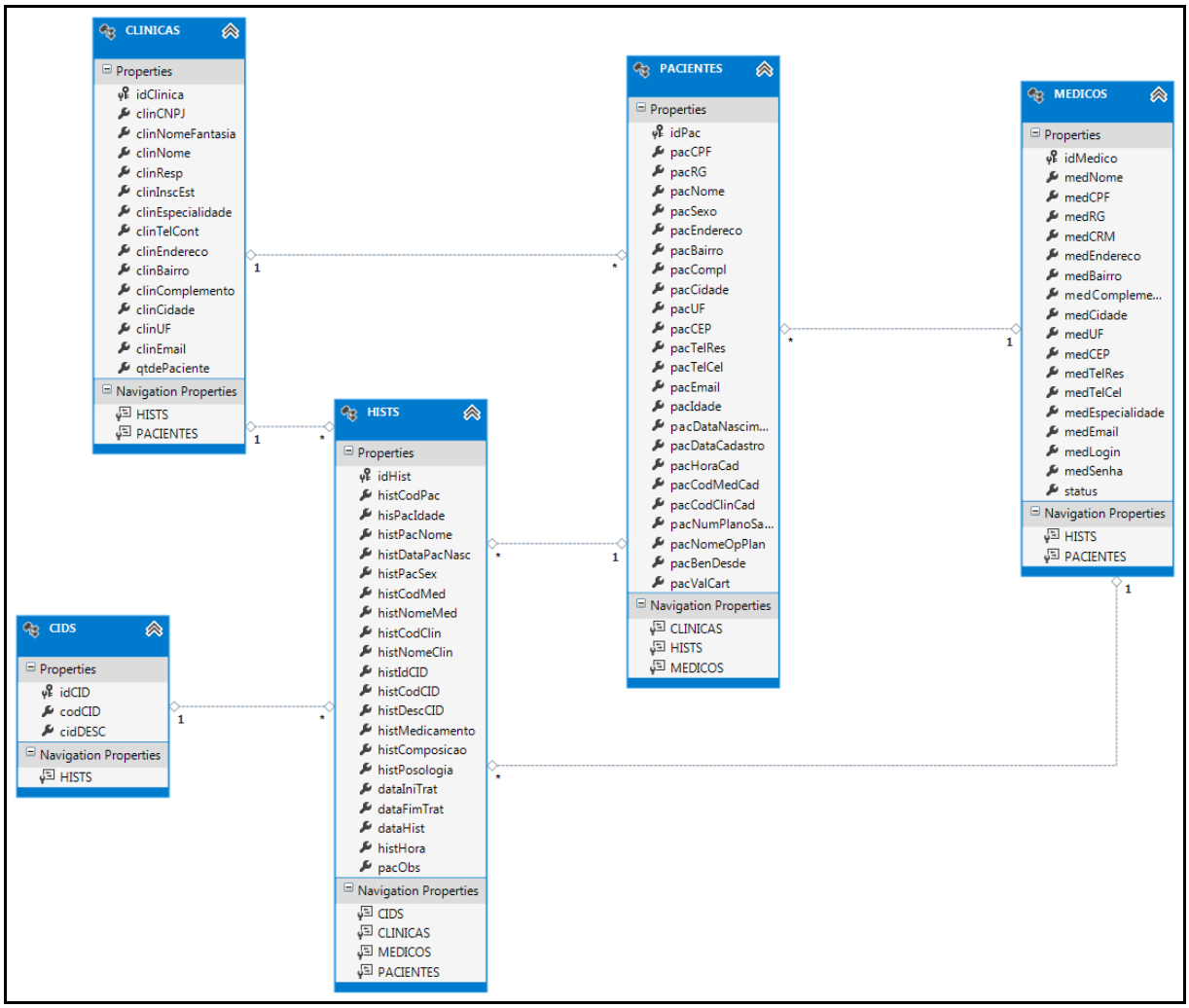


Figura 41 - Diagrama de E-R na ferramenta Entity

Após a construção do modelo, será executado a opção “Gerar Banco de dados a partir do modelo”. Esta opção criará um arquivo *script sql* dentro da pasta do projeto, que será executado, criando assim, o banco de dados com todas as suas entidades e relacionamentos. Dentro deste script, está contido os comandos *Create* para criar as tabelas no banco de dados.

Segue abaixo parte do conteúdo do *script sql*:

-- Creating table 'CIDS'

CREATE TABLE [dbo].[CIDS] ([idCID] int IDENTITY(1,1) NOT NULL, [codCID] nvarchar(200) NOT NULL, [cidDESC] nvarchar(2000) NOT NULL);

GO

-- Creating table 'CLINICAS'

CREATE TABLE [dbo].[CLINICAS] ([idClinica] int IDENTITY(1,1) NOT NULL, [clinCNPJ] varchar(18) NOT NULL, [clinNomeFantasia] varchar(100) NOT NULL, [clinNome] varchar(100) NOT NULL, [clinResp] varchar(100) NOT NULL, [clinInscEst] char(15) NOT NULL, [clinEspecialidade] varchar(100) NOT NULL, [clinTelCont] char(15) NOT NULL, [clinEndereco] varchar(100) NOT NULL, [clinBairro] varchar(50) NOT NULL, [clinComplemento] varchar(100) NOT NULL, [clinCidade] varchar(100) NOT NULL, [clinUF] char(2) NOT NULL, [clinEmail] varchar(50) NOT NULL, [qtdePaciente] int NOT NULL);

GO

-- Creating table 'HISTS'

CREATE TABLE [dbo].[HISTS] ([idHist] int IDENTITY(1,1) NOT NULL, [histCodPac] int NOT NULL, [hisPacIdade] int NOT NULL, [histPacNome] varchar(100) NOT NULL, [histDataPacNasc] datetime NOT NULL, [histPacSex] char(1) NOT NULL, [histCodMed] int NOT NULL, [histNomeMed] varchar(100) NOT NULL, [histCodClin] int NOT NULL,

[histNomeClin] varchar(100) NOT NULL, [histIdCID] int NOT NULL, [histCodCID] char(4) NOT NULL, [histDescCID] varchar(100) NOT NULL, [histMedicamento] varchar(100) NOT NULL, [histComposicao] varchar(100) NOT NULL, [histPosologia] varchar(100) NOT NULL, [dataIniTrat] datetime NOT NULL, [dataFimTrat] datetime NOT NULL, [dataHist] datetime NOT NULL, [histHora] datetime NOT NULL, [pacObs] nvarchar(max) NOT NULL);

GO

-- Creating table 'MEDICOS'

CREATE TABLE [dbo].[MEDICOS] ([idMedico] int IDENTITY(1,1) NOT NULL, [medNome] varchar(100) NULL, [medCPF] varchar(14) NULL, [medRG] varchar(12) NULL, [medCRM] nvarchar(10) NULL, [medEndereco] varchar(100) NULL, [medBairro] varchar(50) NULL, [medComplemento] varchar(50) NULL, [medCidade] varchar(50) NULL, [medUF] char(2) NULL, [medCEP] varchar(10) NULL, [medTelRes] varchar(15) NULL, [medTelCel] varchar(15) NULL, [medEspecialidade] varchar(100) NULL, [medEmail] varchar(50) NULL, [medLogin] char(11) NULL, [medSenha] char(10) NULL, [status] int NULL);

GO

-- Creating table 'PACIENTES'

CREATE TABLE [dbo].[PACIENTES] ([idPac] int IDENTITY(1,1) NOT NULL, [pacCPF] char(14) NOT NULL, [pacRG] char(12) NOT NULL, [pacNome] varchar(100) NOT NULL, [pacSexo] char(1) NOT NULL, [pacEndereco] varchar(100) NOT NULL, [pacBairro] varchar(50) NOT NULL, [pacCompl] varchar(50) NOT NULL, [pacCidade] varchar(50) NOT NULL, [pacUF] char(2) NOT NULL, [pacCEP] char(10) NOT NULL, [pacTelRes] char(15) NOT NULL, [pacTelCel] char(15) NOT NULL, [pacEmail] varchar(50) NOT NULL, [pacIdade] int NOT NULL, [pacDataNascimento] datetime NOT NULL, [pacDataCadastro] datetime NOT NULL, [pacHoraCad] datetime NOT NULL, [pacCodMedCad] int NOT NULL, [pacCodClinCad] int NOT NULL, [pacNumPlanoSaude] int

```
NOT NULL, [pacNomeOpPlan] varchar(100) NOT NULL,
[pacBenDesde] datetime  NOT NULL, [pacValCart] datetime  NOT NULL);
GO
-- ----------------------------------------------------
-- Creating all PRIMARY KEY constraints
-- ----------------------------------------------------
-- Creating primary key on [idCID] in table 'CIDS'
ALTER TABLE [dbo].[CIDS]
ADD CONSTRAINT [PK_CIDS]
   PRIMARY KEY CLUSTERED ([idCID] ASC);
GO
-- Creating primary key on [idClinica] in table 'CLINICAS'
ALTER TABLE [dbo].[CLINICAS]
ADD CONSTRAINT [PK_CLINICAS]
   PRIMARY KEY CLUSTERED ([idClinica] ASC);
GO
-- Creating primary key on [idHist] in table 'HISTS'
ALTER TABLE [dbo].[HISTS]
ADD CONSTRAINT [PK_HISTS]
   PRIMARY KEY CLUSTERED ([idHist] ASC);
GO
-- Creating primary key on [idMedico] in table 'MEDICOS'
ALTER TABLE [dbo].[MEDICOS]
ADD CONSTRAINT [PK_MEDICOS]
   PRIMARY KEY CLUSTERED ([idMedico] ASC);
```

```
GO
-- Creating primary key on [idPac] in table 'PACIENTES'
ALTER TABLE [dbo].[PACIENTES]
ADD CONSTRAINT [PK_PACIENTES]
    PRIMARY KEY CLUSTERED ([idPac] ASC);
GO
-- --------------------------------------------------
-- Creating all FOREIGN KEY constraints
-- --------------------------------------------------
-- Creating foreign key on [histIdCID] in table 'HISTS'
ALTER TABLE [dbo].[HISTS]
ADD CONSTRAINT [FK_HISTS_CIDS]
    FOREIGN KEY ([histIdCID])
    REFERENCES [dbo].[CIDS] ([idCID])
    ON DELETE CASCADE ON UPDATE NO ACTION;
-- Creating non-clustered index for FOREIGN KEY 'FK_HISTS_CIDS'
CREATE INDEX [IX_FK_HISTS_CIDS]
ON [dbo].[HISTS] ([histIdCID]);
GO
-- Creating foreign key on [histCodClin] in table 'HISTS'
ALTER TABLE [dbo].[HISTS]
ADD CONSTRAINT [FK_CLINICAHIST]
    FOREIGN KEY ([histCodClin])
    REFERENCES [dbo].[CLINICAS] ([idClinica])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
```

```
-- Creating non-clustered index for FOREIGN KEY 'FK_CLINICAHIST'

CREATE INDEX [IX_FK_CLINICAHIST]

ON [dbo].[HISTS] ([histCodClin]);

GO

-- Creating foreign key on [pacCodClinCad] in table 'PACIENTES'

ALTER TABLE [dbo].[PACIENTES]

ADD CONSTRAINT [FK_CLINICAPACIENTE]

   FOREIGN KEY ([pacCodClinCad]) REFERENCES [dbo].[CLINICAS] ([idClinica])

   ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Creating non-clustered index for FOREIGN KEY 'FK_CLINICAPACIENTE'

CREATE INDEX [IX_FK_CLINICAPACIENTE]

ON [dbo].[PACIENTES] ([pacCodClinCad]);

GO

-- Creating foreign key on [histCodMed] in table 'HISTS'

ALTER TABLE [dbo].[HISTS]

ADD CONSTRAINT [FK_HISTS_MEDICOS]

   FOREIGN KEY ([histCodMed]) REFERENCES [dbo].[MEDICOS] ([idMedico])

   ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Creating non-clustered index for FOREIGN KEY 'FK_HISTS_MEDICOS'

CREATE INDEX [IX_FK_HISTS_MEDICOS]

ON [dbo].[HISTS] ([histCodMed]);

GO

-- Creating foreign key on [histCodPac] in table 'HISTS'
```

```
ALTER TABLE [dbo].[HISTS]
ADD CONSTRAINT [FK_PACIENTEHIST]
    FOREIGN KEY ([histCodPac]) REFERENCES [dbo].[PACIENTES] ([idPac])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
-- Creating non-clustered index for FOREIGN KEY 'FK_PACIENTEHIST'
CREATE INDEX [IX_FK_PACIENTEHIST]
ON [dbo].[HISTS] ([histCodPac]);
GO
-- Creating foreign key on [pacCodMedCad] in table 'PACIENTES'
ALTER TABLE [dbo].[PACIENTES]
ADD CONSTRAINT [FK_PACIENTES_MEDICOS]
    FOREIGN KEY ([pacCodMedCad]) REFERENCES [dbo].[MEDICOS] ([idMedico])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
-- Creating non-clustered index for FOREIGN KEY 'FK_PACIENTES_MEDICOS'
CREATE INDEX [IX_FK_PACIENTES_MEDICOS]
ON [dbo].[PACIENTES] ([pacCodMedCad]);
GO
-- --------------------------------------------------
-- Script has ended
-- --------------------------------------------------
```

5.2. DEFINIÇÃO DA CAMADA DAL (*DATA ACCESS LAYER*)

O CRUD, ou seja, as classes que terão os métodos de inserção, alteração, busca e exclusão dos dados, será estruturado em uma camada de nome DAL.

Segundo CAMACHO JUNIOR (2008, p. 31):

> *Nessa camada, [...] implementar os métodos de inserção, atualização, exclusão e listagem referentes a todas a s tabelas existentes [...].*

Cada entidade terá seu CRUD, como se segue nas linhas de código apresentadas a seguir:

5.2.1. Classe dalCID

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

public class dalCID
{
   historicoEntities contexto = new historicoEntities();

        public dalCID()
        {
        }

   public List<CIDS> Select()
   {
      return contexto.CIDS.ToList();
   }

   public CIDS SelectCod(int cod)
   {
      CIDS oCid = contexto.CIDS.First(cid => cid.idCID == cod);
      return oCid;
   }

   public void Insert(CIDS oCid)
   {
```

```
        contexto.CIDS.Add(oCid);
        contexto.SaveChanges();
    }

    public void Update(CIDS oCid)
    {
        CIDS cidUp = contexto.CIDS.First(cid => cid.idCID == oCid.idCID);
        cidUp.codCID = oCid.codCID;
        cidUp.cidDESC = oCid.cidDESC;
        contexto.SaveChanges();
    }
}
```

5.2.2. Classe dalClinicas

```
public class dalClinicas
{
    historicoEntities contexto = new historicoEntities();
        public dalClinicas()
        {
        }
    //retorna uma lista de clinicas cadastradas
    public List<CLINICAS> Select()
    {
        return contexto.CLINICAS.ToList();
    }
    //select retorna objeto com o código da clínica igual ao código passado
como parâmetro
    public CLINICAS SelectCod(int cod)
    {
       CLINICAS oCli = contexto.CLINICAS.First(cli => cli.idClinica == cod);
        return oCli;
    }
    //insere um novo objeto do tipo Clinica
    public void Insert(CLINICAS oCli)
    {
        contexto.CLINICAS.Add(oCli);
```

```
        contexto.SaveChanges();
    }
    //atualiza a clínica de acordo com o cod selecionado no GridView
    public void Update(CLINICAS oCli)
    {
        CLINICAS cliUp = contexto.CLINICAS.First(cli => cli.idClinica ==
oCli.idClinica);
        cliUp.clinNomeFantasia = oCli.clinNomeFantasia;
        cliUp.clinNome = oCli.clinNome;
        cliUp.clinInscEst = oCli.clinInscEst;
        cliUp.clinEspecialidade = oCli.clinEspecialidade;
        cliUp.clinEndereco = oCli.clinEndereco;
        cliUp.clinEmail = oCli.clinEmail;
        cliUp.clinComplemento = oCli.clinComplemento;
        cliUp.clinCNPJ = oCli.clinCNPJ;
        cliUp.clinCidade = oCli.clinCidade;
        cliUp.clinBairro = oCli.clinBairro;
        cliUp.clinResp = oCli.clinResp;
        cliUp.clinTelCont = oCli.clinTelCont;
        cliUp.clinUF = oCli.clinUF;
        cliUp.qtdePaciente = oCli.qtdePaciente;

        contexto.SaveChanges();
    }
    //deleta o objeto de acordo com seu código
    public void Delete(CLINICAS oCli)    {
            CLINICAS        delCli=contexto.CLINICAS.First(cli=>cli.idClinica==
Cli.idClinica);
            contexto.CLINICAS.Remove(delCli);
            contexto.SaveChanges();
        }
}
```

5.2.3. Classe dalHistoricos

```
public class dalHistoricos
{
    historicoEntities contexto = new historicoEntities();
    MEDICOS oMed = new MEDICOS();

        public dalHistoricos()
        {
        }
    public List<HISTS> Select()
    {
        return contexto.HISTS.ToList();
    }
    public List<HISTS> SelectCod(int cod)
    {
        return contexto.HISTS.Where(hist => hist.idHist == cod).ToList();
    }
    public void Insert(HISTS oHist)
    {
        contexto.HISTS.Add(oHist);
        contexto.SaveChanges();
    }
    public List<HISTS> SelectMed(int cod)
    {
        return contexto.HISTS.Where(hist => hist.histCodMed ==
cod).ToList();
    }
    public List<HISTS> SelectCli(int cod)
    {
        return contexto.HISTS.Where(hist => hist.histCodClin ==
cod).ToList();
```

```
    }
    public List<HISTS> SelectPac(int cod)
    {
        return contexto.HISTS.Where(hist => hist.histCodPac == cod).ToList();
    }
}
```

5.2.4. Classe dalMedicos

```
public class dalMedicos
{
    historicoEntities contexto = new historicoEntities();
        public dalMedicos()
        {
        }
    public MEDICOS ValidaMed(string login, string senha)
    {
        int testa = contexto.MEDICOS.Count(med => (med.medLogin == login) && (med.medSenha == senha));
        if (testa > 0)
        {
            MEDICOS oMed = contexto.MEDICOS.First(med => (med.medLogin == login) && (med.medSenha == senha));
            return oMed;
        }
        else return null;
    }
    public List<MEDICOS> Select()
    {
        return contexto.MEDICOS.ToList();
    }
```

```
    public void Insert(MEDICOS oMed)
    {
        contexto.MEDICOS.Add(oMed);
        contexto.SaveChanges();
    }
    public void Update(MEDICOS oMed)
    {
        MEDICOS medUp = contexto.MEDICOS.First(med => med.idMedico
== oMed.idMedico);

        medUp.medBairro = oMed.medBairro;
        medUp.medCEP = oMed.medCEP;
        medUp.medCidade = oMed.medCidade;
        medUp.medComplemento = oMed.medComplemento;
        medUp.medCPF = oMed.medCPF;
        medUp.medCRM = oMed.medCRM;
        medUp.medEmail = oMed.medEmail;
        medUp.medEndereco = oMed.medEndereco;
        medUp.medEspecialidade = oMed.medEspecialidade;
        medUp.medLogin = oMed.medLogin;
        medUp.medNome = oMed.medNome;
        medUp.medRG = oMed.medRG;
        medUp.medSenha = oMed.medSenha;
        medUp.medTelCel = oMed.medTelCel;
        medUp.medTelRes = oMed.medTelRes;
        medUp.medUF = oMed.medUF;
        medUp.status = oMed.status;

        contexto.SaveChanges();
    }
    public void Delete(MEDICOS oMed)
    {
```

```
        oMed = contexto.MEDICOS.First(med => med.idMedico ==
oMed.idMedico);
        contexto.MEDICOS.Remove(oMed);
        contexto.SaveChanges();
    }
    public MEDICOS SelectCod(int cod)
    {
        MEDICOS oMed = contexto.MEDICOS.First(med => med.idMedico
== cod);
        return oMed;
    }
    public MEDICOS SelectStatus(string usuario, string senha)
    {
        int testa = contexto.MEDICOS.Count(med => (med.medLogin ==
usuario) && (med.medSenha == senha));
        if (testa > 0)
        {
            MEDICOS oMed = contexto.MEDICOS.First(med =>
(med.medLogin == usuario) && (med.medSenha == senha));
            return oMed;
        }
        else
        {
            return null;
        }
    }
}
```

5.2.5. Classe dalPacientes

```
public class dalPacientes
{
```

```
historicoEntities contexto = new historicoEntities();
        public dalPacientes()
        {
        }
public List<PACIENTES> Select()
{
    return contexto.PACIENTES.ToList();
}
public PACIENTES SelectCod(int cod)
{
    PACIENTES oPac = contexto.PACIENTES.First(pac => pac.idPac ==
cod);
    return oPac;
}
public void Insert(PACIENTES oPac)
{
    contexto.PACIENTES.Add(oPac);
    contexto.SaveChanges();
}
public void Uptade(PACIENTES oPac)
{
    PACIENTES pacUp = contexto.PACIENTES.First(pac => pac.idPac
== oPac.idPac);
    pacUp.pacBairro = oPac.pacBairro;
    pacUp.pacBenDesde = oPac.pacBenDesde;
    pacUp.pacCEP = oPac.pacCEP;
    pacUp.pacCidade = oPac.pacCidade;
    pacUp.pacCodClinCad = oPac.pacCodClinCad;
    pacUp.pacCodMedCad = oPac.pacCodMedCad;
    pacUp.pacCompl = oPac.pacCompl;
    pacUp.pacCPF = oPac.pacCPF;
    pacUp.pacDataCadastro = oPac.pacDataCadastro;
```

```
        pacUp.pacDataNascimento = oPac.pacDataNascimento;
        pacUp.pacEmail = oPac.pacEmail;
        pacUp.pacEndereco = oPac.pacEndereco;
        pacUp.pacHoraCad = oPac.pacHoraCad;
        pacUp.pacIdade = oPac.pacIdade;
        pacUp.pacNome = oPac.pacNome;
        pacUp.pacNomeOpPlan = oPac.pacNomeOpPlan;
        pacUp.pacNumPlanoSaude = oPac.pacNumPlanoSaude;
        pacUp.pacRG = oPac.pacRG;
        pacUp.pacSexo = oPac.pacSexo;
        pacUp.pacTelCel = oPac.pacTelCel;
        pacUp.pacTelRes = oPac.pacTelRes;
        pacUp.pacUF = oPac.pacUF;
        pacUp.pacValCart = oPac.pacValCart;

        contexto.SaveChanges();
        }
    public void Delete(PACIENTES oPac)
    {
        oPac = contexto.PACIENTES.First(pac => pac.idPac == oPac.idPac);
        contexto.PACIENTES.Remove(oPac);
        contexto.SaveChanges();
    }
}
```

5.3. DEFINIÇÃO DA CAMADA BLL (*BUSINESS LOGIC LAYER*)

É na BLL onde ficam todas as regras de negócio do sistema. Regras essas que serão simples, controlando inserções errôneas ou em branco, controlando alterações de dados e também um controle quando se apaga algum cadastro ou alguma movimentação.

Segundo CAMACHO JÚNIOR (2008, p.77):

> *As regras de negócio definem como o seu negócio funciona. Essas regras podem abranger diversos assuntos como suas políticas, interesses, objetivos, compromissos éticos e sociais, obrigações contratuais, decisões estratégicas, leis e regulamentações, entre outros.*

Assim como na camada DAL, cada entidade também terá suas classes de regra de negócio:

5.3.1. Classe bllCID

```
public class bllCID
{
   dalCID dalCID = new dalCID();
        public bllCID()
        {
        }
   public List<CIDS> Select()
   {
      return dalCID.Select();
   }
   public CIDS SelectCod(int cod)
   {
      return dalCID.SelectCod(cod);
   }
   public void Update(CIDS oCid)
   {
      dalCID.Update(oCid);
   }
   public void Insert(CIDS oCid)
```

```
    {
        dalCID.Insert(oCid);
    }
}
```

5.3.2. Classe bllClinicas

```
public class bllClinicas
{
    dalClinicas dalCli = new dalClinicas();

        public bllClinicas()
        {
        }
    public List<CLINICAS> Select()
    {
        return dalCli.Select();
    }
    public CLINICAS Select(int cod)
    {
        return dalCli.SelectCod(cod);
    }
    public void Insert(CLINICAS oCli)
    {
            oCli.qtdePaciente = 0;
            dalCli.Insert(oCli);
    }
    public void Update(CLINICAS oCli)
    {
            dalCli.Update(oCli);
    }
    public void Delete(CLINICAS oCli)
```

```
    {
        try
        {
            dalCli.Delete(oCli);
        }
        catch { }
    }
}
```

5.3.3. Classe bllHistorico

```
public class bllHistoricos
{
        public bllHistoricos()
        {
        }
    dalHistoricos dalHist = new dalHistoricos();
    public List<HISTS> Select()
    {
        return dalHist.Select();
    }
    public List<HISTS> SelectCod(int cod)
    {
        return dalHist.SelectCod(cod);
    }
    public List<HISTS> SelectMed(int cod)
    {
        return dalHist.SelectMed(cod);
    }
    public List<HISTS> SelectPac(int cod)
    {
        return dalHist.SelectPac(cod);
```

```
    }
    public List<HISTS> SelectClin(int cod)
    {
        return dalHist.SelectCli(cod);
    }
    public void Insert(HISTS oHist)
    {
        dalHist.Insert(oHist);
    }
}
```

5.3.4. Classe bllMedico

```
public class bllMedicos
{
    dalMedicos dalMed = new dalMedicos();

    public List<MEDICOS> Select()
    {
        return dalMed.Select();
    }
    public void Insert(MEDICOS oMed)
    {
        oMed.status = 0;
        dalMed.Insert(oMed);
    }
    public void Update(MEDICOS oMed)
    {
        dalMed.Update(oMed);
    }
    public void Delete(MEDICOS oMed)
    {
```

```
        try
        {
            dalMed.Delete(oMed);
        }
        catch { }
    }
    public MEDICOS SelectCod(int cod)
    {
        return dalMed.SelectCod(cod);
    }
}
```

5.3.5. Classe bllPaciente

```
public class bllPacientes
{
    dalPacientes dalPac = new dalPacientes();
          public bllPacientes()
          {
          }
    public List<PACIENTES> Select()
    {
        return dalPac.Select();
    }
    public PACIENTES SelectCod(int cod)
    {
        return dalPac.SelectCod(cod);
    }
    public void Insert(PACIENTES oPac)
    {
        CLINICAS oCli = new CLINICAS();
        bllClinicas bllCli = new bllClinicas();
```

```
            dalPac.Insert(oPac);
            oCli = bllCli.Select(oPac.pacCodClinCad);
            oCli.qtdePaciente++;
            bllCli.Update(oCli);
        }
        public void Update(PACIENTES oPac)
        {
            dalPac.Uptade(oPac);
        }
        public void Delete(PACIENTES oPac)
        {
            CLINICAS oCli = new CLINICAS();
            bllClinicas bllCli = new bllClinicas();
            try
            {
                dalPac.Delete(oPac);
                oCli = bllCli.Select(oPac.pacCodClinCad);
                oCli.qtdePaciente--;
                bllCli.Update(oCli);
            }
            catch { }
        }
    }
```

6. CONCLUSÃO

O objetivo desse trabalho foi de criar e implementar um *software* para informatização do sistema privado de saúde, idealizado para a auxiliar a tomada de decisão de médicos em clínicas particulares, geralmente de médio a pequeno porte, de maneira segura, rápida e eficaz.

O *software* foi idealizado de acordo com os estudos realizados no curso Tecnologia em Análise e Desenvolvimento de Sistemas e foi construído com o *Visual Studio 2012*, o qual oferece uma ampla gama de opções de criação, utilizando-se também, do banco de dados da própria ferramenta, o *Microsoft SQL Server 2012*. Foi utilizada a plataforma *Web ASP.Net* juntamente com a linguagem de programação C# para a criação das páginas e formulários, não exigindo altos custos de implantação e de manutenção, tanto para o usuário quanto para o programador.

Inicialmente, acreditava-se que o sistema serviria de auxiliador na tomada de decisões de possíveis tratamentos, pois, os dados pessoais do paciente, bem como os tratamentos anteriores serão guardados em um banco de dados, e poderão ser cadastrados pelo próprio usuário (médico ou outra pessoa de escolha do médico, como por exemplo, secretárias e recepcionistas), agilizando assim todo o processo desde a consulta até o diagnóstico final. Dessa forma, concluímos que ao término do trabalho e após a implantação deste módulo, viabilizará mais controle, segurança e eficiência para aplicação de tratamentos e maior agilidade na tomada de decisão por parte dos usuários, sendo o mesmo otimizado para clínicas médicas particulares.

As tecnologias usadas no sistema são emergentes, relativamente novas, dessa forma, o trabalho contribui para a fomentação de pesquisas e desenvolvimento de sistemas utilizando essa tecnologia.

ANEXOS

INTERFACE DO SISTEMA

Figura 42 – Página Home.aspx

Figura 43 – Página HomeAdmin.aspx

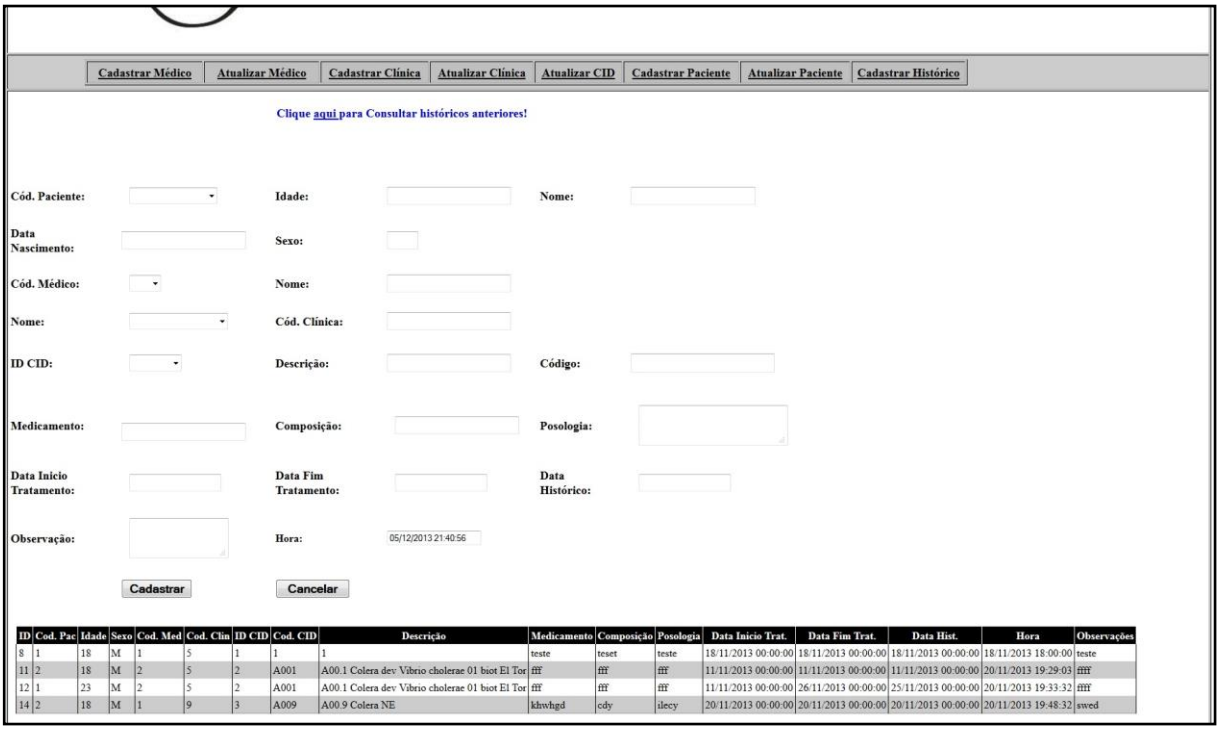
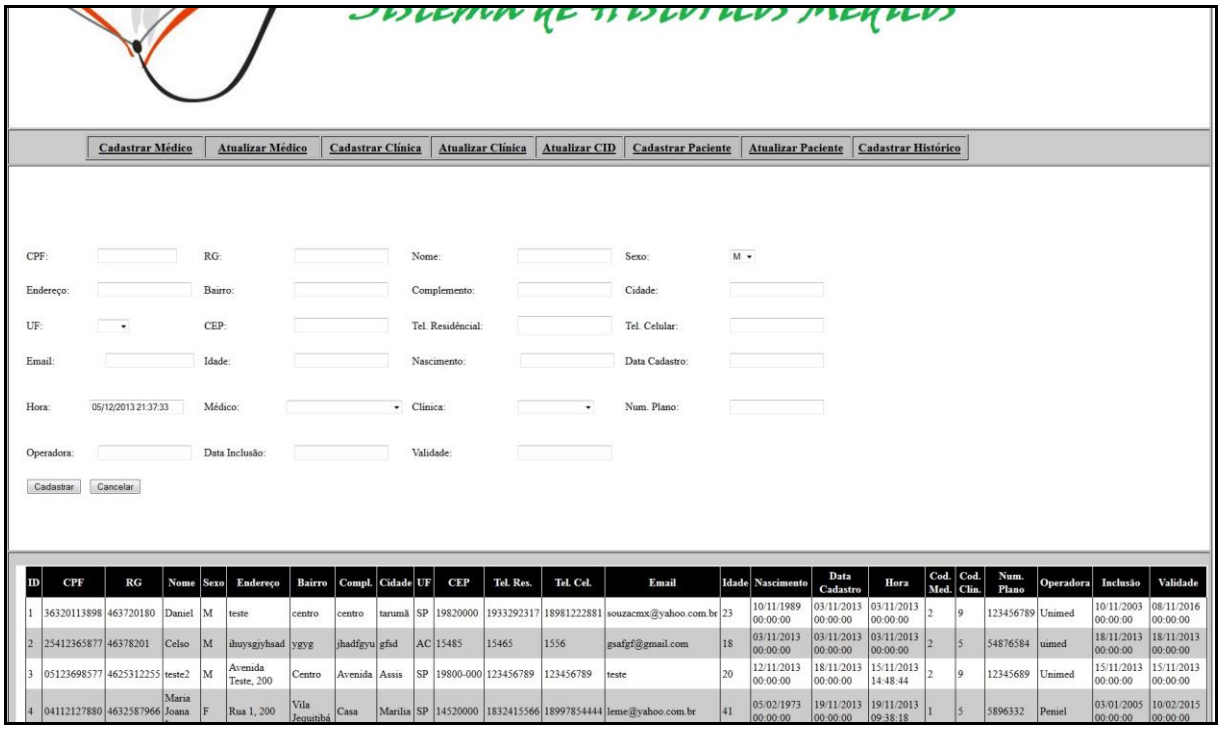


Figura 44 – Página CadastrarHistorico.aspx

Figura 45 – Página AtualizaPaciente.aspx



ID	CPF	RG	Nome	Sexo	Endereço	Bairro	Compl.	Cidade	UF	CEP	Tel. Res.	Tel. Cel.	Email	Idade	Nascimento	Data Cadastro	Hora	Cod. Med.	Cod. Clin.	Num. Plano	Operadora	Inclusão	Validade
1	36320113898	463720180	Daniel	M	teste	centro	centro	tarumã	SP	19820000	1933292317	18981222881	souzacmx@yahoo.com.br	23	10/11/1989 00:00:00	03/11/2013 00:00:00	03/11/2013 00:00:00	2	9	123456789	Unimed	10/11/2003 00:00:00	08/11/2016 00:00:00
2	25412365877	46378201	Celso	M	ihuysgiyhsad	ygyg	jhadfgyu	gfsd	AC	15485	15465	1556	gsafgf@gmail.com	18	03/11/2013 00:00:00	03/11/2013 00:00:00	03/11/2013 00:00:00	2	5	54876584	uimed	18/11/2013 00:00:00	18/11/2013 00:00:00
3	05123698577	4625312255	teste2	M	Avenida Teste, 200	Centro	Avenida	Assis	SP	19800-000	123456789	123456789	teste	20	12/11/2013 00:00:00	18/11/2013 00:00:00	15/11/2013 14:48:44	2	9	12345689	Unimed	15/11/2013 00:00:00	15/11/2013 00:00:00
4	04112127880	4632587966	Maria Joana	F	Rua 1, 200	Vila Jequitibá	Casa	Marília	SP	14520000	1832415566	18997854444	leme@yahoo.com.br	41	05/02/1973 00:00:00	19/11/2013 00:00:00	19/11/2013 09:38:18	1	5	5896332	Peniel	03/01/2005 00:00:00	10/02/2015 00:00:00

Figura 46 – Página CadastrarPaciente.aspx

Figura 47 – Página AtualizarCID.aspx

Figura 48 – Página AtualizarClinica.aspx

Figura 49 – Página CadastrarClínica.aspx

Figura 50 – Página AtualizaMedico.aspx

Figura 51 – Página CadastrarMedico.aspx

Figura 52 – Página PesquisarHistorico.aspx

REFERENCIAS

ANVISA-Agência Nacional de Vigilância Sanitária. **Bulário Eletrônico.** OLIVEIRA, Milton. GlaxoSmithKline Brasil Ltda. Disponível em: < http://www4.anvisa.gov.br/base/visadoc/BM/BM%5B25700-1-0%5D.PDF> Acesso em 14/03/2012.

ARAÚJO, Marcos Antônio. **Artigo SQL Magazine 35: DB Designer, uma ferramenta gratuita para modelagem de dados.** Disponível em: <http://www.devmedia.com.br/dbdesigner-uma-ferramenta-gratuita-para-modelagem-de-dados-artigo-sql-magazine-35/6840>. Acesso em 03/07/2013.

Artigos Oficina da Net. **SQL Server.** Disponível em: <http://www.oficinadanet.com.br/artigo/501/sql_server>. Acesso em 08/07/2013.

Biblioteca Online Microsoft. **Visual C#.** Disponível em: <http://msdn.microsoft.com/pt-br/library/vstudio/kx37x362.aspx>. Acesso em 09/07/2013

Biblioteca Online Microsoft. **Visão geral sobre o ASP.Net.** Disponível em: <http://technet.microsoft.com/pt-br/library/cc728044%28v=ws.10%29.aspx>. Acesso em 06/07/2013

Change Vision Inc. **astah Basic Operation Guide.** Change Vision Inc., 2009. Edição Digital.

fabFORCE.net. **DB Designer 4 Documentation.** fabFORCE.net, 2003. Edição Digital.

GOUVEIA, Dep. Roberto. **Projeto de Lei 546/97, Lei Nº 10.241 de 17 de março de 1999.** Disponível em: <http://dobuscadireta.imprensaoficial.com.br/default.aspx?DataPublicacao=19990318&Caderno=DOE-I&NumeroPagina=1> Acesso em 10/03/2013

GUEDES, Gilleanes T. A. UML2 **Uma abordagem prática**. São Paulo: Novatec Editora, 2009.

GUNNERSON, Éric. **Introdução a Programação em C#.** Editora Ciência Moderna, 2001.

CAMACHO JÚNIOR, Carlos Olavo de Azevedo. **Guia prático para o desenvolvimento de Aplicações C# em Camadas.** Florianópolis, Visual Books Editora, 2008.

LARMAN, Craig. **Utilizando UML e Padrões.** Tradução de Rosana Vaccare Braga ... [et al.]. – 3. Ed. – Porto Alegre, Editora Bookman, 2007.

LEE, Richard C e TEPFENHART, William M. **UML e C++ Guia Prático de Desenvolvimento Orientado a Objeto.** Tradução de Celso Roberto Paschoa. São Paulo, Editora Morkron books Ltda, 2001.

LIMA, Edwin. **C# e .Net para desenvolvedores.** Rio de Janeiro, Editora Campus, 2002.

LOTAR, Alfredo. **Como programar em ASP.Net e C#.** 2ª Edição. São Paulo: Novatec Editora, 2010.

Microsot Corporation. **Introduction to Visual Studio Technologies.** Apress Books, 2012. Edição Digital.

MISTRY, Ross e MISNER, Stacia. **Introducing Microsoft SQL Server 2012.** Microsoft Press, 2012. Edição Digital.

Organização Mundial de Saúde. **CID10-Código Internacional de Doenças.** Editora da Universidade de São Paulo-EDUSP.

P. S., Marcos. **O que é Visual Studio?** Disponível em: <http://engineer-scraps.blogspot.com.br/2012/02/o-que-e-visual-studio.html>. Acesso em 03/07/2013

PACIEVITCH, Yuri. **C#.** Disponível em: <http://www.infoescola.com/informatica/c-sharp/>. Acesso em 09/07/2013

PEREIRA, Ana Paula. **O que é CSS?** Disponível em: <http://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>. Acesso em 07/07/2013

Project Management Institute, Inc. **Um guia do Conjunto de Conhecimentos em Gerenciamento de Projetos (Guia PMBOK).** Project Management Institute, Inc., 2004.

SILVA, Mauricio Samy. **Tutorial em HTML**. Disponível em: <http://pt-br.html.net/tutorials/html/lesson2.php>. Acesso em 08/07/2013

SILVA, Mauricio Samy. **jQuery: a biblioteca do programador JavaScript.** Segunda Edição. São Paulo: Editora Novatec, 2008.

TERUEL, Evandro Carlos. **HTML 5: Guia Prático.** Primeira Edição. São Paulo: Editora Érica, 2011.

TROELSEN, Andrew. **Profissional C# e a Plataforma .NET 3.5 Curso Completo.** Rio de Janeiro: Alta Books Editora, 2010.

THOMAS, Alexandre. BARASHEV, Dimitry. **Learn Gantt Project**. Disponível em: <http://www.ganttproject.biz/learn#faq>. Acesso em 08/07/2013