Faculdade de Tecnologia de Americana

Curso Análise e Desenvolvimento de Sistemas

Murilo Gabriel Neves Luder

Proposta de projeto do sistema gerenciador de multimídia – File Manager

Americana, SP

2016

Faculdade de Tecnologia de Americana

Curso Análise e Desenvolvimento de Sistemas

Murilo Gabriel Neves Luder

Proposta de projeto do sistema gerenciador de multimídia – File Manager

Trabalho de Conclusão de Curso, desenvolvido em cumprimento à exigência curricular do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Msc Wagner Siqueira Cavalcante.

Área de concentração: Análise e Desenvolvimento de Sistemas.

Americana, SP

2016

- Ficha Catalográfica –

L975p LUDER, Murilo Gabriel Neves
Proposta de projeto do sistema gerenciador de multimídia - *file manager*. / Murilo Gabriel Neves Luder. – Americana: 2016.
73f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Ms. Wagner Siqueira Cavalcante

1. Engenharia de software I. CAVALCANTE, Wagner Siqueira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.3.05

Murilo Gabriel Neves Luder

Proposta de projeto do sistema gerenciador de multimídia - File Manager

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/Americana.
Área de concentração: Engenharia de Software.

Americana, 07 dezembro de 2016.
Banca Examinadora:

Wagner Siqueira Cavalcante (Presidente)

Mestre

Faculdade de Tecnologia de Americana - FATEC

Rodrigo Viviani (Membro)

Especialista

Faculdade de Tecnologia de Americana - FATEC

Rogério Nunes de Freitas (Membro)

Especialista

Faculdade de Tecnologia de Americana - FATEC

AGRADECIMENTOS

Em primeiro lugar queria agradecer aos meus pais por estarem sempre ao meu lado, e em segundo ao meu orientador, Wagner Siqueira Cavalcante, pelo apoio durante a execução desse trabalho.

RESUMO

O presente texto conceitua o produto de engenharia de *software*[1], para o gerenciamento de documentos, intitulado "File Manager", que poderá ser desenvolvido para instituições policiais, com o objetivo de resolver o problema quanto ao acúmulo de documentos impressos presente em seus recintos.

Durante a leitura do texto, será possível identificar a abordagem a ser utilizada para desenvolver o *software*, além de o presente ter sido escrito com o intuito de que qualquer pessoa, mesmo sem conhecimento na área de desenvolvimento de *software* e afins, consiga entender os processos realizados que serão abordados e como se dará o funcionamento do mesmo.

Serão abordados os conceitos da Engenharia de Software, ciclo de vida, dos requisitos funcionais e não funcionais, bem como serão apresentados alguns diagramas referentes ao produto, File Manager. Através da leitura dos itens citados, o leitor poderá ter uma compreensão de como será projetado o desenvolvimento do software.

Com a realização de todos os processos contidos nessa documentação, será possível resolver o problema enfrentado pelas instituições polícias referente ao acúmulo excessivo de documentos nos estabelecimentos.

Palavra Chave: Acúmulo de documentos, desenvolvimento de *software*, processos realizados.

[1] Conjunto de programas e toda a documentação associada a ele.

ABSTRACT

The present text conceptualizes the File Manager software engineering product, which will be developed for police institutions, with the goal of solving the problem of the current documents accumulation in police station.

During the reading, it will be possible to identify an approach that will be used to develop the software. In addition, the present text has been written to make anyone, even without any knowledge in software development and related area, understand the process realized that will be discussed and how the work of system will be.

Concepts of software engineering, system lifecycle, functional and nonfunctional requirements will be approached, as well as the File Manager product diagrams. Through the reading of the aforementioned items, the reader may understand how the software development will be designed.

By accomplishing all the processes contained in this documentation, it will be possible to solve the problem faced by the police institutions regarding the excessive accumulation of documentation.

Keyword: Documents accumulation, software development, processes realized.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

MER	MODELO ENTIDADE RELACIONAMENTO
DER	DIAGRAMA ENTIDADE RELACIONAMENTO
ERP	ENTERPRISE RESOURCE PLANNING
GDL	GERENCIADOR DE LAUDOS
RF	REQUISITOS FUNCIONAIS
RNF	REQUISITOS NÃO FUNCIONAIS
SGBD	SISTEMA GERENCIADOR DE BANCO DE DADOS
RA	REGISTRO DO ALUNO
UML	LINGUAGEM DE MODELAGEM UNIFICADA
TI	TECNOLÓGIA DA INFORMAÇÃO
POO	PROGRAMAÇÃO ORIENTADA A OBJETOS
IDE	AMBIENTE DE DESENVOLVIMENTO INTEGRADO
VV&T	VALIDAÇÃO, VERIFICAÇÃO E TESTE
SQL	LINGUAGEM DE CONSULTA ESTRUTURADA

SUMÁRIO

INTRODUÇÃO

O objetivo principal desse trabalho é documentar os processos e métodos que serão necessários para se atender a proposta de desenvolvimento do *software* File Manager (FM).

Primeiramente é necessário compreender o que é um *software*, para assim ser possível prosseguir com o entendimento do projeto em questão.

Um sistema pode ser definido como sendo um conjunto de programas que quando executados fornecem informações características, funções e desempenho desejados, acrescidos de uma estrutura de dados que possibilita aos programas manipular informações de forma adequada, bem como da informação descritiva, tanto de forma impressa quanto na forma virtual, descrevendo a operação e o uso dos programas (PRESSMAN, 2011).

Se os *softwares* não tivessem sido utilizados como fonte de geração de dinheiro, ou seja, um produto, os mesmos não teriam chegado onde estão agora, portanto, seu desenvolvimento teria sido lento. O sistema proposto nesse projeto de engenharia, não se categoriza como um *software* livre, devido ao fato de o mesmo apresentar fins lucrativos.

O *software* File Manager terá como base a instituição técnico-científica, no que diz respeito ao acúmulo excessivo de documentos impressos.

Nas empresas e instituições, na maioria dos casos é possível identificar um excesso de documentações de forma impressa, e que muitas vezes não são guardadas de forma adequada ou mesmo organizada, o que futuramente pode vir a acarretar uma sobreposição de documentos.

As empresas hoje em dia estão notando a necessidade de preservar a natureza. Em vista disto, tecnologias estão sendo desenvolvidas para suprimir a necessidade da utilização de documentações impressas. Certamente, a melhor forma de se lidar com isso é fazer uso de um *software* que tem por objetivo principal armazenar documentos, como o *software* File Manager, aqui proposto.

Sendo esse um dos problemas enfrentados pelas empresas e instituições, este *software* deve ser desenvolvido com o objetivo de eliminar esse problema de acúmulo excessivo de documentações, e fazer com que a empresa se torne um ambiente mais organizado em relação a sua documentação.

Esse projeto de desenvolvimento de *software* tem por objetivo principal eliminar o volume de documentos impressos nos departamentos da polícia técnico-científica. Entretanto,

existem outros problemas que essas instituições enfrentam que o sistema também precisará solucionar, afinal, um grande problema precisa de outras pequenas soluções para poder de fato ser resolvido.

O segundo capítulo reserva-se a expor conceitos fundamentais de Engenharia de Software, criando uma base de entendimento desta área de conhecimento para o leitor.

O terceiro capítulo abordará o problema principal enfrentado pela instituição, assim como os secundários que também necessitam de correção.

O quarto capítulo abordará possíveis soluções para os problemas citados no terceiro capítulo, aplicando-se os conhecimentos discorridos no capítulo 2, fazendo com que o leitor entenda o porquê de cada etapa ser realizada naquela ordem e daquele modo como foram propostos.

O quinto capítulo, reserva-se às considerações finais, sugerindo adições possíveis ao *software* File Manager.

1. FUNDAMENTAÇÃO

Quando se desenvolve um *software,* tenha ele um objetivo específico ou não, é necessário adotar formas de proceder em seu desenvolvimento. Desde o ano de 1968, quando *hardwares* de computadores de terceira geração estavam sendo implantados, notou-se que existia uma deficiência nos processos de desenvolvimento de *softwares*, quanto mais complexo se tornavam, mais atrasos geravam, e isso devido à adoção de medidas informais aplicadas no desenvolvimento do mesmo, propiciando o surgimento da Engenharia de *Software* (SOMMERVILLE, 2003, p.4).

No desenvolvimento de um *software*, a base do mesmo se deriva da Engenharia de Software, termo referente à própria Engenharia, que faz menção a construir algo. Ou seja, existe a necessidade de entender e estudar o todo, antes de construir um *software*, evitando assim, gerar mais problemas à empresa do que soluções ao problema.

Segundo Sommerville (2003, p.5), "Engenharia de Software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de *software*, desde os estágios iniciais de especificação do sistema até a sua manutenção, depois que ele entrou em operação".

A Engenharia de Software é a língua utilizada no âmbito da área da tecnologia da informação (TI), que tem como meta fornecer normas e padrões a serem seguidos no desenvolvido de um *software*, a fim de obter uma melhor qualidade do mesmo.

Entendendo o que é Engenharia de Software, torna-se possível escolher qual modelo de ciclo de vida será utilizado, ou seja, qual o modelo que será adotado no desenvolvimento do *software*.

Antes de começar a discorrer sobre alguma metodologia a se aplicar, é necessário explicar o que é ciclo de vida.

1.1. Ciclo de Vida

Os ciclos de vida do *software* são as etapas pelas quais um *software* vai precisar passar para ser desenvolvido, ou seja, os processos, atividades e tarefas. Este método vai abranger desde o levantamento de requisitos até o final do processo de desenvolvimento, fornecendo assim um controle sobre tudo o que for feito para a projeção do *software*.

Tendo isso em mente, a escolha do ciclo de vida torna-se fundamental, pois esta será a primeira decisão a ser realizada no planejamento do *software*. Entretanto, não existe um modelo pré-definido, ou seja, são os fatores externos que irão influenciar na escolha.

Como dito anteriormente, esses processos do ciclo de vida serão agrupados em fases, sendo elas: definição dos requisitos, análise, projeto, desenvolvimento, teste e implantação. Nessas etapas também serão definidas as atividades dos integrantes da equipe, assim como a importância de cada função.

Assim como nas atividades exercidas por um membro da equipe, onde estas são compostas por diferentes níveis de importância, isso também irá acontecer em relação as fases citadas anteriormente. A ordem, o tempo e as atividades exercidas em cada etapa têm níveis diferentes de importância.

Existem diversos modelos de ciclo de vida, que, de certa forma, podem ser classificados como:

Sequencial (cascata, clássico ou *waterfall*): aplica-se quando há uma notória ordem determinada de etapas.

Incremental: aplica-se quando o sistema aumenta conforme a necessidade, ou seja, acrescenta funcionalidades ou as modifica conforme a necessidade explicitada pelo cliente.

Iterativo: permite aperfeiçoar o *software* de forma gradual, fazendo com que a equipe tenha o trabalho de fazer o levantamento de requisitos em cada iteração.

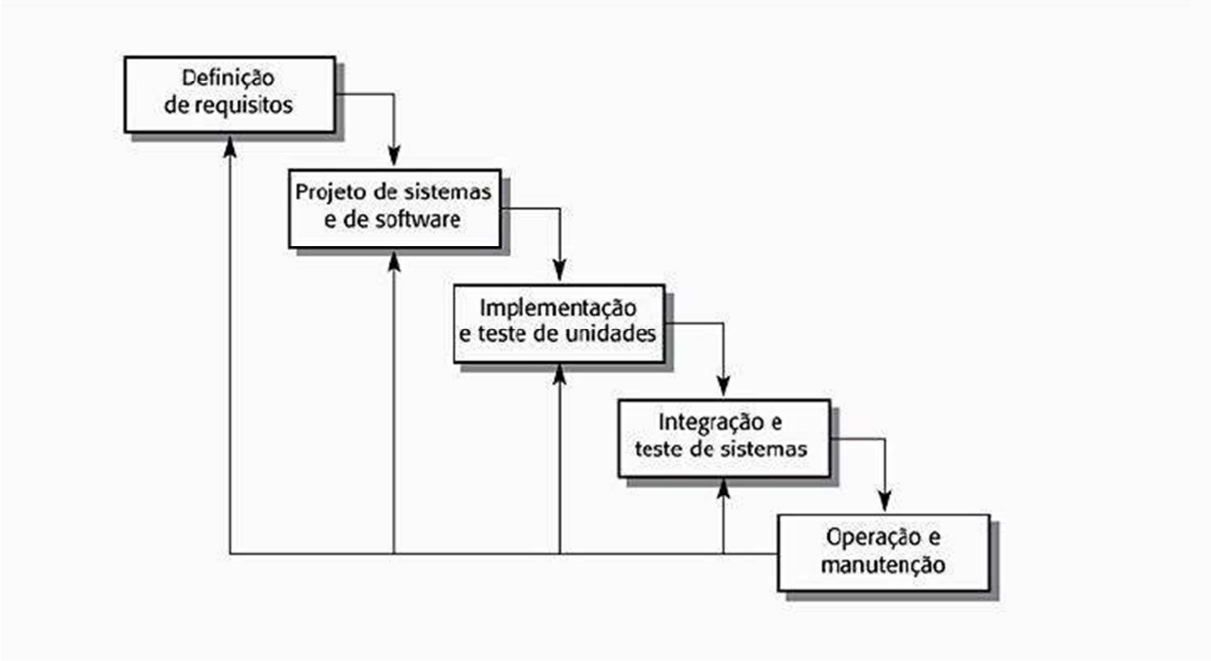
Evolutivo: este permite o desenvolvimento a partir de protótipos desenvolvidos inicialmente (prototipação, esta podendo ser exploratória, experimental ou evolutiva), sendo eles protótipos de apresentação, protótipos autênticos, protótipos funcionais e sistemas piloto.

O modelo sugerido para o desenvolvimento desse *software* é o Ciclo de Vida em Cascata, também denominado como “clássico”.

1.1.1. Modelo Cascata

O modelo em cascata foi o primeiro método a utilizar fases pré-definidas, fornecendo um desenvolvimento formal e continuo do *software*. Este ciclo de vida apresenta esse nome devido as suas fases serem executadas em forma cascata, uma fase após a outra (SOMMERVILLE, 2003, p.37).

Figura 1 – Modelo Cascata



Fonte: SOMMERVILLE, 2003, p.38.

O modelo em cascata foi o primeiro a colocar em uso o planejamento de fases e o gerenciamento dos processos e atividades exercidas. Este modelo, como pode ser observado na Figura 1, faz com que seja necessário concluir uma etapa para prosseguir para a próxima no desenvolvimento. Toda fase que é concluída gera um documento aprovando a continuidade do projeto, como a chave para se entrar na etapa seguinte.

Devido a sua simplicidade, é possível mostrar ao cliente como vai ser o desenvolvimento do sistema, apresentando as etapas que serão abordadas e estimando também um custo no início da projeção.

Apesar de ser um modelo de simples entendimento e suas etapas bem estabelecidas, o fator humano é presente, é isso pode gerar diversos problemas resultantes da ação do mesmo, ou seja, mesmo com as fases bem definidas existem problemas que podem vir a ocorrer.

1.2. Requisitos

Com a explicação sobre o que é um ciclo de vida, assim como o que é o modelo cascata, torna-se necessário explicar o que são requisitos, bem como requisitos funcionais e requisitos não funcionais.

Os requisitos são as necessidades a serem atendidas pelo sistema, advindas de várias fontes (o próprio cliente, o usuário, o ambiente corporativo, as leis e diretrizes externas, as restrições computacionais, etc.), além de constituírem uma parte fundamental para a elaboração de um projeto de *software*, e consequentemente o sistema em si. Este vai englobar as funções, propriedades, objetivos, restrições e padrões que o mesmo a fim de atender o objetivo proposto.

Quando feito o levantamento de requisitos, estes devem atender as necessidades do cliente, fornecendo uma solução para seu problema. Estes são divididos em duas categorias: Requisitos Funcionais (RF) e Requisitos Não-Funcionais (RNF).

A conversa entre o líder da equipe de desenvolvimento e o cliente se tratando do levantamento de requisitos, é uma tarefa difícil para ambas as partes, cenários dos quais podem fornecer base para entendimentos diversos tanto, por parte tanto do cliente, no que diz respeito ao que de fato deseja, quanto ao líder de equipe, que pode ter uma compreensão diferente do desejo real do cliente.

1.2.1. Requisitos Funcionais

Os requisitos funcionais estão associados ao que o *software* deve fazer, sendo assim representa as funções, operações e informações do sistema. Estes vão estar preocupados em atender as funcionalidades e serviços do programa, ou seja, as funções e como se dará o funcionamento do mesmo.

1.2.2. Requisitos Não Funcionais

Segundo Mendes (2008), os requisitos não funcionais estão associados ao que qualifica os requisitos funcionais, ou seja, dedicam-se a definir propriedades e restrições (muitas vezes subjetivas) do sistema, tais como tempo, espaço, linguagem de programação, versão do compilador, SGBD, sistema operacional, método de desenvolvimento, legalidade, aparência da interface, entre outros.

No desenvolvimento de um *software*, como já dito anteriormente, o levantamento de requisitos é a parte fundamental, sendo assim, cada sistema terá seu conjunto específico de requisitos além de ter suas próprias prioridades.

Os requisitos obtidos no começo e durante o projeto são utilizados em diversas fases, até mesmo quando o software está finalizado, e é feita a bateria de testes. Esses testes são elaborados em cima do que coletou-se no começo do projeto, para ver se o sistema atendeu as exigências do cliente e se de fato tais requisitos, sendo atendidos, solucionam o problema que a instituição em questão enfrenta.

1.3. Modelo Entidade Relacionamento

Os programas devem se adequar à proposta do projeto. Para este projeto em questão, serão utilizadas ferramentas aprendidas durante o decorrer do curso.

Após o levantamento dos requisitos, são identificadas as principais partes do *software,* relação exercida entre elas, objetivos que devem ser atendidos, ações a serem executadas com prioridades e as características que o mesmo deve possuir.

Com essas informações obtidas, é possível criar o Modelo Entidade Relacionamento (MER), que seria um modelo conceitual daquilo que será elaborado durante o projeto.

O Modelo Entidade Relacionamento é um modelo conceitual que é responsável por descrever os objetos, chamados de entidades, envolvidos dentro de um domínio, além de suas características, chamados atributos, e o relacionamento entre tais objetos.

Segundo Rodrigues (2014), quando fala-se sobre o MER, refere-se à estrutura abstrata do banco de dados, daquilo que estará contido no mesmo. Devido à complexidade e tamanho de alguns *softwares*, a elaboração de um modelo entidade relacionamento para o sistema inteiro se torna inviável. Dito isto, uma solução utilizada em muitos projetos é a elaboração de modelos para cada módulo do *software*, tornando mais simples o entendimento sobre o mesmo.

Para se desenvolver esse modelo, é necessário ter compreensão do que estará contido no mesmo, ou seja, obter conhecimento sobre o que é entidade, atributo e relacionamento.

Uma entidade deve ser compreendida como sendo algo ou um objeto distinguível de outro objeto (uma pessoa, um computador, uma ideia, uma espécie animal, etc.). As entidades podem ser tanto físicas quanto lógicas, sendo que a primeira diz respeito àquilo que é

tangível, ou seja, a algo palpável, que consequentemente, é visível também. Qualquer coisa presente no cotidiano, que possa ser palpável é uma entidade física, por exemplo, um ônibus, uma pessoa, um computador, entre outros. As entidades lógicas, no entanto, estão associadas a existência em um certo contexto, e que não fariam sentido fora deste, ou seja, no mundo real propriamente dito, isto devido a não ocuparem um espaço físico. Quando classifica-se um animal por sua raça, ou um cargo de um funcionário, por exemplo, a isso corresponderiam as entidades lógicas, devido a não ocuparem de fato um espaço físico.

O nome dado para cada entidade tem que representar de forma clara o que aquela entidade representa, por exemplo: raça, cargo, computador, carro, etc. As entidades também são classificadas conforme a necessidade da sua existência, portanto existem três tipos: entidade forte, entidade fraca e entidade associada.

Quando classificadas, as entidades compõem conjuntos próprios, que agrupam apenas as entidades semelhantes, aquelas que agrupam o maior conjunto de características observáveis possível para cada entidade, ao que intitula-se “conjunto de entidades”, ou seja, um conjunto de entidades comporta apenas entidades absolutamente semelhantes e, portanto, isola em outros conjuntos de entidades aquelas condizentes com cada conjunto. Alguns exemplos são todos os funcionários pertencem ao conjunto Funcionários, que, embora muito parecidos, são diferentes dos clientes, que, por sua vez, pertencem ao conjunto Clientes, assim como a entidade produto, pertence ao conjunto Produto, é totalmente distinta, tanto de funcionário, quanto de cliente, pois possui um conjunto de características absolutamente distinto de ambas.

É muito fácil compreender que um funcionário e um produto são entidades absolutamente distintas, porém, para se distinguir um funcionário de outro em seu conjunto, assim como distinguir um produto de outro em seu conjunto, deve-se escolher (ou criar) alguma característica (denominada “atributo”) que permita tal distinção, ao que nomina-se de “chave primária” (como é o CPF para a Receita Federal, um código de cliente para uma empresa, o chassis de um automóvel para a fábrica e Polícia, etc.), que contém valor único e unívoco (cada entidade tem seu próprio valor de chave primária, portanto não há como haver duas ou mais com tal valor).

O conjunto de entidades forte é aquele que não necessita de outro conjunto de entidades para existir, sendo assim, existe por si só, pois suas entidades possuem sentido próprio, são independentes e, portanto, têm uma chave primária. Ao contrário do anterior, o

Conjunto de Entidades Fraca necessita de outro conjunto de entidades para existir, do qual é dependente, e, portanto, sua chave primária é composta pela chave primária da entidade de quem esta depende. O conjunto de entidades associado passa a existir quando há um relacionamento de muitas entidades de um conjunto de entidades com muitas outras entidades de outro conjunto de entidades (chamado relacionamento “muito para muitos”), cujos valores só existirão na ocorrência de entidades que necessariamente relacionam-se, advindas de ambas as envolvidas, o que será explicado a seguir. Essa entidade funciona como uma mediadora entre outras duas, formada assim pela chave primária de ambas.

Entendido o que são entidades e quais são seus derivados, é hora de entender a respeito de relacionamento entre entidades.

Depois de coletar os requisitos, analisá-los e então definir as entidades, é necessário então relacioná-las, e isso ocorre dependendo da quantidade de objetos presentes em cada uma e a quantos de outra cada um pode se relacionar. Esses relacionamentos realizam-se naturalmente, e devem ser representados através de cardinalidades, ou seja, número de ocorrências feitas ou recebidas por outra determinada entidade. A letra “N” representa muitas ligações, 1 representa apenas uma única ligação e 0 representa nenhuma ligação.

Este relacionamento é a representação lógica da relação entre as entidades e pode ser dividido em: relacionamento “um para um” (1..1), “um para muitos” (1..N ou 1..*) ou “muitos para muitos” (M..N ou *..*), este último já citado durante a explicação das entidades associativas.

O relacionamento do tipo “um para um” ocorre entre uma entidade de um conjunto de entidades e outra entidade de outro conjunto e vice-versa, porém de forma exclusiva, ou seja, quando uma entidade se relaciona com outra, este relacionamento se torna exclusivo não havendo possibilidade de outra relação, o que provoca que a chave primária de um conjunto siga para o outro como chave estrangeira sem direito a repetições, e isto ocorre em apenas uma das direções. Por exemplo, entre os conjuntos de entidades relacionados, “um para um”, Funcionário e Sessão, somente um funcionário pode ser gerente de uma sessão e uma sessão poderá ter exclusivamente um funcionário como gerente.

O relacionamento do tipo “um para muitos”, ou “muitos para um”, caracteriza-se por uma entidade de um conjunto poder se relacionar com muitas de outro conjunto (a chave primária do primeiro conjunto torna-se estrangeira no segundo, cujos valores equivalem-se), porém, cada entidade deste segundo conjunto pode se relacionar com apenas um da primeira.

Como exemplo, quando um aluno ingressa na universidade, este vai cursar um único curso, sendo que cada curso tem diversos alunos associado a ele.

A cardinalidade do tipo “muitos para muitos” caracteriza-se pelo fato de permitir que uma entidade de um conjunto possa relacionar-se com muitas do outro conjunto, sendo que cada entidade deste segundo conjunto também pode se relacionar com muitos do primeiro. O efeito deste relacionamento provoca a criação de uma tabela intermediária, cujos atributos são, inicialmente, a chave primária de cada entidade envolvida, aqui como estrangeiras, além de atributos descritivos, caso existam, que são os que referem-se ao relacionamento em si. Por exemplo, um aluno pode matricular-se em várias disciplinas, cada qual podendo ter vários alunos a ela matriculados. Se a data em que o aluno matricula-se deve ser armazenada, esta não pertence nem ao aluno, nem à disciplina, porém ao relacionamento em si, o que faz com que a tabela “Matrícula” conterá os atributos chave primária de aluno e chave primária da disciplina como estrangeiras e compondo a sua chave primária, e o atributo descritivo data de matrícula.

Ainda segundo Rodrigues (2014), os atributos são características das entidades, aquilo que a descreve, usando como exemplo os seres humanos, estes apresentam cinco dedos, dois olhos, duas orelhas, pigmentação da pele, entre outros. São características que definem o ser humano, o mesmo se dá com as entidades.

Podem-se encontrar três tipos de classificação, sendo possível encontrar atributos descritivos, atributos nominativos ou atributos referenciais.

Os atributos descritivos fazem uso das características intrínsecas da entidade, ou seja, pertencente à mesma. Exemplos: sexo, idade, pigmentação da pele, etc.

Os atributos nominativos fazem além da função dita anteriormente, também a de definir identificações a entidade que pertencem. Exemplos: Código do aluno, número de telefone do aluno, número de matrícula.

E, por último, o atributo referencial, que assim como o nome sugere, faz referência a outras entidades, portanto não fazem parte, propriamente dito, do conjunto onde estão.

1.4. Linguagem de Modelagem Unificada (UML)

A UML, na Engenharia de Software, é uma linguagem de modelagem unificada que permite padronizar a representação de um sistema. Ela foi criada por Grady Booch, Ivar Jacobson & Jaimes Rumbaugh, tendo como objetivo padronização na especificação,

documentação e estruturação do sistema, de forma a facilitar o entendimento da lógica completa do sistema.

1.4.1. Diagramas da UML

Há quatorze modelos de diagramas na versão 2.2 da UML, entretanto boa parte das ferramentas CASE disponibiliza os principais, o que varia de uma ferramenta para outra. Deve-se ressaltar que todos têm o mesmo objetivo geral, que é proporcionar um maior entendimento para a equipe no todo, evidenciando os processos que devem ser elaborados, todavia cada um tem objetivo específico, seja para evidenciar as classes e métodos a serem criados, seja para delinear a sequência das operações, entre outros.

Os diagramas elaborados para o sistema variam conforme a necessidade e complexidade do *software*, como já dito, e facilitam na estruturação e visualização do que será feito. Porém, em muitos casos não é necessário elaborar todos os modelos de diagramas.

A descrição de cada um dos diagramas que serão abordados no decorrer do projeto será apresentada a seguir, sendo eles o Diagrama de Classe, Diagrama de Caso de Uso e Diagrama de Sequência.

1.4.1.1. Diagrama de Classe

O objetivo do diagrama de classe é descrever os tipos de objetos e o relacionamento entre eles. É possível fazer este modelo em três perspectivas: conceitual, especificação e implantação, através dos quais é possível mostrar a pessoas de diferentes níveis técnicos, facilitando assim a comunicação.

A perspectiva conceitual está voltada ao cliente, que deve representar os conceitos do domínio da aplicação, respeitando-se as regras de negócio.

A perspectiva de especificação é destinada aos gerentes, que necessitam de um pouco mais de informação do que o cliente, porém não tão aprofundado. Este deve abordar os métodos e classes principais da estrutura do projeto.

A perspectiva de implantação está voltada para a equipe, que abordará de forma detalhada tudo aquilo que é necessário ao sistema, seguindo as restrições estabelecidas pelo e para o sistema de informação, ou seja, métodos, atributos, visibilidades e relacionamentos.

1.4.1.2. Diagrama de Caso de Uso

Segundo UFCG (2008), os diagramas de caso de uso têm a função primordial de mediar à conversa entre a equipe e o cliente, apresentando-lhe uma visão ampla sobre as funcionalidades a serem atendidas, sem ressaltar aspectos técnicos de T.I. Neste cenário serão apresentados atores, casos de uso e o relacionamento que se dá entre eles.

É possível haver relacionamentos entre os atores e os casos de uso e entre os casos de uso e outros casos de uso (através de execuções opcionais – *extend*, ou obrigatórias – *incude*). Algumas vezes, quando elaborado o diagrama, é imposto um retângulo em volta dos casos de uso a fim de mostrar a delimitação do sistema.

Os atores são normalmente pessoas usuárias do sistema e que executam tarefas, bem como as que recebem os resultados delas, o que também pode ocorrer com outros sistemas interativos.

Os casos de uso são as funções e operações que ocorrerão no sistema.

Já os relacionamentos descrevem interações que podem ocorrer entre atores e casos de uso, e casos de uso com outros casos de uso; destaca quem executa a operação em questão e a quem o resultado de tal operação será destinado. Quando existe a ligação de um ator com um caso de uso, pode-se ter uma associação, ou seja, do ponto de vista do usuário aquele caso de uso assume uma funcionalidade no sistema.

Quando há uma relação entre atores, pode existir uma generalização, que é uma forma elegante e de simplificação de entendimento e de definição de código, para dizer que os atores, embora distintos por força de alguns detalhes, são semelhantes, por compartilhar atributos comuns. Um exemplo pode ser a relação que há entre vendedor e cliente, realizada através da venda/compra de produto, visto que ambos têm muitas características comuns, pois são pessoas, porém mantêm algumas distintas, o que caracteriza que um vende e o outro compra, um está constante na empresa e dela é empregado e o outro surge eventualmente.

O relacionamento entre casos de uso pode ocorrer de três formas, sendo elas: *include*, *extend* ou generalização. O primeiro relacionamento será tratado como uma dependência, ou seja, quando há uma ligação *include,* o caso de uso que está apontado necessariamente será executado.

Já a ligação *extend* trata como opcional a existência do outro, podendo ou não vir a consultar ou realizar a função do outro caso de uso, ou seja, não é essencial.

Por último, tem-se o relacionamento de generalização, que tem como propósito generalizar um caso de uso, herdando assim todas as características do pai.

1.4.1.3. Diagrama de Sequência

O diagrama de sequências descreve as iterações entre as classes por meio das trocas de mensagens ao longo do tempo. Este vai mostrar o comportamento do sistema e, quando realizada uma ação, qual consequência este vai gerar no *software*.

As iterações do diagrama são lidas de cima para baixo, ou seja, aquelas no topo são executadas antes das que estão abaixo, seguindo-se a linha de tempo, como se este fosse iniciado no topo, encerrando-se no final da linha vertical. Este diagrama também mostra as mensagens que serão exibidas quando realizadas alguma ação.

Os diagramas descritos anteriormente serão aqueles abordados na proposta de desenvolvimento desse projeto.

1.5. Validação, Verificação e Teste

Quando desenvolvido um *software*, levando em consideração suas dimensões e complexidade, acaba por acarretar alguns problemas no final do projeto. A principal fonte dos problemas que surgem no sistema é devido a falhas humanas, ou seja, erros derivados das capacidades humanas.

Antes de o *software* entrar em funcionamento, ou seja, ser disponibilizado para os usuários utilizarem, é realizado uma série de atividades denominadas VV&T, está será responsável por descobrir os erros contidos no sistema.

As atividades realizadas pela VV&T tem cunho estático ou dinâmico, sendo que no primeiro não necessita de um programa para serem usados, enquanto as atividades dinâmicas utilizam de um programa ou modelo para serem realizadas.

1.5.1. Teste

Segundo Rios (2006, p.8), “teste de software é o processo que visa a sua execução de forma controlada, com o objetivo de avaliar o seu comportamento baseado no que foi especificado. A execução dos testes é considerada um tipo de validação”.

Os testes, devido ao amplo campo que estes atingem, tendem a dividir em fases de acordo com os objetivos que almejam, portanto, podem ser definidas de modo geral em fases: teste de unidade, teste de integração e o teste de sistema.

Segundo Delamaro (2007, p.47), “técnicas e critérios de testes fornecem ao projetista do *software* uma abordagem sistemática e teoricamente fundamentada para a condução da atividade de testes”.

1.5.1.1. Teste de unidade

Segundo Delamaro (2007, p.4), o teste de unidade consiste em realizar testes em partes especificas do sistema, identificando então algoritmos incorretos ou implementados de forma incorreta, estrutura de dados incorretas e erros de programação simples. Este tipo de teste é normalmente realizado pelo próprio desenvolvedor durante o desenvolvimento do *software*.

1.5.1.2. Teste de integração

Segundo Delamaro (2007, p.4), o teste de integração deve ser realizado após a execução dos testes unitários, focando então na estruturação do sistema. Esses testes vão estar focados em detectar erros durante a interação das partes, portanto, devido a necessidade de conhecimento sobre as partes e a interação que se deve ter entre elas, essas atividades devem ser realizadas pela própria equipe de desenvolvimento.

1.5.1.3. Teste de sistema

Segundo Delamaro (2007, p.4), o teste de sistema será realizado com todas as partes integradas, sendo assim, este vai prover a verificação das funcionalidades especificadas no documento de requisitos, averiguando se aquelas estão funcionando adequadamente.

Dependendo da estruturação da equipe é possível existir uma equipe independente designada para executar os testes de sistema. Os três testes apresentados anteriormente são categorias genéricas, a seguir serão explicados testes de cunho específico, que vão se enquadrar em uma dessas.

1.5.1.4. Teste Funcional

O teste funcional ou teste de caixa preta, em princípio aborda todos os dados de entrada que o sistema pode ter, fornecendo assim um teste exaustivo. Entretanto, dependendo da complexidade e magnitude do sistema, a realização de todas as entradas se torna inviável, ou mesmo impossível devido a quantidade de informações.

No *software* File Manager será possível realizar esse método, devido às dimensões não serem grandes o suficiente para impossibilitar a realização de todos os tipos de testes de entrada.

1.5.1.5. Teste Estrutural

Segundo Delamaro (2007, p.47), o teste estrutural ou teste de caixa branca vai estabelecer os requisitos de teste, fazendo uso de componentes ou partes do sistema, portanto, consegue fazer testes sobre os caminhos lógicos percorridos durante uma atividade.

Este modelo de teste fornece inúmeras limitações, isto devido diversos problemas tal como: não contem testes de cunho geral, computação de uma mesma função exercida por dois programas diferente ou por dois componentes não pode ser definida como iguais, entre outros.

Por mais que existam diversas limitações, esta se faz necessária para complementar os demais testes existentes, fornecendo testes a unidade e conjuntos distintos. As atividades realizadas por este modelo podem ser usadas a fim de prover confiabilidade ao sistema, além de ser relevante na atividade de manutenção e depuração.

O teste estrutural tem base na estrutura interna do *software*, fazendo uso das características de implantação para a criação e seleção de casos de testes adequados para a aplicação.

1.6. Linguagem de Programação

Para o desenvolvimento do *software* File Manager, será utilizada a linguagem de programação C#, lembrando que este trabalho apresenta somente a proposta de projeto para o desenvolvido do produto em questão.

1.6.1. C#

A linguagem de programa C# é voltada para a programação orientada a objetos, esta foi baseada nas linguagens de programação C++ e Java. Então é importante explicar o que seria a programação orientada a objetos.

O padrão de desenvolvimento orientado a objetos está voltado para o reaproveitamento de código, sendo possível também ter uma ideia melhor sobre a representação do sistema.

A Programação Orientada a Objetos (POO) apresenta quatro pilares, por assim dizer, que devem ser atendidos, sendo eles: abstração, encapsulamento, herança e polimorfismo (MACHADO, 2014).

A abstração é um dos pontos fundamentais neste tipo de programação, pois estamos lidando com a representação do objeto real, criando-se assim um pensamento sobre o que aquele determinado objeto vai realizar dentro do sistema. A abstração consiste em três coisas: identidade, propriedade e método.

É necessário criar uma identidade para o objeto dentro do sistema ou de um pacote, da qual não poderá ser repetida, sendo assim única, isto para que não existam conflitos dentro do *software*.

Todo objeto tem características que o marcam e isto também será representado no sistema como propriedades, por exemplo, um objeto pessoa poderia ter: altura, peso, cor, idade, entre outros.

Os métodos são ações ou eventos que aquele objeto vai ter dentro do sistema, por exemplo, um objeto carro pode ter os métodos: andar(), frear(), entre outros.

O encapsulamento, como o próprio nome diz, vai servir como uma caixa preta do *software*, fornecendo segurança, ou seja, escondendo as propriedades de um objeto. Este geralmente faz uso de métodos chamados *getters* e *setters*, que irão setar e retornar o valor de uma propriedade privada.

A herança está associada à otimização do código, ou seja, esta vai fazer uso de características e métodos de terceiros. A POO apresenta um sistema de hierarquia onde aqueles em baixo herdam as características daqueles em cima, chamados assim “ancestrais”.

Conforme a linguagem de programação utilizada é possível herdar as características de mais de um ancestral, ou seja, herança múltipla. Se tratando da linguagem de programação C#, está faz uso de um objeto base para todos os demais criados, sendo está a classe *object*, sendo assim fornece características para todos os objetos criados pelo sistema e usuário.

O polimorfismo vai tratar de alterar um método herdado de um pai, algumas vezes esta prática se faz necessária nos processos realizados pelo sistema. A linguagem de programação C# faz uso de métodos virtuais, dos quais serão reimplementados nos filhos como *override*.

Explicado os quatro pilares da POO, então será possível compreender que tipo de linguagem de programação será utilizada no desenvolvido do *software* File Manager, fazendo uso do ambiente de desenvolvimento (IDE) Visual Studio.

1.6.2. ASP.NET

Segundo a Microsoft (2016), ASP.NET é uma plataforma *web* que fornece os meios para o desenvolvimento de aplicativos de alto desempenho e qualidade para *web*. Tendo sua base no .NET Framework, está fornece todos os recursos do mesmo, além de fornecer suporte para desenvolvimento em C#, linguagem de programação proposta para desenvolvimento do aplicativo.

1.6.3. Web Forms

Segundo a Microsoft (2003), Web Forms são os elementos de interação com usuário, ou seja, as interfaces da aplicação *web*. Este se baseia na tecnologia ASP.NET Microsoft, dando suporte para diversas linguagens de programação, tal como: HTTP, HTML, XML, WML e ECMAScript (JScript, JavaScript).

1.7. Banco de Dados

Um banco de dados pode ser considerado um agrupamento de dados que fornecem informações sobre o mesmo assunto. Exemplo: lista telefônica.

1.7.1. Sistema Gerenciador de Banco de Dados

O SGBD escolhido para o *software* File Manager, é o MySQL, do qual utilizasse da linguagem SQL, que serve para manipular as informações contidas no banco de dados, ou seja, tem como função: inserir, acessar, modificar e excluir.

1.8. Ferramentas

1.8.1. brModelo 3.0

Para desenvolver o Diagrama Entidade Relacionamento será utilizada a ferramenta brModelo 3.0, este é comumente utilizado em instituições de ensino.

O brModelo 3.0 apresenta uma interface simples, porém com todas as ferramentas essenciais, ou seja, mesmo sendo simples é funcional. O interessante deste programa é a possibilidade de gerar outro modelo através de um lá elaborado, por exemplo, o usuário cria um modelo conceitual e partir deste gerar o modelo lógico, com todos os atributos e

relacionamentos presentes no conceitual. Caso venha a ter um relacionamento muitos para muitos à ferramenta cria uma entidade mediadora entre as duas, igual seria feito de forma manual.

1.8.2. Astah Community

Outra ferramenta que será utilizada no desenvolvimento do projeto será o Astah Community, está se apresenta como uma ferramenta de modelagem de UML, sendo desenvolvida no Japão na plataforma Java.

O Astah Community é de fácil manuseio para aqueles que estão começando no campo de desenvolvimento de *softwares*, entretanto, também possui ferramentas o suficiente para suportar sistemas complexos.

1.8.3. Visual Studio

> Visual Studio é um conjunto completo de ferramentas de desenvolvimento para construção de aplicações Web ASP.NET, serviços Web XML, aplicações desktop e aplicativos móveis. Visual Basic, Visual C# e Visual C++ usam todos o mesmo ambiente de desenvolvimento integrado (IDE), que permite o compartilhamento de ferramentas e facilita a criação de soluções com mistura de linguagens. Além disso, essas linguagens usam a funcionalidade do .NET Framework, que fornece acesso às tecnologias chaves que simplificam o desenvolvimento de aplicativos Web em ASP e serviços Web XML (MICROSOFT, 2010).

Está ferramenta de ambiente de desenvolvimento integrado foi escolhida devido aos conhecimentos já obtidos sobre a mesma, aprendidos no decorrer do curso. Além de esta fornecer meios necessários para o desenvolvimento do *software* File Manager.

1.8.4. MySQL Workbench

> MySQL Workbench é uma ferramenta visual unificada para arquitetos de banco de dados, desenvolvedores e DBAs. MySQL Workbench fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração abrangentes para configuração do servidor, administração de usuários, backup e muito mais. MySQL Workbench está disponível no Windows, Linux e Mac OS X (MySQL).

A escolha dessa ferramenta se deve a fácil adequação da mesma no projeto, isto devido a mesma ser compatível com o sistema gerenciador de banco de dados MySQL, e também por ter sido obtido conhecimentos sobre a mesma no decorrer do curso em questão.

2. ESTUDO DE CASO

Com o passar dos anos, as instituições se tornaram cada vez mais dependentes da Tecnologia da Informação. Devido a isto, seus sistemas passaram a ser mais complexos, com o propósito de atender as necessidades apresentadas no cotidiano. Contudo, o *software* utilizado pela empresa deve apresentar um bom custo-benefício.

As instituições, independentemente de seu tamanho, em grande parte das vezes têm excessivo acúmulo de documentações, em vista da necessidade crescente de manter seus registros. Devido a esse problema, ambientes que poderiam estar sendo utilizados para outros fins acabam por se tornar recintos para armazenagem de documentações.

Quando se adentra em uma instituição de pequeno ou médio porte, é possível identificar caixas espalhadas nas salas ou ambientes exclusivos para armazenagem de documentações, isso devido a necessidade de ter esse registro para futuras consultas. Empresas de grande porte tendem a apresentar um *Enterprise Resource Planning* (ERP), um *software* com diversos setores associado a ele, facilitando o fluxo de informações na empresa.

Os ERPs apresentam um módulo destinado à documentação, fazendo com que a armazenagem impressa dos documentos não seja necessária, visto que armazena em forma digital, sendo possível acessá-los a qualquer momento de qualquer dispositivo que disponha de acesso à internet, desde que a pessoa que tente acessar o sistema tenha um *login*[2] e uma senha.

O problema de desenvolver um novo *software* para uma instituição ou organização, é a questão da mudança que os usuários terão que passar, e devido a isso ocasiona uma resistência por parte dos mesmos.

O medo não fica limitado às pessoas que vão utilizar o sistema, mas a todas as que compõem a instituição. Entretanto, assumir riscos é uma característica do empreendedor, daqueles que têm por objetivo grandes conquistas, sonhos e metas.

Para tirar esse receio por parte dos usuários e das demais pessoas é necessário apresentar um *software* que tenha uma usabilidade e acessibilidade semelhante àquelas utilizadas no sistema que será substituído, mas que consiga resolver os problemas enfrentados.

[2] Identificação de acesso para um sistema.

Devido ao pensamento de propor o desenvolvimento de um sistema que atenda às necessidades e que não seja totalmente estranho para aqueles que irão utilizá-lo, foi realizada uma pesquisa no âmbito de encontrar-se algo que se assemelhasse ao que se propõe, sobretudo com o foco no trabalho realizado pela Polícia Científica.

Mesmo as grandes instituições mostrando a necessidade de utilizar um bom gerenciador de arquivos, algumas questões por parte dos entrevistados foram levantadas durante pesquisas realizadas para elaborar este trabalho, tais como: “Não é perigoso deixar registrado apenas em forma digital?”, “Não poderia vazar informações a respeito desses mesmos registros?”.

Ao realizar a pesquisa para o desenvolvimento deste trabalho, o sistema Gerenciador de Laudos (GDL) foi um dos que se assemelham ao *software* em desenvolvimento. Este é utilizado com o propósito de armazenar os laudos feitos pelos peritos, como também os equipamentos que foram periciados, porém apresenta ausência de funcionalidades necessárias para se adequar as atividades presentes no cotidiano do trabalhador que utiliza o sistema. Esse *software* foi desenvolvido exclusivamente para a polícia científica.

Durante a pesquisa foi constatado que não existem muitos sistemas destinados a esse propósito. Na maioria dos casos observados, estes utilizam um módulo do ERP para essa função.

Entretanto, fazer com que os usuários aceitem o novo *software*, mais completo, não é uma tarefa fácil, em vista da comodidade dos usuários. Sendo assim, apresentando uma explicação geral sobre como funcionaria o *software*, fazendo comparações com o já em uso, como será dito posteriormente, e como seria disposta essa informação para acesso, acabaram por entender que seria uma solução vantajosa para a instituição.

3. SOLUÇÃO PROPOSTA

O *software*, intitulado "File Manager" terá como função principal o armazenamento de arquivos de texto e imagem, sendo possível exercer as atividades nestes, tais como adição de imagens, dados e informações.

Cada documento de texto será único, ou seja, não podendo haver dois com o mesmo nome na base de dados da empresa. Isto devido ao mesmo ser associado ao número de registro, ou seja, o nome que o laudo precisa ter quando for salvo é o número do registro da instituição, exemplo: 290296_2016.

Quando na base de dados, o arquivo estará disposto para qualquer usuário visualizar como foi dito anteriormente, porém fazer alterações, tal como exclusão, edição e *upload*[3] de novos conteúdos, só será disposto para aquele que o postou e seu associado, no caso um estagiário, que terá um nível de acesso inferior ao de seu supervisor e com tempo determinado de acesso, que seria a duração do contrato de estágio ou quando o contrato for interrompido.

Através de um *login* e senha o usuário terá acesso aos arquivos da base de dados de qualquer dispositivo, somente sendo necessário um equipamento com acesso à internet. No caso dos estagiários, seu acesso ao site será limitado ao seu tempo de estágio no dia, ou seja, se um estagiário trabalhar das 13h00min até às 17h00min, então o mesmo só poderá acessar o site nesse tempo. Isso não é apresentado no sistema atualmente utilizado; o acesso é restrito a um endereço de rede, e o mesmo também não apresenta um sistema de acesso para estagiários.

O que se propõe para este sistema, também é o acesso ao *site* possa ser feito de fora da instituição, o que facilitará na hora dos chamados abertos ou ordens de serviço, pois, no caso da polícia científica, os plantonistas poderiam ver quais locais que necessitam de perícia e ir direto a eles, sem a necessidade de voltar para a instituição para verificar o que necessita ser feito e em qual localização que é como se faz atualmente. Além disso, as imagens do local poderão ser postadas com o laudo no *site*, de maneira que outros usuários possam visualizar o laudo e as imagens. O *software* utilizado não apresenta divisão das funcionalidades. É possível fazer apenas a inserção de um arquivo em formato de texto.

[3] Quando se envia um arquivo para algum lugar ou alguém por meio da internet.

Quando a polícia científica fizer uso do *software* File Manager, abre-se um campo para adicionar o grau de urgência do chamado, afim de agilizar a perícia no local do incidente.

Quando efetuada a perícia no local, em alguns casos, são apreendidos materiais para análises. Tendo isto em vista, abrem-se campos para registro de quais materiais foram apreendidos, quem são os indiciados, vítima, responsável pela perícia, lacre do saco plástico em que a peça veio, data de apreensão, data de elaboração de boletim de ocorrência ou denúncias, entre outros possíveis e necessários dados e informações.

Após apresentar as funcionalidades do *software* File Manager em comparação ao sistema Gerenciador de Laudos (GDL), percebe-se a carência de funcionalidades do segundo, fazendo-se necessário um sistema mais completo, como propõe-se para o File Manager, para atender a todas as atividades e funcionalidades que um funcionário que utiliza o sistema necessita.

Com esse foco, o *software* File Manager poderá ser desenvolvido para instituições policiais científicas, sendo necessário rodar na internet, por meio de um navegador, ou seja, será um *website*[4], que tem como função hospedar documentos, fazendo com que seja possível que outras pessoas tenham acesso aos arquivos de texto e imagem por meio de um usuário, senha e um acesso à internet. Através da autoridade que o usuário possuir, este poderá ter permissões para efetuar alterações no arquivo escolhido.

Para a análise, projeto, desenvolvimento e testes do sistema proposto, File Manager, sugere-se a aplicação das metodologias de Engenharia de Software, bem como as de Banco de Dados, tais como seguir o ciclo de vida em cascata, que, como pode-se observar na figura 1, determina que cada nova fase deverá ser desenvolvida após a anterior ter sido concluída, com o levantamento, análise e documentação dos requisitos, o desenvolvimento de diagramas UML adequados, a utilização de linguagens apropriadas, tanto para a manipulação de dados no Banco de Dados, quanto para a sua disponibilidade na Internet, entre outras.

Durante a definição de requisitos serão definidas as funções, restrições e objetivos que o sistema deve seguir e apresentar, após os mesmos serão organizados, priorizados, detalhados e documentados, fornecendo então uma especificação para o sistema.

[4] Conjunto de programas que rodam por meio de uma conexão via internet.

O projeto de sistema consistirá em agrupar os requisitos do sistema, tanto de *hardware* quanto de sistema, estabelecendo uma arquitetura geral do mesmo. Também serão identificadas e descritas as abstrações do sistema e suas relações.

No processo de implantação e teste de unidades, como o nome sugere, será verificado se as unidades correspondem às especificações necessárias.

Na fase da integração e teste de sistema, as unidades de programas ou programas individuais serão testados, a fim de garantir que os requisitos levantados anteriormente tenham sido atendidos. Após esse processo o *software* é entregue ao cliente.

A operação e manutenção do sistema é um processo realizado com o *software* já em execução. Caso existam erros que não foram identificados nas fases anteriores do ciclo de vida, a manutenção serve para corrigi-los, como também requisitos que não foram levantados anteriormente, podem ser implantados em versões futuras (SOMMERVILLE, 2003, p.38).

3.1. Levantamento de Requisitos para o File Manager

Com o levantamento de requisitos, esses são divididos em requisitos funcionais e requisitos não funcionais, como descrito anteriormente no capitulo 2, sendo assim estes fornecem as características e funcionalidades que o sistema deve possuir quando desenvolvido.

3.1.1. Levantamento dos Requisitos Funcionais

Conforme explicitado no capítulo 2, os requisitos funcionais correspondem às funcionalidades e serviços que o *software* deve apresentar para melhor entendimento sobre o que são requisitos funcionais.

- Cadastrar, editar e desativar contas de usuários.
- Será necessário um *login* e uma senha para adentrar no site e acessar suas funcionalidades.
- O sistema deverá permitir e controlar vários níveis de acesso, um para cada categoria de usuários.
- O usuário pode pesquisar os arquivos pelos seus respectivos registros.
- O Sistema deve oferecer telas separadas para cada funcionalidade que apresente, por exemplo, pesquisa, *upload* de arquivos e imagens, *download* de arquivos e imagens, etc.

- O usuário pode fazer *upload* de arquivos.
- O usuário pode fazer *download*[5] de arquivos.
- O usuário pode visualizar a imagem e o arquivo sem precisar fazer *download* dos mesmos.
- O *upload* de documentos só aceitará arquivos em formato de *.pdf*.
- A edição do documento não poderá ser feita de forma *online*.
- O arquivo somente poderá ser editado e excluído por quem fez *upload* e por alguém com nível de acesso superior.
- Após dada baixa no registro, é possível realizar apenas a visualização e *download* do documento e imagens referente ao mesmo.
- Deve haver um registro de logs, para criar um maior controle sobre as operações realizadas e trilhas de auditoria.

3.1.2. Levantamento dos Requisitos Não Funcionais

Também conforme o capítulo 2, os requisitos não funcionais vão definir propriedades e restrições do sistema.

- O *software* deverá ser executável em qualquer plataforma operacional (MS-Windows, Linux, Android, iOS…).
- É necessário estar *online*, para que possa ser acessível de qualquer local por meio de uma rede de internet.
- A interface deverá ser simples e enxuta, porém com uso de ícones que evidenciem claramente a operação a ser realizada.
- Os processos de *upload* e *download* deverão apresentar opções facilitadas.

3.1.3. Modelo Conceitual e Modelo Lógico

Após o levantamento dos requisitos é possível passar para a projeção do Modelo Entidade Relacionamento e Diagrama Entidade Relacionamento.

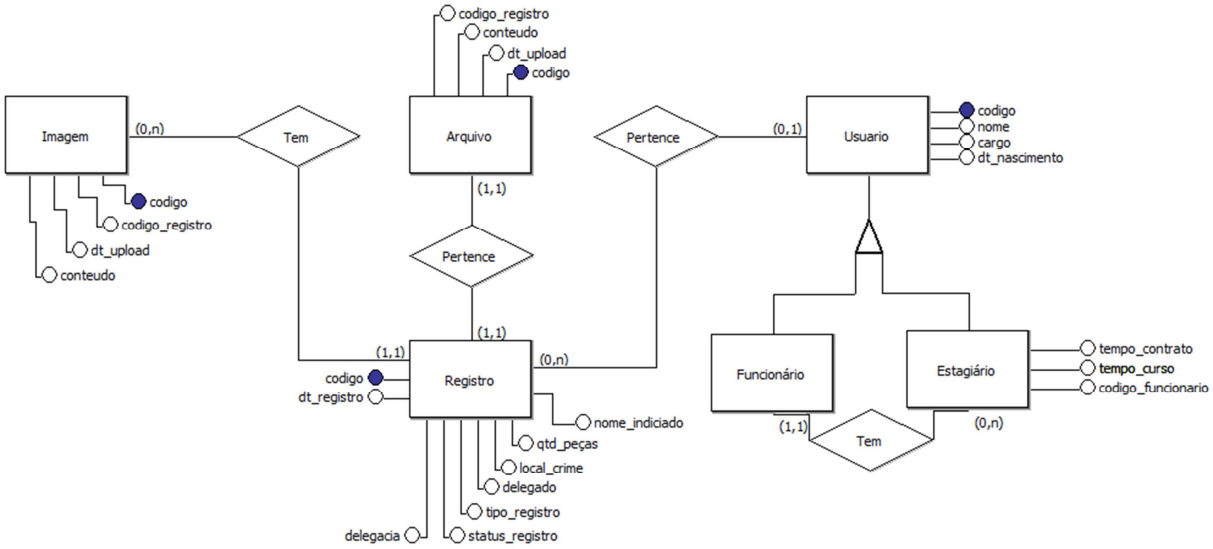
Como o próprio nome sugere, o Modelo Entidade-Relacionamento (MER) trata-se de uma representação abstrata do banco de dados, fornecendo a base para o desenvolvimento do diagrama entidade relacionamento, apresentando as entidades e os relacionamentos entre elas.

[5] Quando se faz cópia de um documento de outro equipamento.

O Diagrama Entidade Relacionamento (DER) será a representação gráfica do Modelo Entidade Relacionamento. Quando se desenvolve um sistema, algumas vezes a escolha de gerar uma representação gráfica do que será desenvolvido é mais plausível do que apenas explicar o que será feito, e como será feito. A fim de facilitar a comunicação entre os membros da equipe ou mesmo com o cliente, desenvolve-se o modelo conceitual e lógico do sistema.

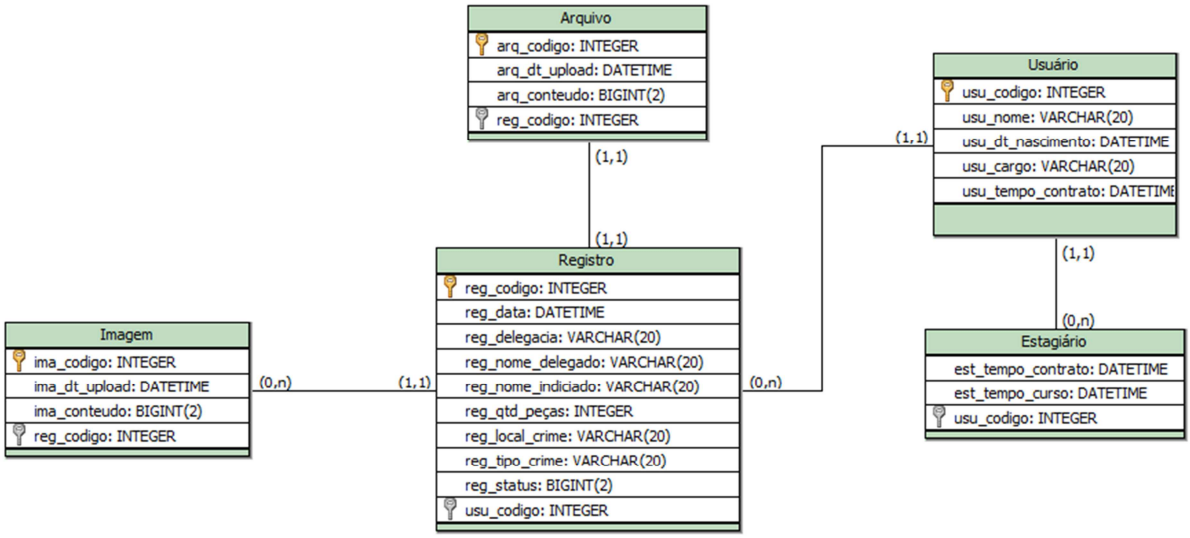
No projeto do *software* File Manager foram feitos dois diagramas que seriam o modelo conceitual e o modelo lógico. O primeiro modelo tem como foco envolver o cliente e mostrar como será o funcionamento do sistema de maneira visual. No modelo lógico serão definidos padrões e nomenclaturas, e é criado a partir do modelo conceitual, tendo em vista o desenvolvimento do *software*, e não interação com cliente.

Figura 2 – Modelo Conceitual.



Fonte: Próprio Autor.

Figura 3 – Modelo Lógico.



Fonte: Próprio Autor.

Na Figura 2 e Figura 3 é possível observar as entidades e seus atributos, a serem criados para o *software* File Manager, e é através desses modelos que o banco de dados e todo o resto será desenvolvido.

3.1.4. Modelo Físico

Com base nos modelos apresentados é possível desenvolver o banco de dados, ou pelo menos a estrutura do mesmo. Como em qualquer *software* estes modelos estão sujeitos à mudança conforme a necessidade apresentada no desenvolvido do sistema.

A seguir, exemplificação da criação de tabelas com SQL (Structured Query Language, aceita por virtualmente quaisquer Sistemas Gerenciadores de Banco de Dados) que deverão conter os dados no File Manager:

```
CREATE TABLE Usuario (
usu_codigo INTEGER,
usu_nome VARCHAR(20),
usu_dt_nascimento DATETIME,
usu_cargo VARCHAR(20),
usu_tempo_contrato DATETIME,
tempo_curso DATATIME,
constraint PK_Usuario PRIMARY KEY (usu_codigo)
)

CREATE TABLE Registro (
reg_codigo INTEGER,
usu_codigo INTEGER,
reg_data DATETIME,
```

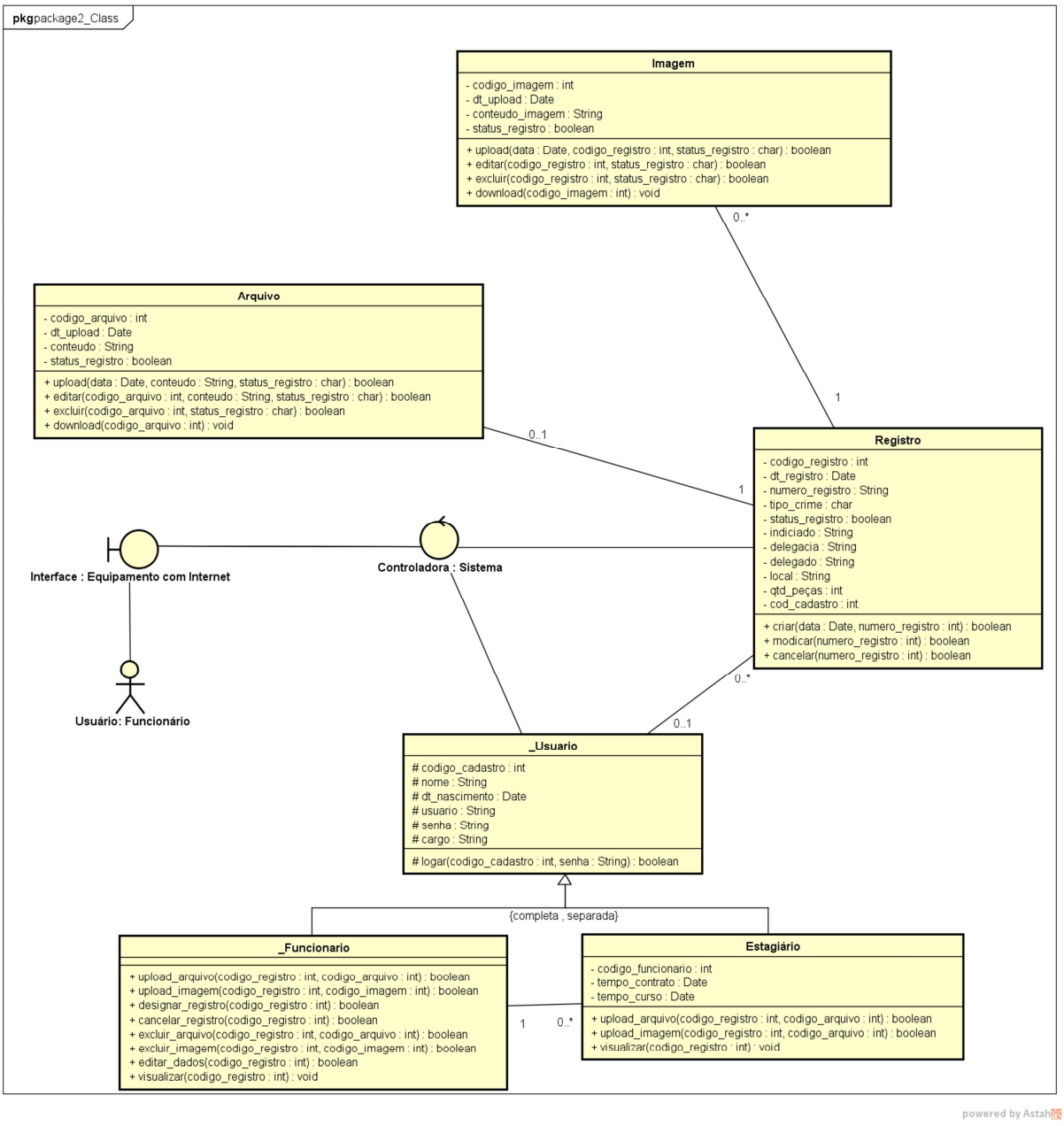
```
reg_delegacia VARCHAR(20),
reg_nome_delegado VARCHAR(20),
reg_nome_indiciado VARCHAR(20),
reg_qtd_peças INTEGER,
reg_local_crime VARCHAR(20),
reg_tipo_crime VARCHAR(20),
reg_status BIGINT(2),
constraint PK_Registro PRIMARY KEY (reg_codigo),
constraint FK_Usuario FOREIGN KEY (usu_codigo)
        REFERENCES Usuario (usu_codigo)
        on update cascade
)

CREATE TABLE Arquivo (
arq_codigo INTEGER,
arq_dt_upload DATETIME,
arq_conteudo BIGINT(2),
reg_codigo INTEGER,
constraint PK_Arquivo PRIMARY KEY (arq_codigo),
constraint FK_Arq_Registro FOREIGN KEY (reg_codigo)
        REFERENCES Registro (reg_codigo)
        on update cascade on delete cascade
)


CREATE TABLE Imagem (
ima_codigo INTEGER PRIMARY KEY,
ima_dt_upload DATETIME,
ima_conteudo BIGINT(2),
reg_codigo INTEGER,
constraint PK_Imagem PRIMARY KEY (ima_codigo),
constraint FK_Ima_Registro FOREIGN KEY (reg_codigo)
        REFERENCES Registro (reg_codigo)
        on update cascade on delete cascade
)
```

3.2. Diagramas de Classe, Caso de Uso e de Sequência

Apresentado o modelo conceitual, modelo lógico e modelo físico, é possível desenvolver os diagramas do sistema, ou seja, representar como as funcionalidades do sistema vão ser realizadas.

Para a criação dos diagramas de UML foi utilizado a ferramenta case Astah Community 7.0.0, os diagramas vão auxiliar no desenvolvimento do sistema.

Figura 4 – Diagrama de Classe do software proposto.



Fonte: Próprio Autor.

Como já explicado no capitulo 2, o diagrama de classe vai descrever os tipos de objetos que serão encontrados no sistema e como se dará o relacionamento entre eles. Nesse *software* os objetos encontrados são: usuário (funcionário, estagiário), registro, arquivo e imagem.

A interação entre os objetos se dá de maneira simples, facilitando na leitura do mesmo. O diagrama em questão apresenta a classe usuário, sendo a mãe de duas filhas: funcionário e estagiário. Ambas apresentam suas próprias características, entretanto herdam da mãe os atributos: codigo_cadastro, nome, dt_nascimento (data de nascimento) e usu_senha

(usuário senha), ou seja, ambas possuem essas características além de suas próprias. Além dos atributos, estas vão herdar o método "*logar*".

Como já explicado em capítulos passados, os atributos serão características que os objetos vão possuir, e os métodos são ações que exercem no sistema.

O objeto Usuário está associado diretamente à controladora do sistema, isto faz com que qualquer tentativa de acesso no sistema, exerça a função de busca no objeto em questão. A outra ligação realizada por esta, será com o objeto Registro, tendo então uma ligação de 1 para 0..*, ou seja, o usuário poderá ter nenhum a muitos registros associados a ele, enquanto que o objeto Registro estará associado a somente um objeto Usuário, o funcionário.

O objeto Registro estará associado aos outros objetos, porém, só poderá ser associado a um único usuário, bem como associado a um único arquivo também. Será possível fazer *upload* de diversas imagens, associando-as a um mesmo registro. O objeto Arquivo associa-se ao objeto Registro, assim como ao objeto Imagem.

Com o diagrama de classe feito e explicado, podem ser elaborados e explicados os diagramas de caso de uso e de sequência, conforme descritos no Capítulo 2. Apenas reforçando, o primeiro modelo de diagrama apresentará as funcionalidades e interações que ocorrerão no sistema, tendo o cliente como o foco.

O segundo modelo de diagrama apresentará as funcionalidades de modo sequencial, fornecendo assim conhecimento sobre as ações e suas respectivas consequências.

O primeiro diagrama de sequência que será apresentado após a elaboração do diagrama de classe, ao menos nesta proposta de projeto, será em relação ao *login* no sistema, isto devido a tudo começar com o acesso do usuário no sistema, conforme "ANEXO A".

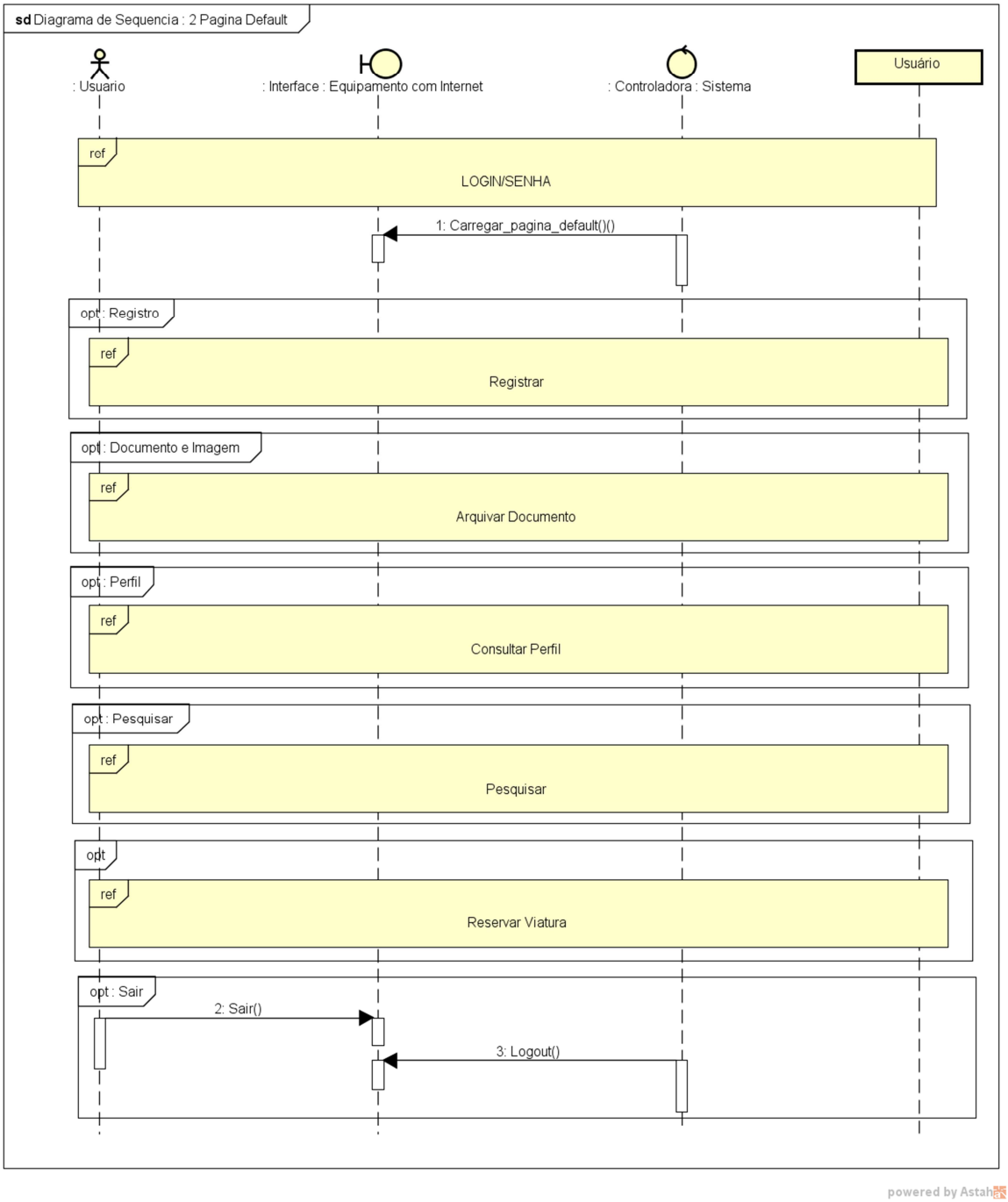
O processo para acesso ao *software* se dá de maneira simples. Quando acessado o endereço de rede do mesmo, este solicita o *login* e a senha do usuário. O primeiro diz respeito ao código de cadastro, atributo observado no objeto usuário do diagrama de classe, Figura 4, e a senha correspondente a chave de acesso, atribuída no momento do cadastro, podendo ser alterada após o cadastramento. Portanto, os requisitos básicos para entrar no sistema seriam: acesso à internet, um *login* e por fim uma senha.

Depois de preenchidos os campos, é realizada a validação dos dados fornecidos. Caso o *login* e a senha estejam corretos, apresenta-se uma mensagem de sucesso e a página padrão do sistema. Caso um dos dados fornecidos não esteja correto, apresenta-se uma

mensagem com o erro pertinente e retorna a solicitar o *login* e senha, reiniciando-se todo o processo, até que haja sucesso.

Como pode ser observado, na pesquisa não foram abordadas questões de segurança, além destas, e demais ações preventivas, isto devido ao foco do projeto não ser esta e sim a de apresentar uma proposta de projeto para a solução dos problemas em relação ao acúmulo de documentações impressas nos recintos das instituições técnico-científicas.

Figura 5 – Diagrama de Sequência referente à página inicial do sistema após login.

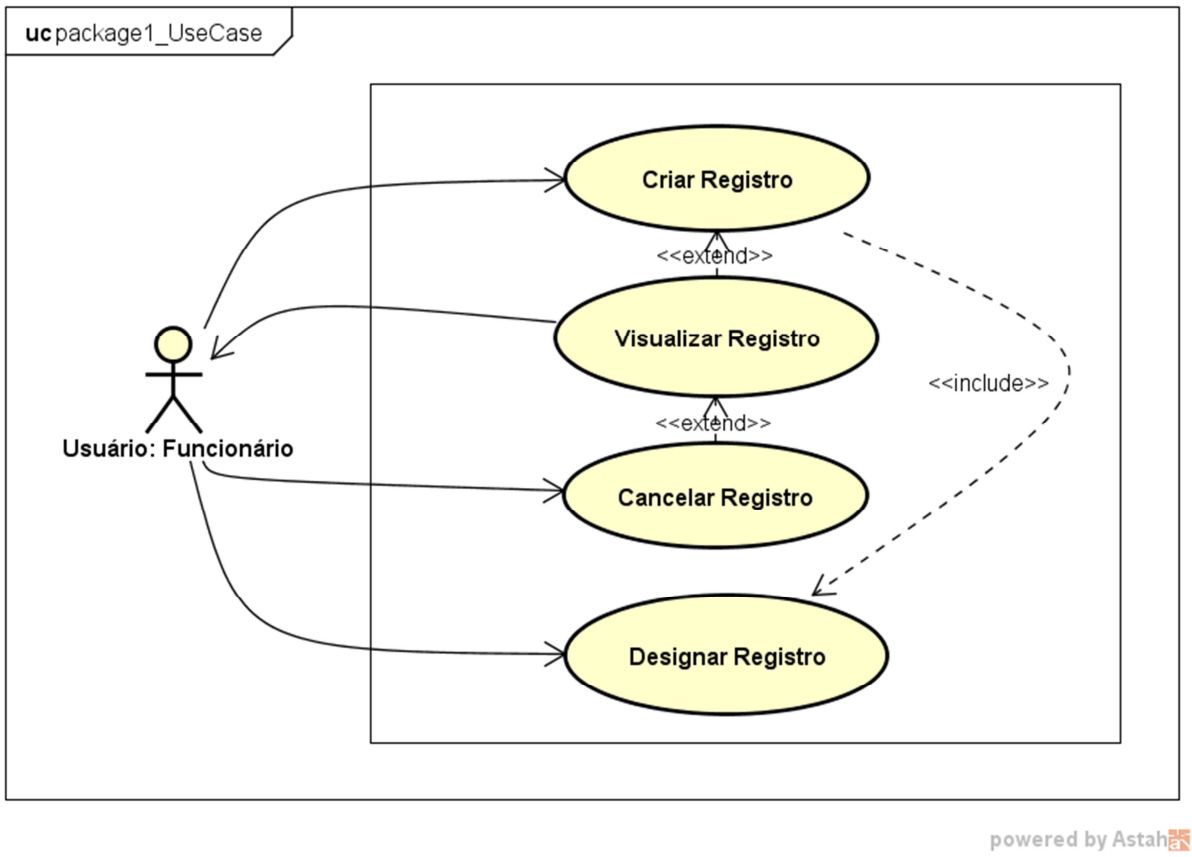


Fonte: Próprio Autor.

A página default do sistema vai apresentar as ações que o usuário poderá vir a executar durante sua permanência no sistema, sendo então possível: registrar, arquivar documento, consultar perfil, pesquisar e reservar viatura.

As atividades exercidas dentro de cada ação referenciada no sistema serão explicadas futuramente nos momentos oportunos no decorrer do trabalho.

Figura 6 – Diagrama do Caso de Uso do registro do número de controle interno.



Fonte: Próprio Autor.

O diagrama caso de uso referente ao registro mostra as funcionalidades e casos de usos que virão a ser exercidos no *software* File Manager. O Usuário: Funcionário criará um registro e o designará a um serviço, sendo que esse serviço será onde o usuário que fizer o serviço fará *upload* do arquivo de texto, este sendo o laudo do ocorrido. Algumas vezes um serviço pode vir a ser registrado duas vezes com números diferentes, caso isso ocorra o funcionário pode cancelar um desses registros. Futuramente caso venha a ser necessário para consultas futuras, o usuário poderá acessar o registro do serviço e fazer *download* do laudo.

Tabela 1 – Documentação do Caso de Uso referente ao Registro de Controle.

Nome do Caso de Uso	Registro
Ator Principal	Usuário: Funcionário
Atores Secundários	-
Resumo	Este estudo de caso mostra as funcionalidades que podem ser exercidas sobre um registro, ou seja, criação, edição e cancelamento.
Ações do Ator	Ações do Sistema
1. Solicitar criação de registro.	
	2. Solicitar dados para preenchimento dos campos.
3. Informar os dados para criação do registro: tipo de crime, local, delegacia, delegado, quantidade de peças, etc.	
	4. Validar campos.
	5. Registrar dados no banco e dados.
	6. Solicitar designação de perito.
7. Informar Perito	
	8. Atribuir registro ao perito no banco de dados.
	9. Criar número de registro.
	10. Exibir mensagem do resultado da operação.
	11. Imprimir número de registro.
12. Pesquisar registro	
	13. Pesquisar registros.
	14. Exibir resultado.
15. Solicitar edição dos dados de registro.	
	16. Solicitar dados.
17. Editar dados ou preencher campos ainda não informados.	
	18. Validar dados editados ou preenchidos.
	19. Registrar no banco de dados.
	20. Exibir mensagem do resultado da

	operação.
21. Pesquisar registro.	
	22. Buscar no banco de dados o registro.
23. Solicitar cancelamento de registro.	
	24. Verificar existência de arquivos ou imagens associados ao número de registro.
	25. Exibir mensagem com o resultado da operação.
Restrições/Validações	1. O cancelamento ou edição do conteúdo associado ao registro, só é possível pela pessoa a qual o registro está designado.
	2. Os campos: Tipo de crime, delegacia, delegado, quantidade de peças e status do registro devem ser apresentadas como lista para o usuário.

Fonte: Próprio Autor.

O diagrama referente à criação de um novo registro se dá a partir do carregamento dos campos padrões, especificados no diagrama de classe, ou seja, as informações que podem ser pertinentes ao crime. Conforme Anexo B.

Os campos carregados serão preenchidos com os dados contidos em requisição, portanto, poderá ser informado: tipo de crime, indiciado, data de chegada da requisição, delegacia a qual deve ser encaminhado o laudo, nome do delegado e perito responsável.

Importante ressaltar que não é obrigatório o preenchimento dos campos, isto devido, a como foi dito anteriormente, os dados serem retirados da requisição, sendo possível que a mesma não apresente todos os dados do crime investigado.

O registro é criado pelo sistema de forma aleatória, por exemplo, será apresentado o registro dessa maneira: 652_2016. Sendo o número inicial 652 gerado de forma sequencial, ou seja, o último número de registro acrescentado mais um seguido do ano de registro.

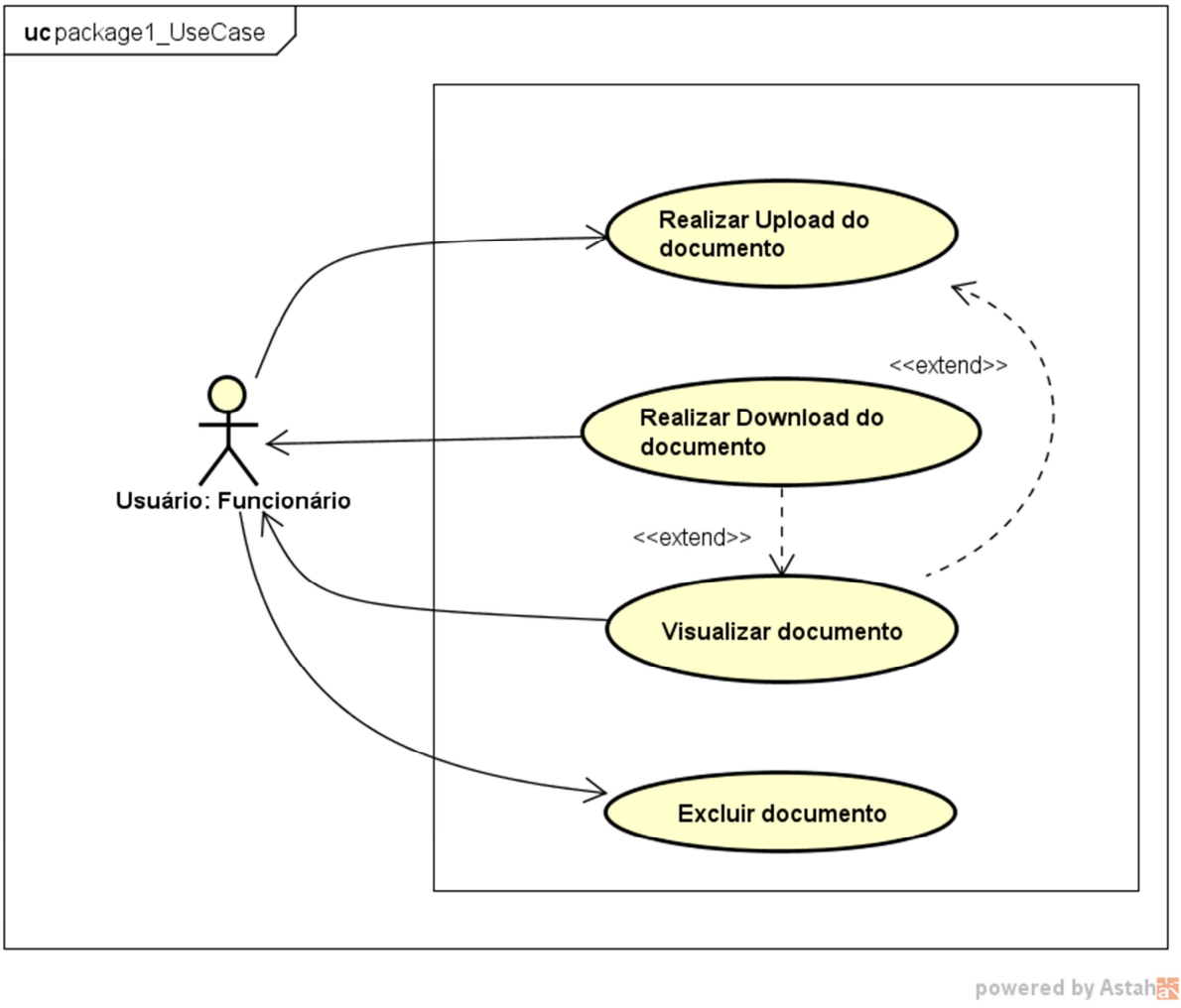
O diagrama explicativo sobre a edição dos dados do registro pode ser visualizado no Anexo C, este vai apresentar as atividades realizadas na edição dos dados de um determinado registro. Primeiramente é necessário existir uma busca pelo registro, conforme Anexo J.

Então é possível preencher os campos em branco ou editar aqueles já preenchidos, quando finalizada a ação ocorre uma validação dos dados, caso estejam todos corretos é

realizado o registro no banco de dados e impressa uma mensagem na tela informando o sucesso da ação, caso porventura não tenha sido preenchido corretamente as informações, é impressa uma mensagem com os erros encontrados.

Assim como o processo explicado anteriormente, o cancelamento do registro começa a partir de uma busca realizada por um registro, quando o mesmo é encontrado, é solicitado então o cancelamento do mesmo, entretanto, caso existam documentos ou imagens associadas a este, não é possível realizar a exclusão do mesmo, caso não existam todos os dados referentes ao registro são excluídos. Conforme Anexo D.

Figura 7 – Diagrama do Caso de Uso: Documento.



Fonte: Próprio Autor.

O diagrama de caso de uso referente ao documento apresenta o que o usuário poderá fazer em cima do mesmo, como mostra a Figura 7 – Diagrama do Caso de Uso: Documento, o usuário poderá fazer *upload* do arquivo desde que o mesmo esteja associado a um registro já

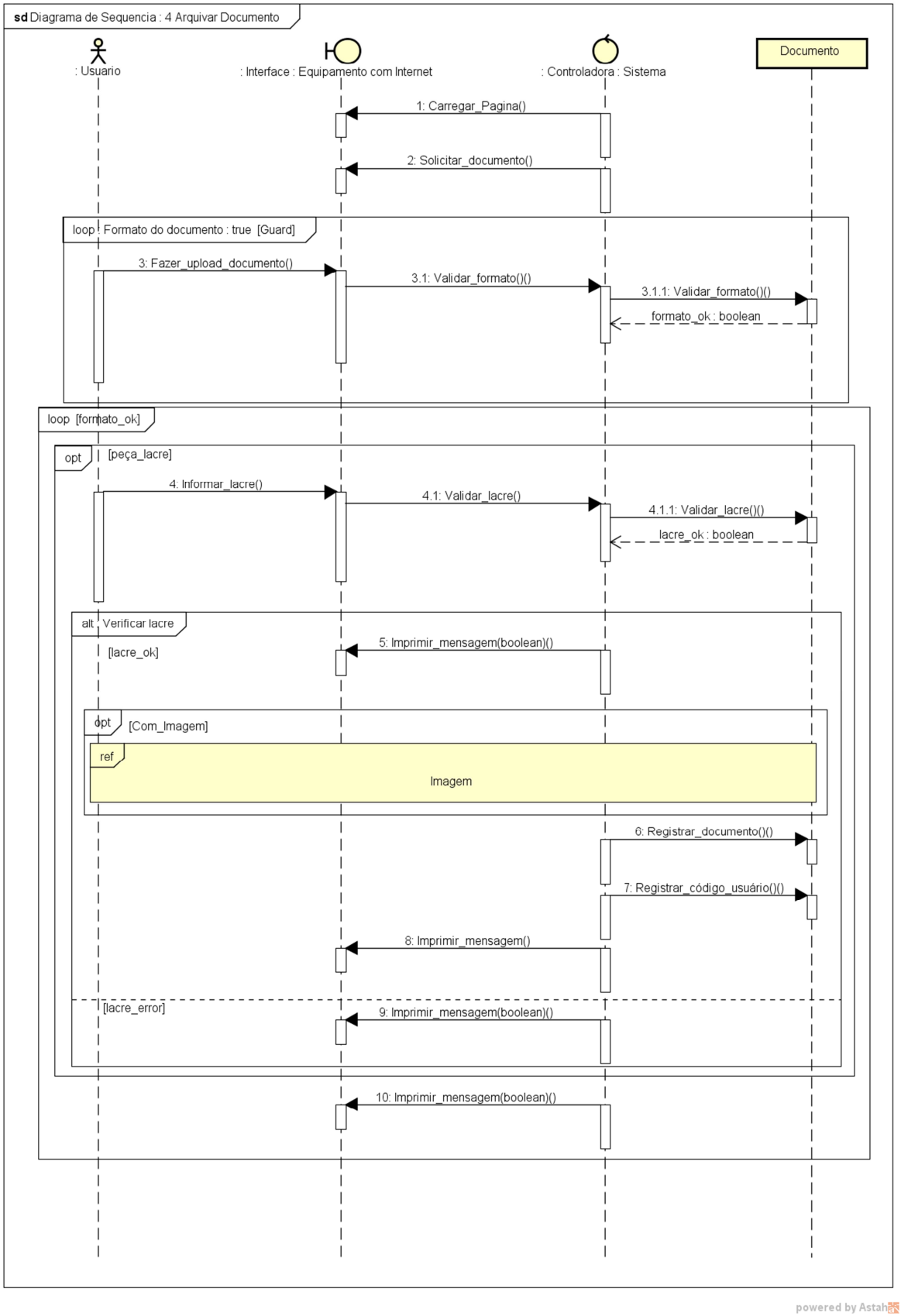
criado, após o *upload* o usuário terá a opção de visualizar o arquivo de forma online. Além de poder excluir o documento do qual fez *download*, importante ressaltar que tal atividade só pode ser praticada pelo usuário que fez o *upload* ou por alguém com maior acesso, no caso deste sistema, o perito chefe.

Tabela 2 – Documentação do Caso de Uso do documento.

Nome do Caso de Uso	Documento
Ator Principal	Usuário
Atores Secundários	-
Resumo	Este estudo de caso descreve as etapas para cadastrar, visualizar e fazer *download* do arquivo.
Ações do Ator	Ações do Sistema
	1. Solicitar o documento.
2. Fazer *upload* do documento.	
	3. Validar o formato do documento.
	4. Registrar no banco de dados o documento.
	5. Exibir mensagem do resultado da operação.
6. Fazer pesquisa do registro.	
	7. Buscar número de registro na base de dados.
8. Solicitar *download* do documento.	
	9. Efetuar *download* do arquivo.
	10. Exibir mensagem do resultado da operação.
11. Solicitar visualização do documento.	
	12. Disponibilizar documento *online*.
13. Solicitar exclusão do documento.	
	14. Analisado código de registro do usuário.
	15. Exclusão do documento.
	16. Exibir mensagem do resultado da operação.
Restrições/Validações	1. O documento que for feito *upload*, deve ser em formato pdf.

Fonte: Próprio Autor.

Figura 8 – Diagrama de Sequência referente ao *upload* do documento.



Fonte: Próprio Autor.

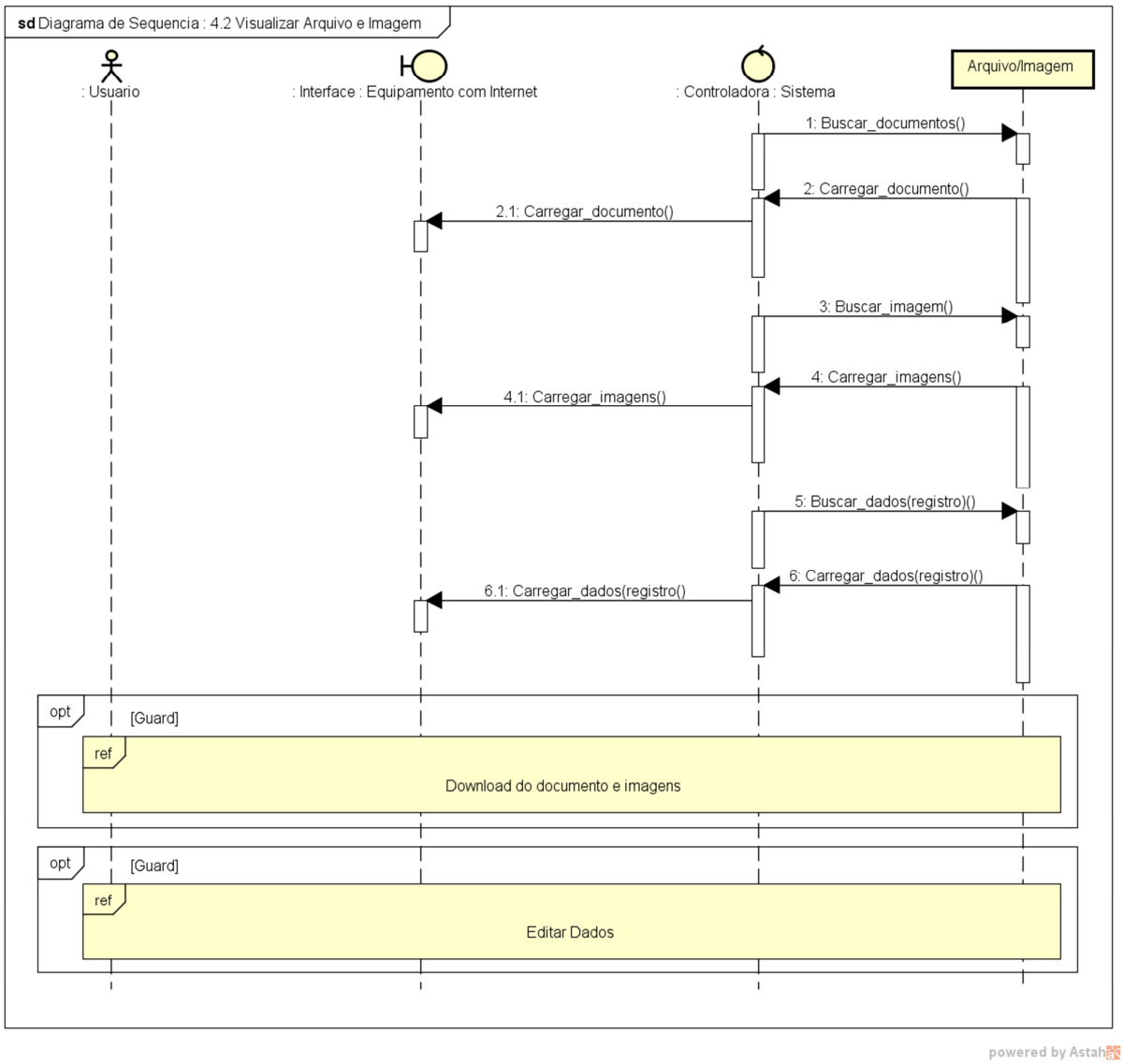
Quando o usuário quer fazer *upload* de um documento à página para tal ação é carregada, conforme “ANEXO K”. Solicitando assim a inserção do documento, onde o mesmo vai ser validado, ou seja, será analisado o formato do arquivo e se está de acordo com o que o sistema exige.

Caso o formato esteja correto, é liberada a página para prosseguir com a inserção de dados, por exemplo, a inserção de um lacre. O lacre diz respeito a um número elaborado pelo núcleo de polícia do Estado de São Paulo, onde este se apresenta com oito dígitos, além de único. A numeração do lacre, devido ao mesmo ser desenvolvido externamente, só apresenta restrição quanto a ser numérico e possuir oito dígitos, a combinação desses numerais não é validada.

Quando inserido algo divergente de números no campo do lacre ou não possuir os oito dígitos, é impressa uma mensagem na tela informando o erro que está impedindo o prosseguimento.

Feito isso é dada a opção de inserção de imagens, caso venha a querer realizar tal ação é encaminhado à outra tela, onde diz respeito à imagem, como será explicado no diagrama seguinte.

Figura 9 – Diagrama de Sequência referente à visualização do documento, imagem e dados do registro.

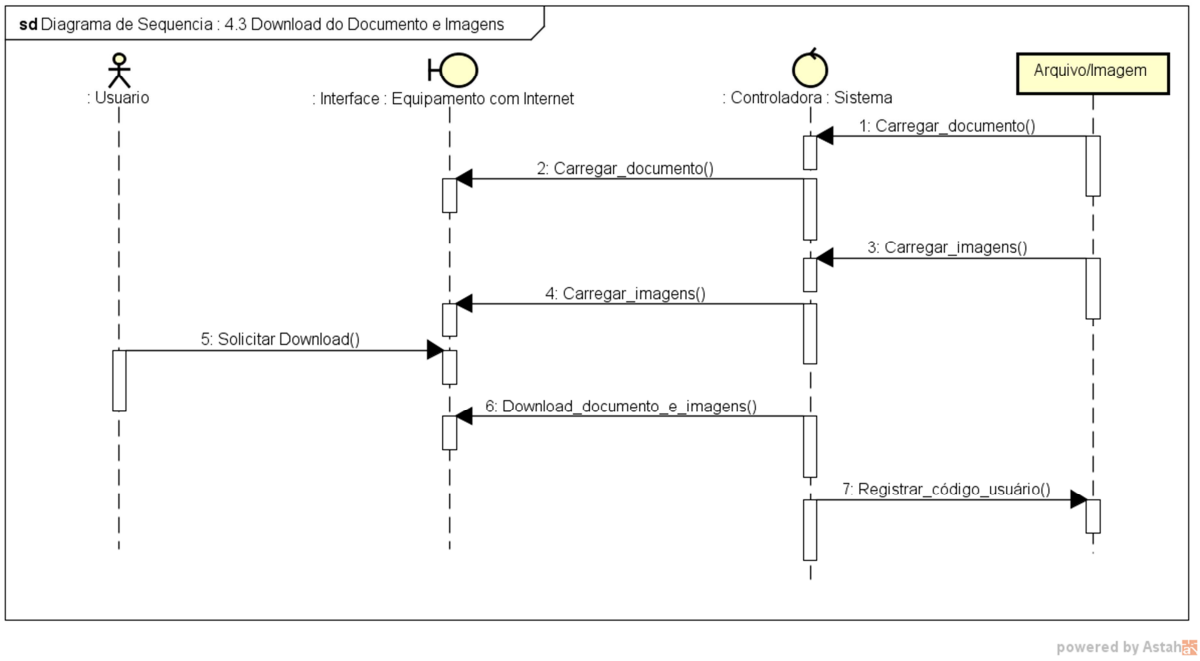


Fonte: Próprio Autor.

Caso o usuário deseje somente visualizar o documento e as imagens, estes se apresentam de forma *online*, ou seja, sendo possível visualizar sem realizar o *download* dos mesmos. Apesar de fornecer o conteúdo de forma *online* a página deixa as opções de *download* dos documentos e imagens, e edição dos dados do registro. Como pode ser visualizado no diagrama a seguir.

Em relação à referência ao diagrama de edição de dados do registro, já foi explicado anteriormente.

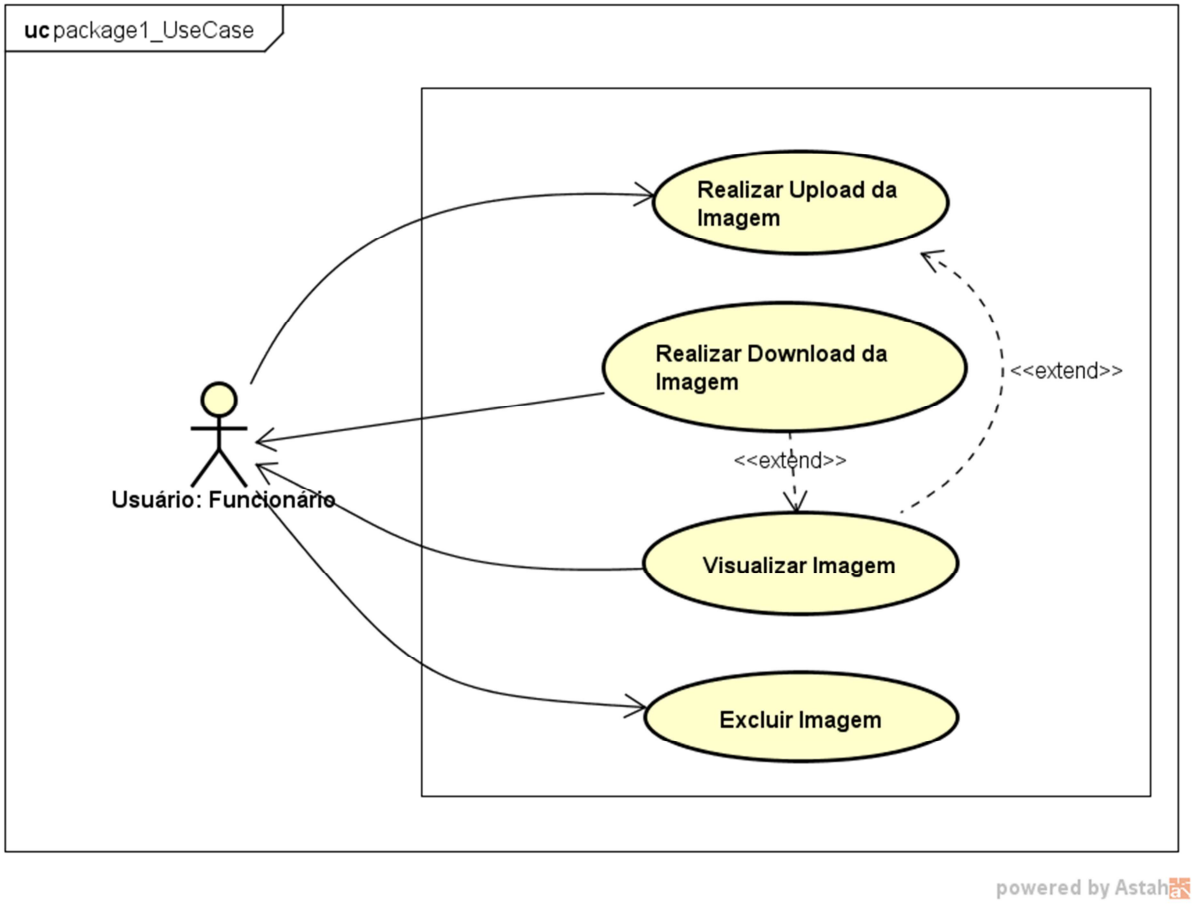
Figura 10 – Diagrama de Sequência referente ao *download* do documento e imagem.



Fonte: Próprio Autor.

Neste diagrama é apresentada a funcionalidade de *download* de documento e imagens referentes ao registro, como pode ser observado são carregados e exibidos os documentos e as imagens, o usuário solicita o *download* dos arquivos e este é realizado, por mim existe um registro do código do usuário para fins de controle.

Figura 11 – Diagrama Caso de Uso referente à imagem.



Fonte: Próprio Autor.

O diagrama caso de uso referente à imagem consiste em apresentar as funcionalidades e casos de uso existentes, como também a relação entre eles. O Usuário: Funcionário fará *upload* de imagens no registro em que o serviço foi associado, sendo possível que o mesmo seja visualizado de forma *online*, ou mesmo consultado futuramente. Além de possibilitar a exclusão da imagem, caso venha a ser necessário.

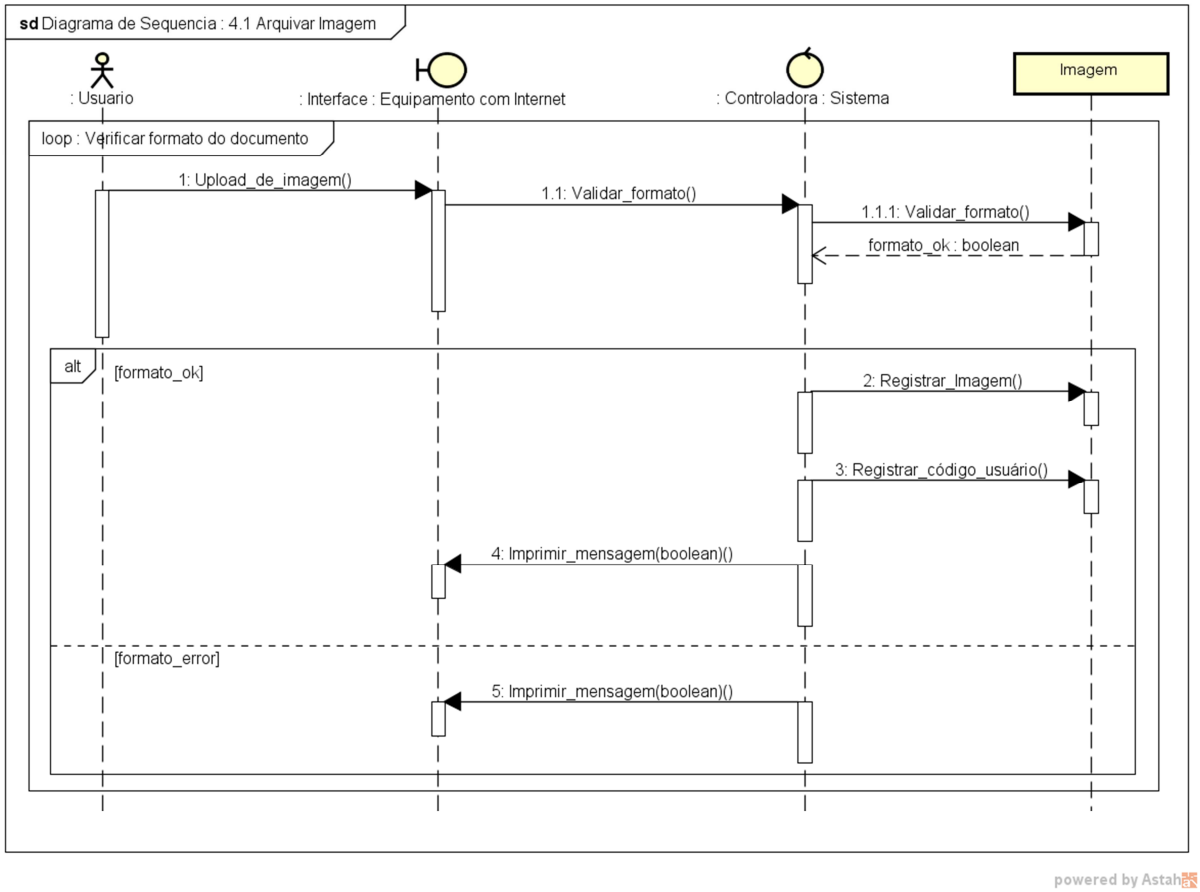
Tabela 3 – Documentação do Caso de Uso da imagem.

Nome do Caso de Uso	Imagem
Ator Principal	Usuário
Atores Secundários	-
Resumo	Este estudo de caso descreve as etapas para cadastrar, visualizar e fazer *download* e excluir a imagem.
Ações do Ator	Ações do Sistema

	1. Solicitar *upload* de imagem.
1. Fazer *upload* da imagem.	
	2. Verificar a formato da imagem.
	3. Salvar no banco de dados a imagem.
	4. Exibir mensagem do resultado da operação.
5. Visualizar imagem.	
	6. Procurar imagem na base de dados.
	7. Exibir o resultado.
8. Fazer *download* da imagem.	
	9. Procurar na base de dados a imagem.
	10. Fazer *download* da imagem.
Restrições/Validações	1. A imagem deve estar em formato *.jpg*.

Fonte: Próprio Autor.

Figura 12 – Diagrama de Sequência referente ao *upload* de imagens.

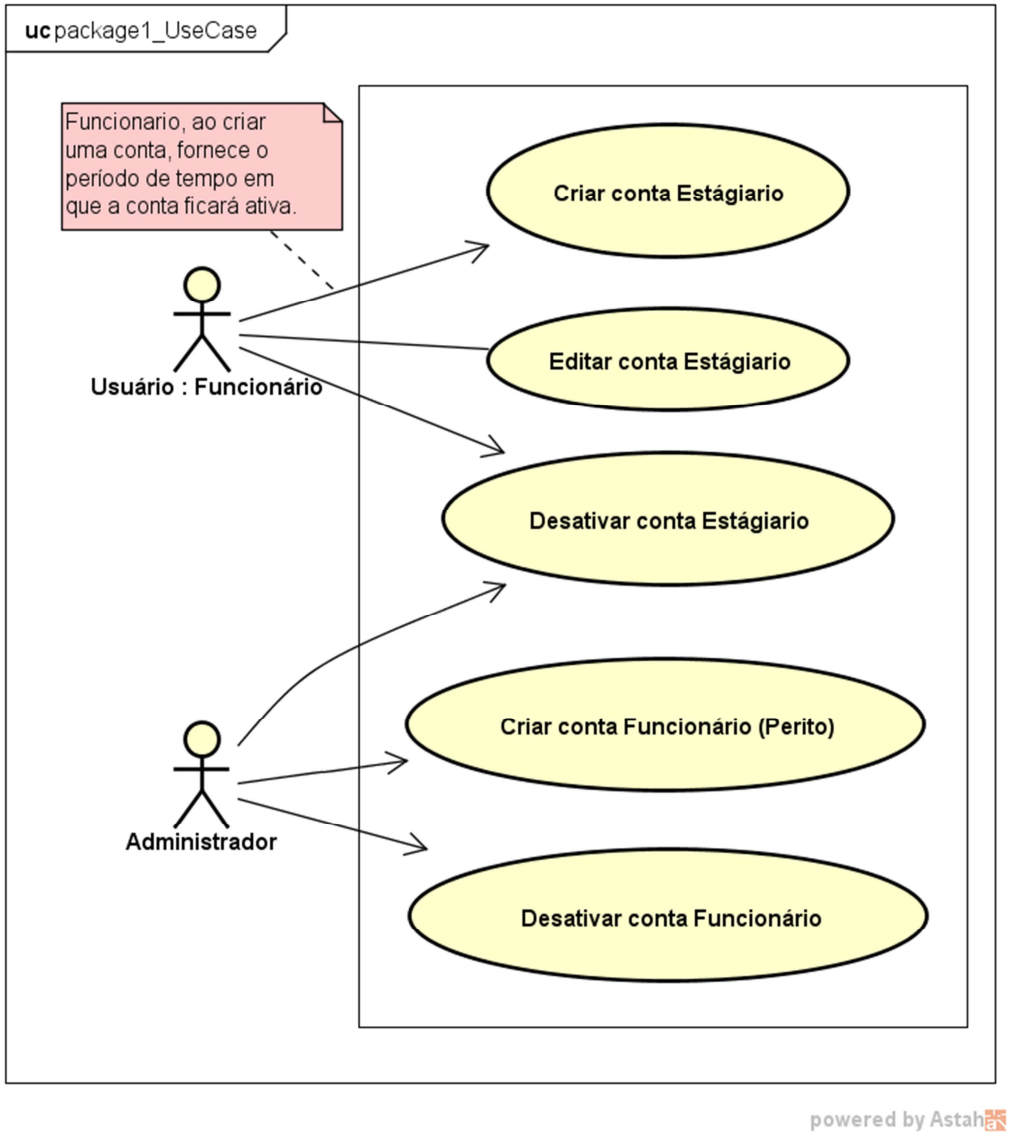


Fonte: Próprio Autor.

O processo de *upload* de imagens não se difere muito do que já foi apresentado anteriormente quando explicado sobre o arquivamento de documentos. Inserida a imagem é feita a sua validação da mesma e então registrada no banco de dados, exibindo na tela uma mensagem sobre o resultado da operação, para que o usuário tenha ciência do ocorrido. Caso porventura não seja compatível o formato da imagem, é exibida uma mensagem sobre o erro.

Os diagramas referentes ao processo realizado no *download* e visualização de imagens fora explicado anteriormente, quando explicado sobre os documentos, para mais informações consultar Figuras 9 e 10.

Figura 13 – Diagrama de Caso de Uso referente às contas do funcionário e estagiário.



Fonte: Próprio Autor.

O diagrama caso de uso referente às contas criadas tem como objetivo apresentar os atores, casos de uso, funcionalidades e relação exercida entre eles. Nesse diagrama são apresentados três atores, sendo o Usuário: Funcionário, Administrador e Banco de Dados.

O Usuário: Funcionário terá permissão apenas de criar conta de estagiário, isso é, uma conta para seu funcionário, com acesso limitado, que está associada ao tempo do contrato. O Administrador será aquele com autoridade para criar conta funcionário, como também possuirá permissão para excluir ou desativar a mesma.

Tabela 4 – Documentação do Caso de Uso de contas do funcionário e estagiário.

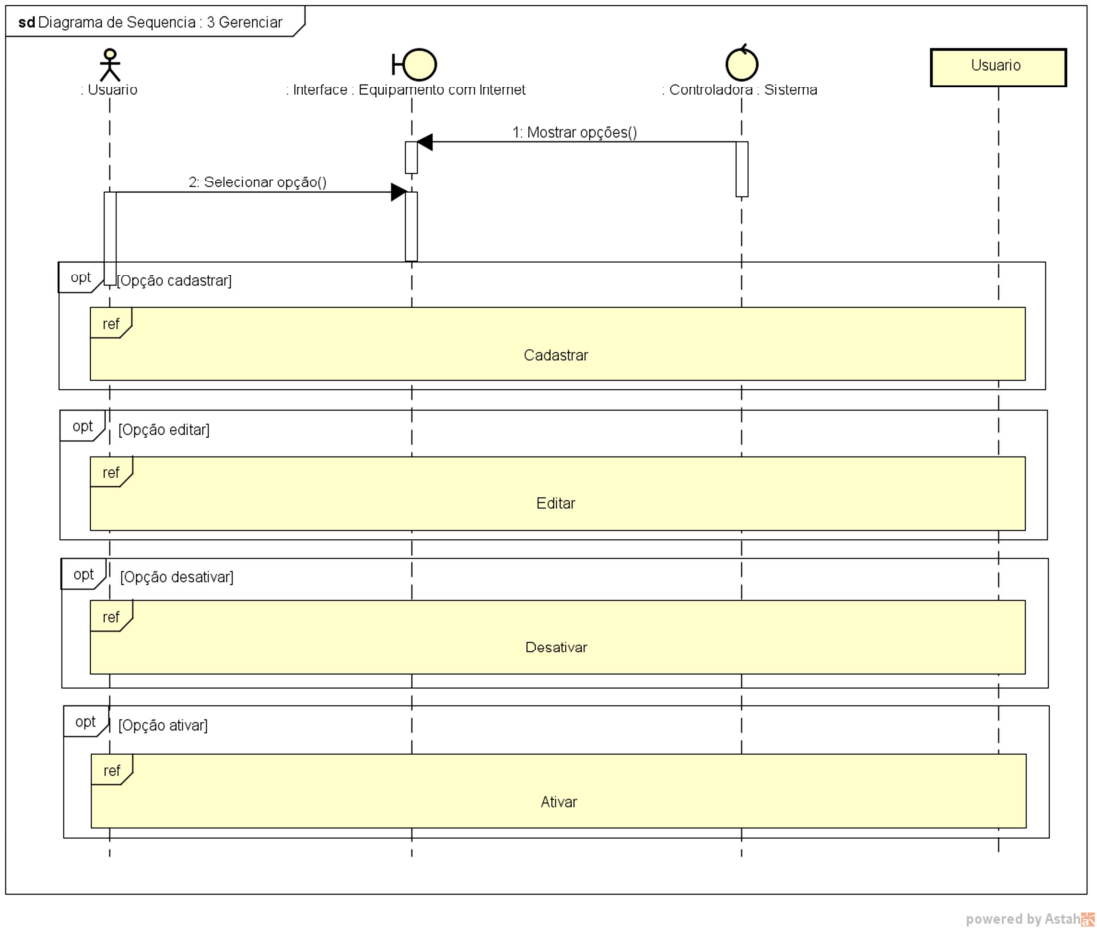
Nome do Caso de Uso	Contas
Atores Principais	Administrador e Usuário: Funcionário
Ator Secundário	-
Resumo	Este estudo de caso descreve as etapas para criar, editar e desativar contas.
Ações do Ator Funcionário	Ações do Sistema
	1. Solicitar dados a serem cadastrados.
2. Fornecer dados para a criação da conta: Nome, a data de nascimento, cargo, tempo de contrato e tempo de curso.	
	3. Validar dados preenchidos.
	4. Registrar dados.
	5. Criar conta.
	6. Exibir mensagem do resultado da operação.
7. Editar conta.	
	8. Carregar lista de estagiários.
9. Selecionar estagiário.	
	10. Carregar campos.
11. Editar ou preencher campos.	
	12. Validar campos.
	13. Registrar campos.
	14. Exibir mensagem do resultado da operação.
15. Desativar conta.	
	16. Carregar lista de estagiários.
17. Selecionar estagiário.	

	18. Desativar conta.
	19. Exibir mensagem do resultado da operação.
Ações do Ator Administrador.	Ações do Sistema.
20. Desativar conta funcionário.	
	21. Carregar lista de funcionários.
	22. Exibir lista de funcionários.
23. Selecionar funcionário.	
	24. Desativar conta.
Restrições/Validações	25. Validar os campos preenchidos ou editados na função de Edição.
	26. Os campos: “cargo” e “tempo curso” devem ser carregados em forma de lista.
	27. O campo “data nascimento” deve ser carregado em forma de calendário.
	28. Carregar a lista de estagiários associados ao número de cadastro de quem solicita a ação.

Fonte: Próprio Autor.

Quando descrita as atividades do Ator Administrador, pode ser observado a descrição de apenas uma função exercida pelo mesmo, isto devido as demais operações serem iguais a do Ator Funcionário, sendo assim não se faz necessário a descrição novamente de tais funcionalidades.

Figura 14 – Diagrama de Sequência referente as funcionalidades da página de gerenciamento de usuários.



Fonte: Próprio Autor.

Os processos realizados nas contas são os mesmos para administrador e perito, a diferente está em quais contas vão poder exercer a função de cadastrar, editar e desativar.

O administrador terá nível de acesso para fazer todas as operações tanto em contas do tipo estagiário, quanto nas de perito, enquanto que este último só será permitido fazer alterações em contas associadas ao seu código de cadastro.

No diagrama de cadastramento de um novo usuário, é possível observar que o administrador possui uma opção que não é presente para os outros usuários, que seria a escolha do tipo, este associado a: funcionário ou estagiário. Conforme Anexo E.

Depois de fornecido o tipo de conta que será criada, o administrador segue para preencher os mesmos dados que o perito irá fornecer na criação de uma conta do tipo estagiário.

Também é possível realizar a edição destes dados, entretanto, a divisão apresentada anteriormente no processo de criação de contas, não estará presente neste, isto devido à edição ser realizada da mesma maneira a todos, pois não existe mudança de cargo, por exemplo, não é possível ingressar como estagiário e ascender ao cargo de perito criminal. Conforme Anexo F.

O processo realizado na edição dos campos, ou seja, dos dados da conta, tal como: nome, a data de nascimento, data de duração do contrato, etc. Consiste na mudança dos campos já preenchidos ou completar aqueles em branco, havendo então a validação das informações, e se caso não existam erros é registrado no banco de dados os dados e então exibida uma mensagem de sucesso.

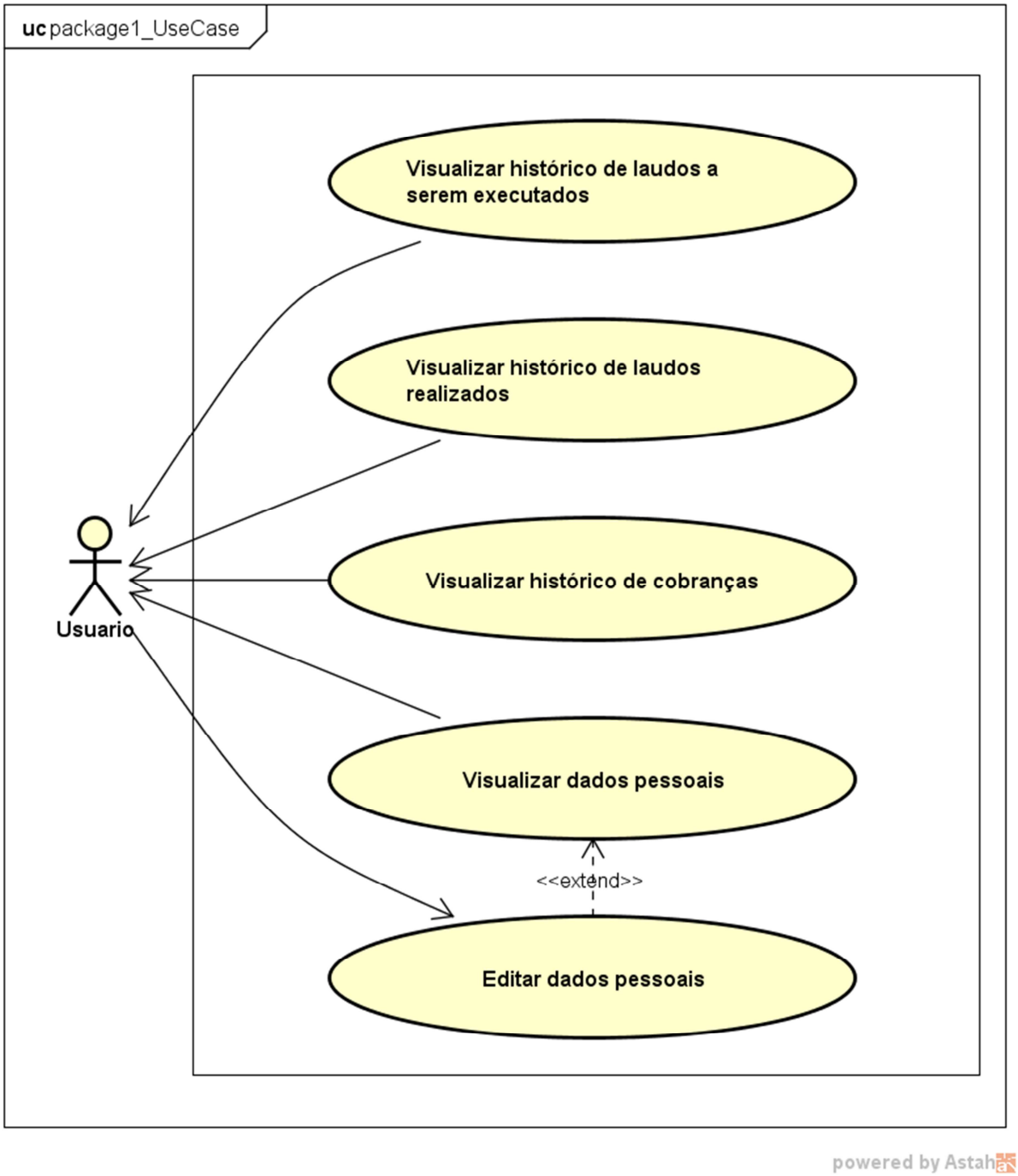
Caso exista alguma inconformidade com os dados preenchidos ou alterados é exibida uma mensagem alertando sobre o erro ocorrido.

Quando é solicitado o processo de desativação de conta estagiário, é então carregada uma lista com os nomes associados a aquela conta funcionário, permitindo então que o usuário selecione quem deseja desativar. A ação é então registrada no banco de dados, imprimindo na tela uma mensagem do sucesso da operação. Conforme Anexo G.

Quando o perito ou administrador solicita a ativação de um contrato que tenha sido interrompido ou finalizado, está ação só é permitida quando a data de termino do curso ainda está em vigor, caso tenha expirado o tempo de conclusão do curso, a ação se torna invalida e é impressa uma mensagem informando o porquê não foi possível reativar o contrato. Conforme Anexo H.

Caso o estagiário desligado ainda esteja no período para conclusão do seu respectivo curso, então a ação é validada e o contrato reativado.

Figura 15 – Diagrama de Caso de Uso referente às ações do usuário quando no próprio perfil.



Fonte: Próprio Autor.

Este diagrama de caso de uso está associado às operações que podem ser realizadas quando consultado o próprio perfil, ou seja, o usuário terá cinco operações, sendo elas apresentadas acima.

A única ação de fato exercida na página de consulta de perfil é a edição dos dados pessoais, ou seja, dados fornecidos na criação da conta. Todas as outras operações são apenas de visualização, de consulta.

Tabela 5 – Documentação do Caso de Uso de Consulta de Perfil.

Nome do Caso de Uso	Consulta de perfil
Atores Principais	Usuário
Ator Secundário	-
Resumo	Este estudo de caso descreve as operações que o usuário encontra ao acessar seu próprio perfil.
Ações do Ator	Ações do Sistema
1. Visualizar dos laudos a serem executados.	
	2. Pesquisar laudos a fazer.
	3. Carregar números de registro dos laudos a fazer.
4. Visualizar dos laudos executados.	
	5. Pesquisar laudos feitos.
	6. Carregar documentos e imagens.
7. Visualizar o das cobranças.	
	8. Pesquisar cobranças.
	9. Exibir números de registros com cobrança.
10. Visualizar dados pessoais.	
	11. Buscar dados cadastrados no código de usuário.
	12. Carregar dados pessoais.
13. Editar dados pessoais.	
	14. Validar formato dos dados inseridos.
	15. Registrar dados no banco.
	16. Exibir mensagem do resultado da operação.
Restrições/Validações	Inserir os dados nos campos no formato adequado.

Fonte: Próprio Autor.

O diagrama de sequência referente ao perfil do usuário vai conter os dados pessoais do próprio, fornecendo a possibilidade de edição de suas informações, isto para os

funcionários, no caso dos estagiários está função não está disponível. A edição dos dados pode ser observada no Anexo F, pois o processo realizado será o mesmo que aquele já explicado. Conforme Anexo I.

4. CONSIDERAÇÕES FINAIS

A proposta de desenvolvimento do *software* File Manager propõe a solução do problema enfrentado com o acumulo excessivo de papeis presente nas instituições técnico-científicas.

O sistema proposto deverá fornecer apoio para todas as atividades realizadas pelos peritos, propiciando uma organização das requisições recebidas, laudos realizados e peças recebidas e enviadas.

Com advento dos métodos e atividades propostas na Engenharia de Software, o sistema deve fornecer uma estrutura que estabeleça uma qualidade a custo-benefício razoável, atendendo a necessidade e capacidade daqueles que irão utiliza-lo.

Com a realização de especificações pré-definidos e objetivos claros estabelecidos, as práticas obtidas puderam ser escolhidas de forma precisa, limitando os erros possíveis que poderiam surgir no desenvolvimento futuro do sistema.

Com este sistema, levando em consideração seu porte, será possível a realização de um prosseguimento em relação ao mesmo, tendo como foco áreas específicas, tal como a segurança da informação, curso presente na mesma instituição de ensino do aluno em questão.

5. REFERÊNCIAS

DELAMARO, E. Márcio.; MALDONADO, C. Carlos.; JINO, Mario. **Introdução ao Teste de Software**. Rio de Janeiro: Elsevier Editora Ltda, 2007. 339 p.

MACHADO, Henrique. **Os 4 Pilares da Programação Orientada a Objetos**. DevMedia, Rio de Janeiro, 30 out. 2014. Disponível em:< http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em 03 ago. 2016.

MENDES, Antônio. **Artigo Engenharia de Software 3 – Requisitos Não Funcionais**. DevMedia, Rio de Janeiro, 03 jul. 2008. Disponível em:< http://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525>. Acesso em 16 out. 2016.

MICROSOFT. **Introdução ao Visual Studio**. Visual Studio 2010. Disponível em:<https://msdn.microsoft.com/pt-br/library/fx6bk1f4(v=vs.100).aspx> . Acesso em 28 set. 2016.

MICROSOFT. **Introduction to Web Forms Pages**: Visual Studio .NET 2003, 2016. Disponível em:<https://msdn.microsoft.com/en-us/library/65tcbxz3(v=vs.71).aspx>. Acesso em 29 out. 2016.

MICROSOFT. **ASP.NET e Visual Studio para Web**. Microsoft Developer Network, ago. 2016. Disponível em:<https://msdn.microsoft.com/pt-br/library/dd566231.aspx>. Acesso em 22 out. 2016.

MYSQL. **Chapter 1 General Information**. MySQL. Disponível em:<http://dev.mysql.com/doc/workbench/en/wb-intro.html>. Acesso em 06 nov. 2016.

PRESSMAN, Roger S. **Engenharia de Software – Uma Abordagem Profissional**. 7ª. ed. São Paulo: McGraw-Hill, 2011. 773 p.

RIOS, Emerson.; MOREIRA, Trayahú. **Teste de Software**. 2ª ed. Rio de Janeiro: Alta Books, 2007. 222 p.

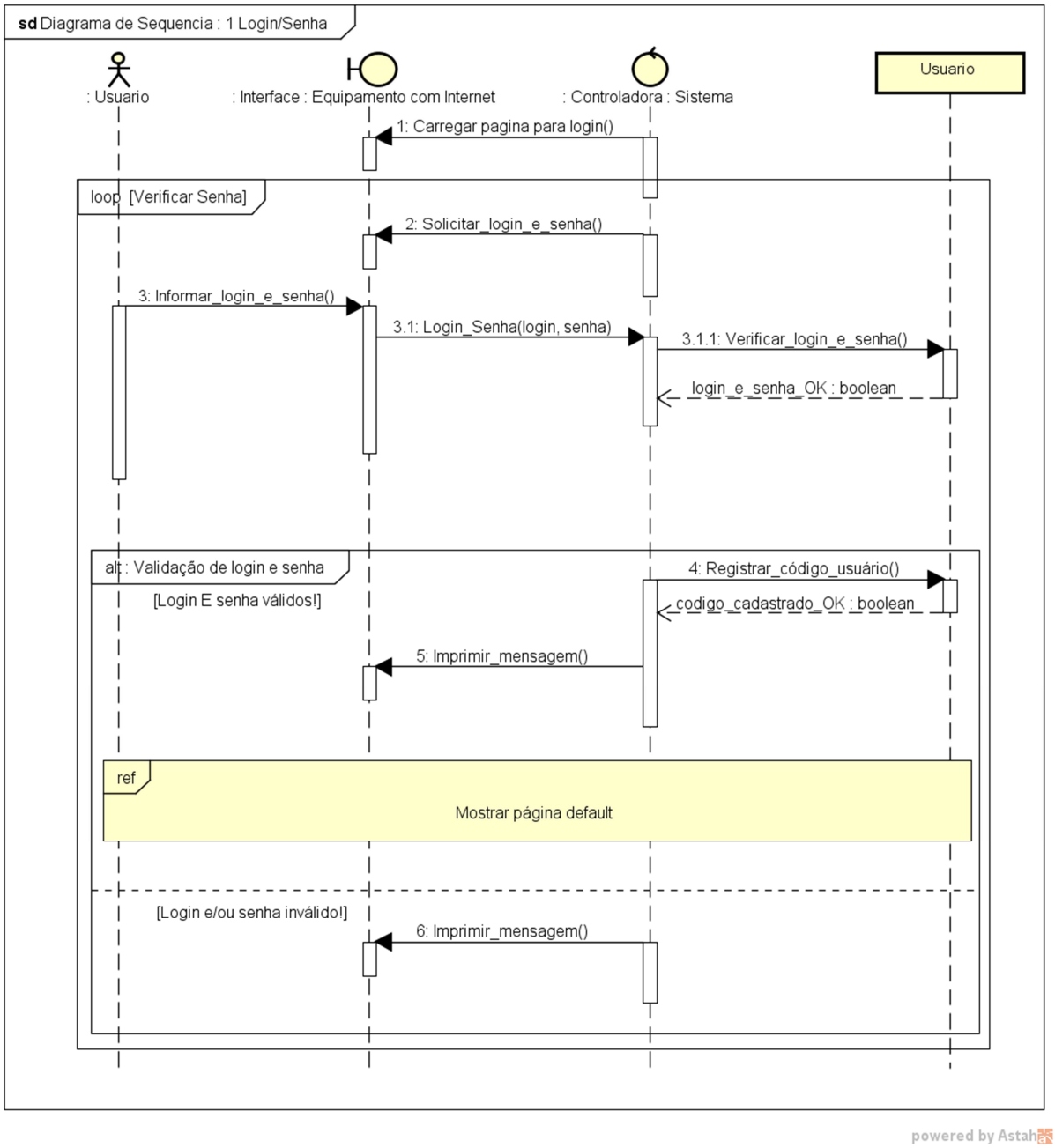
RODRIGUES, Joel. **Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER)**. DevMedia, Rio de Janeiro, 16 dez. 2014. Disponível em:<http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>. Acesso em 28 mai. 2016.

SOMMERVILLE, Ian. **Engenharia de Software.** 6ª. ed. São Paulo: Addison Wesley, 2003. 585 p.

UFCG. **Caso de Uso: Diagrama de Casos de Uso**, 2007. Disponível em:<http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/usecases.htm>. Acesso em 16 mai. 2016.

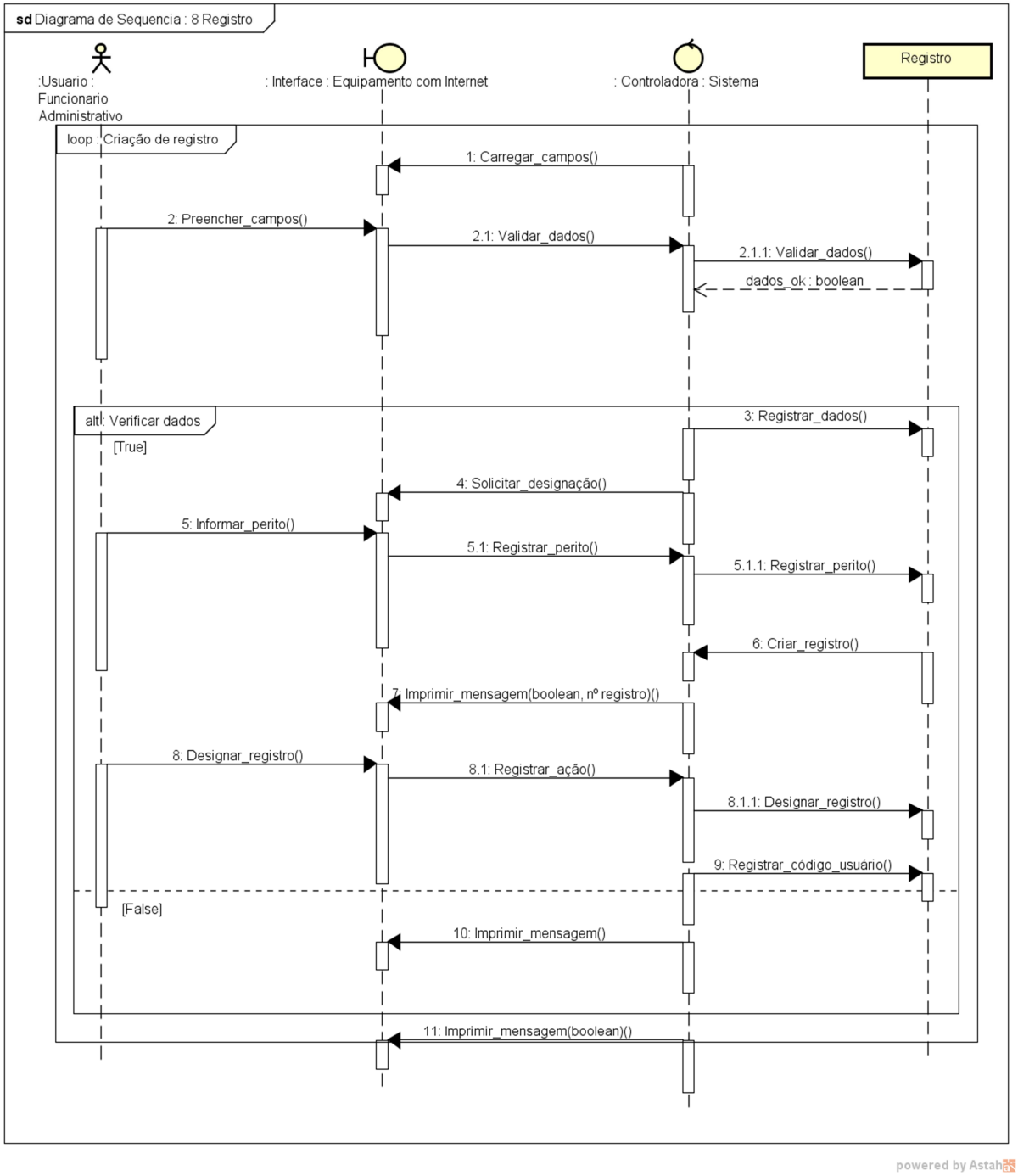
6. ANEXOS

ANEXO A - Diagrama de Sequência referente ao *login* no sistema.



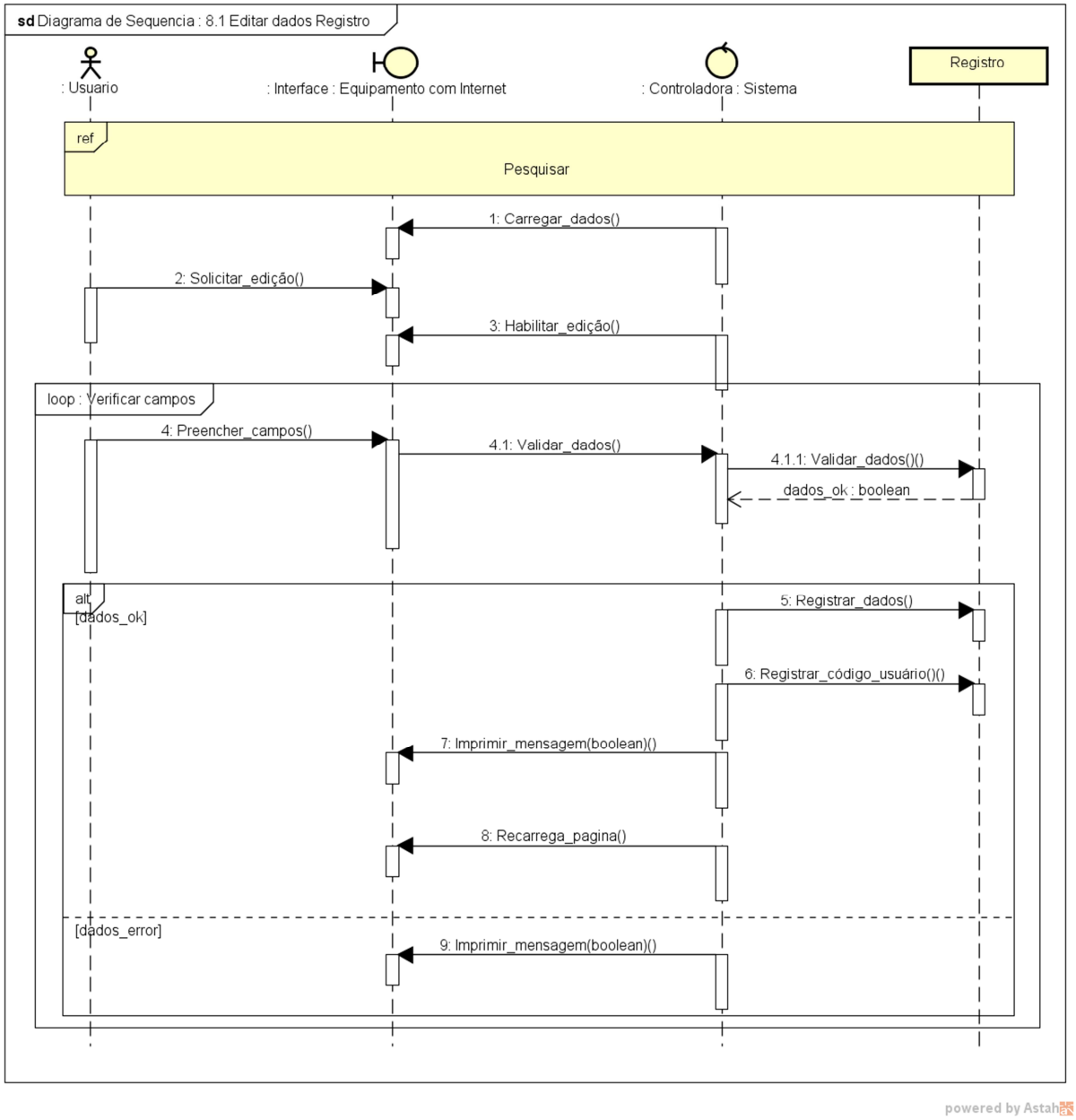
Fonte: Próprio Autor.

ANEXO B - Diagrama de Sequência referente à criação do registro.



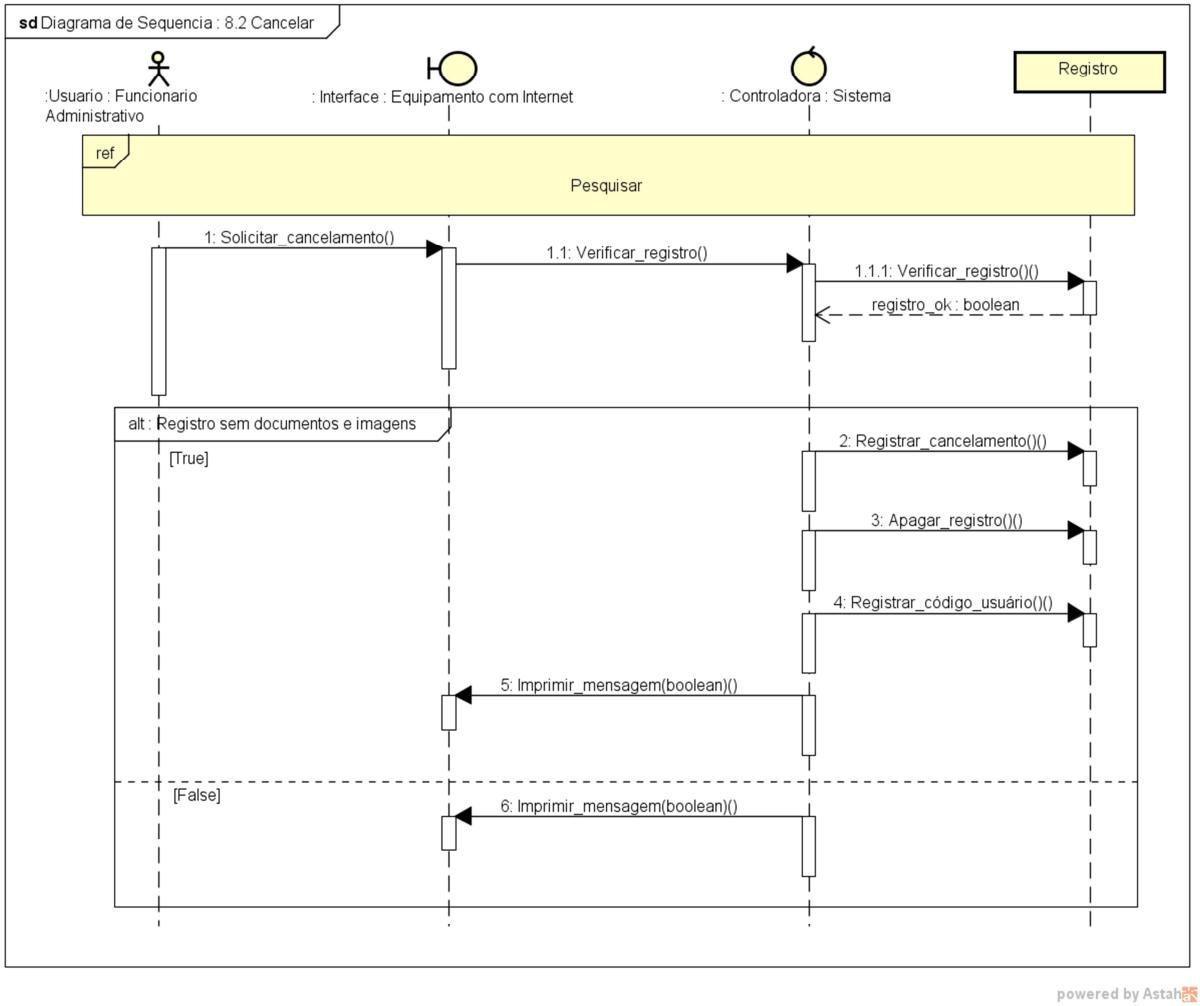
Fonte: Próprio Autor.

ANEXO C - Diagrama de Sequência referente à edição dos dados do registro.



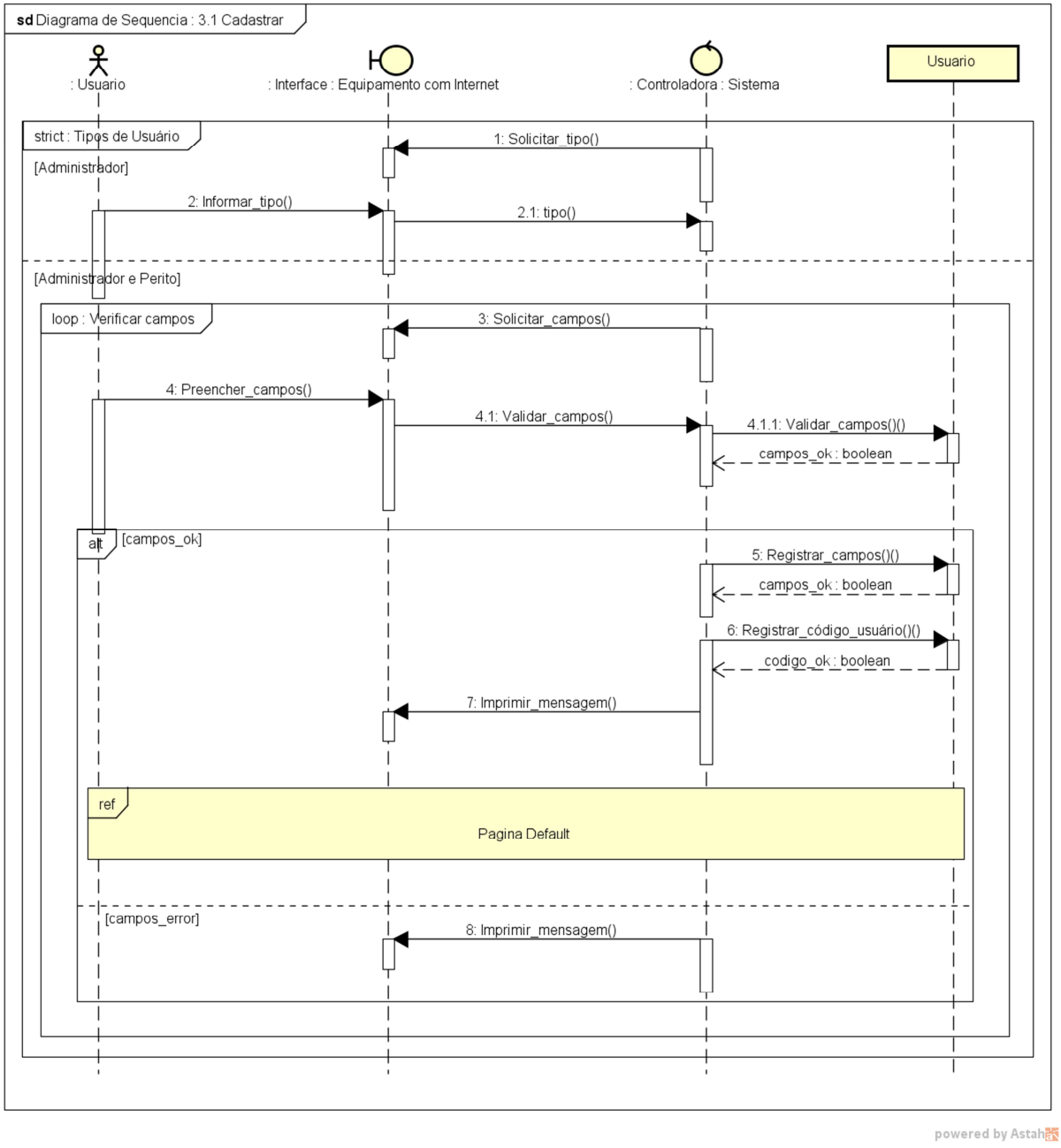
Fonte: Próprio Autor.

ANEXO D - Diagrama de Sequência referente ao cancelamento do registro.



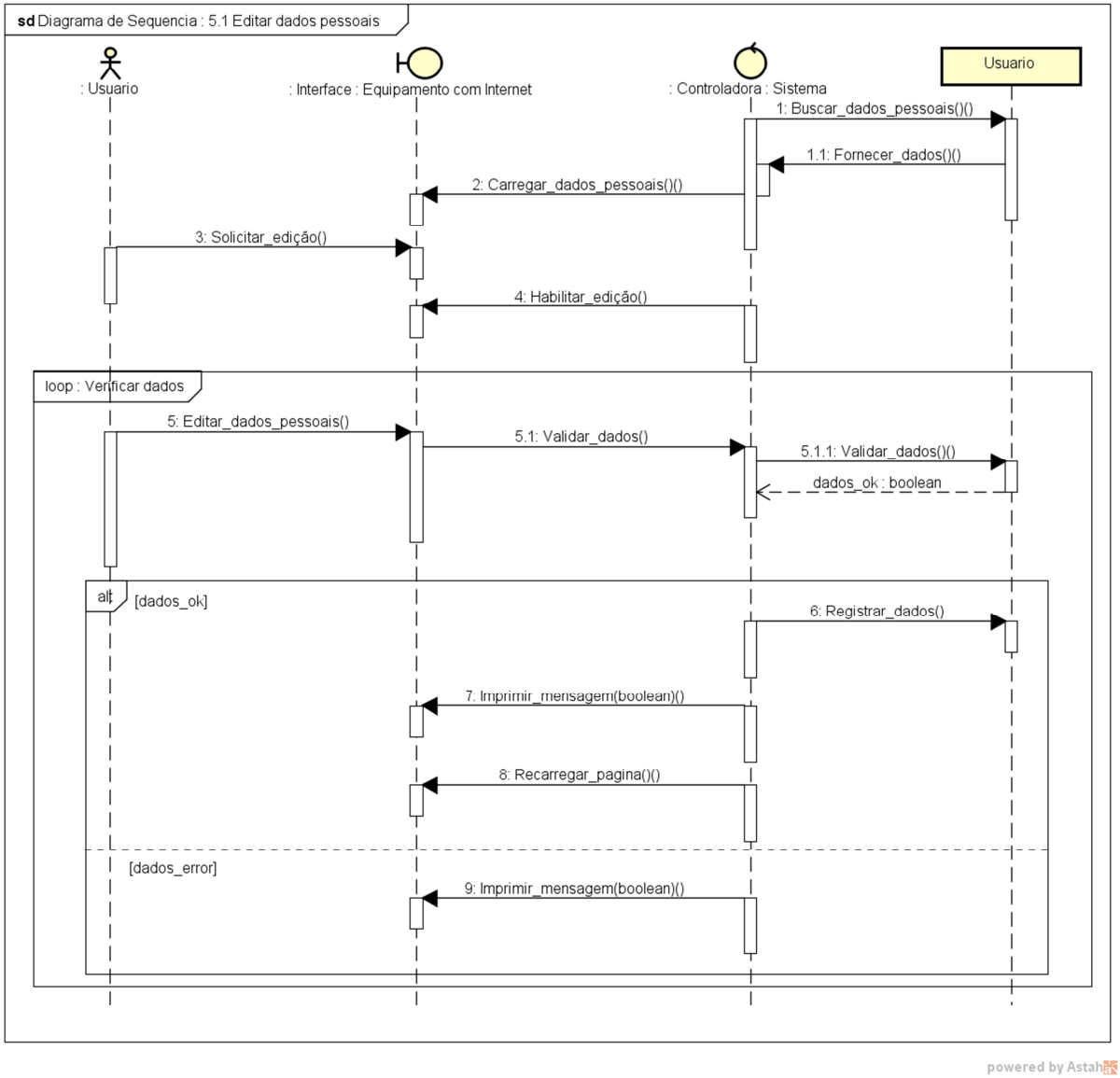
Fonte: Próprio Autor.

ANEXO E - Diagrama de Sequência referente ao cadastro de um novo usuário.



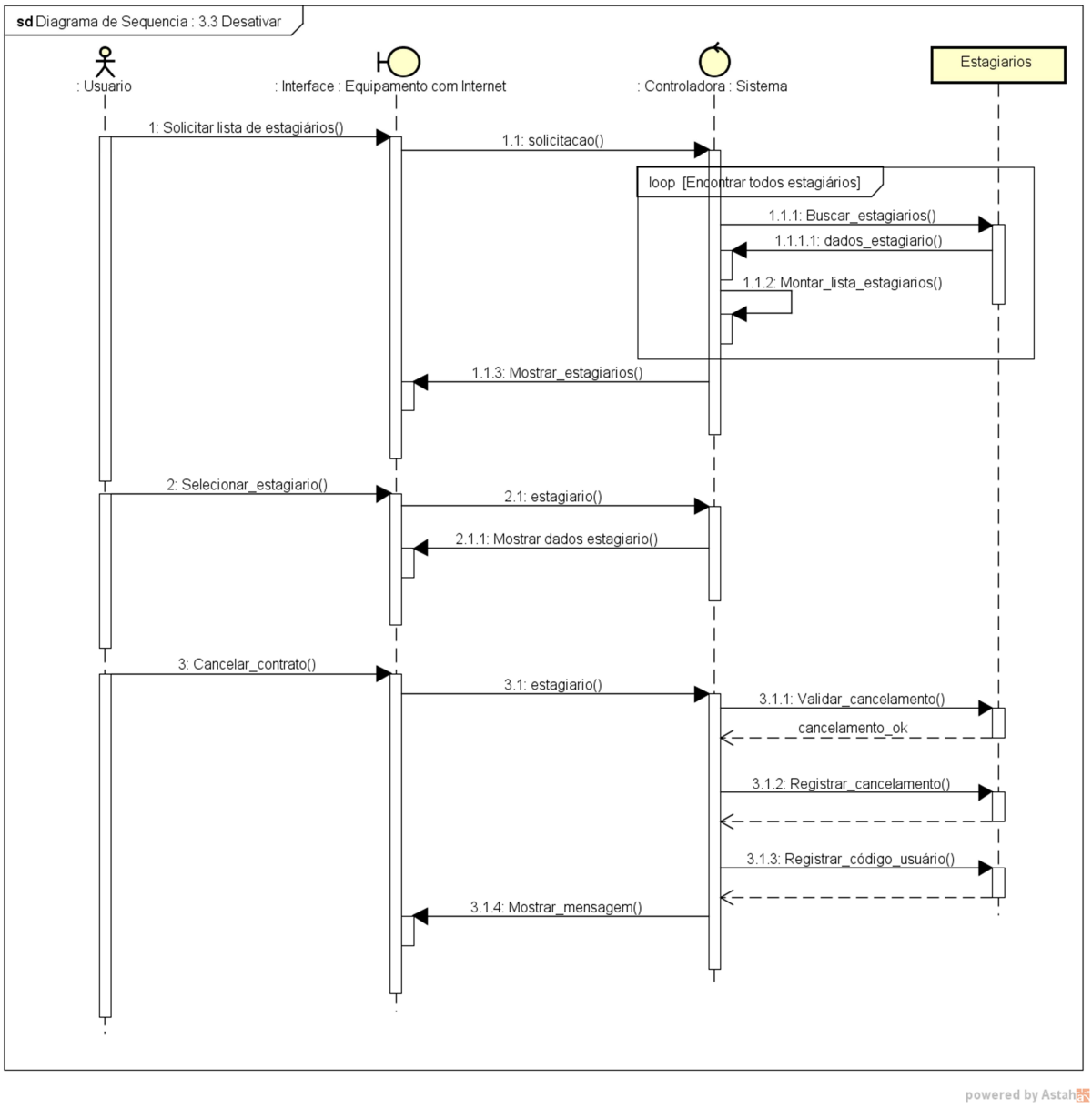
Fonte: Próprio Autor.

ANEXO F - Diagrama de Sequência referente à edição dos dados do usuário.



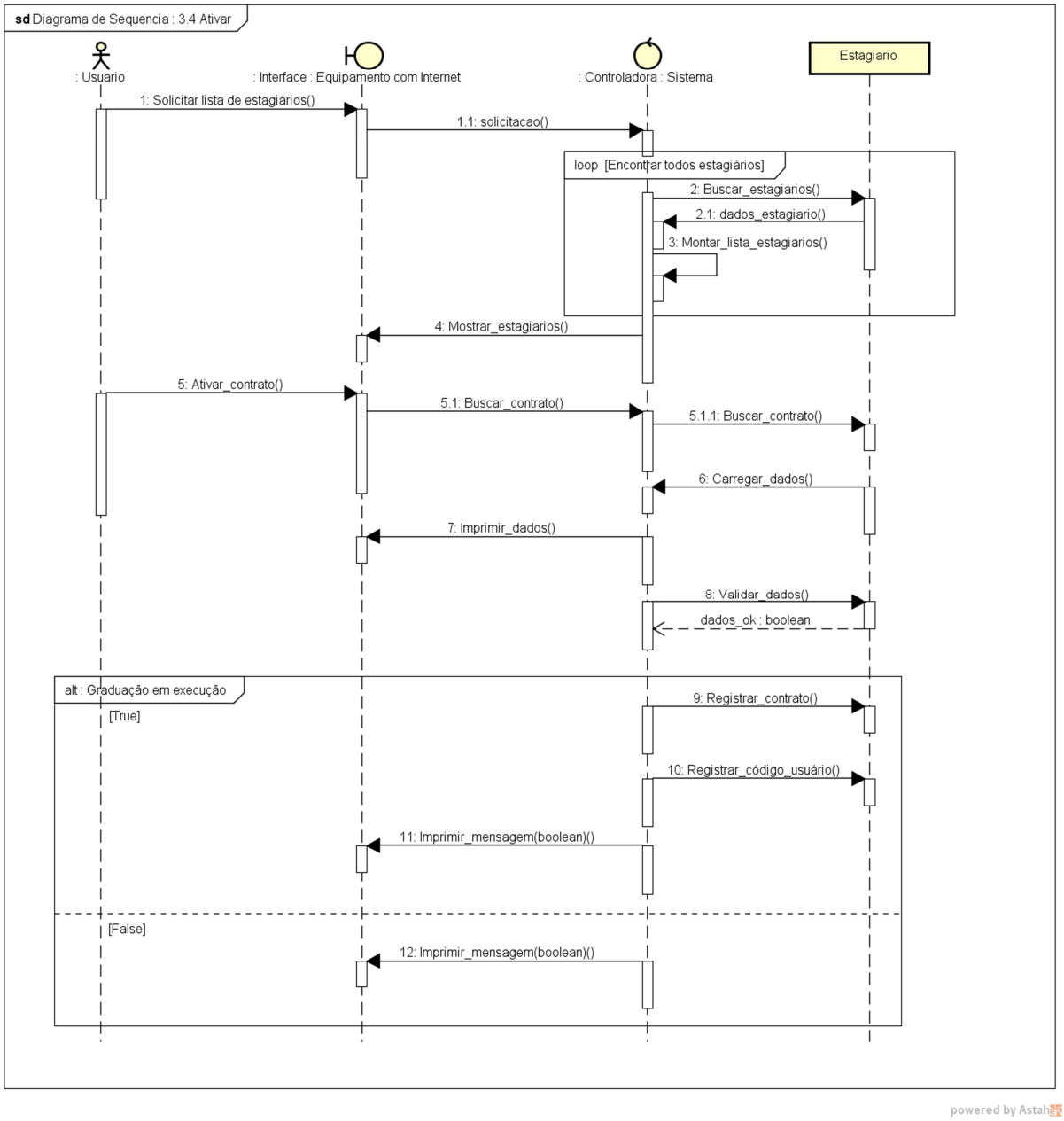
Fonte: Próprio Autor.

ANEXO G - Diagrama de Sequência referente ao cancelamento da conta do estagiário.



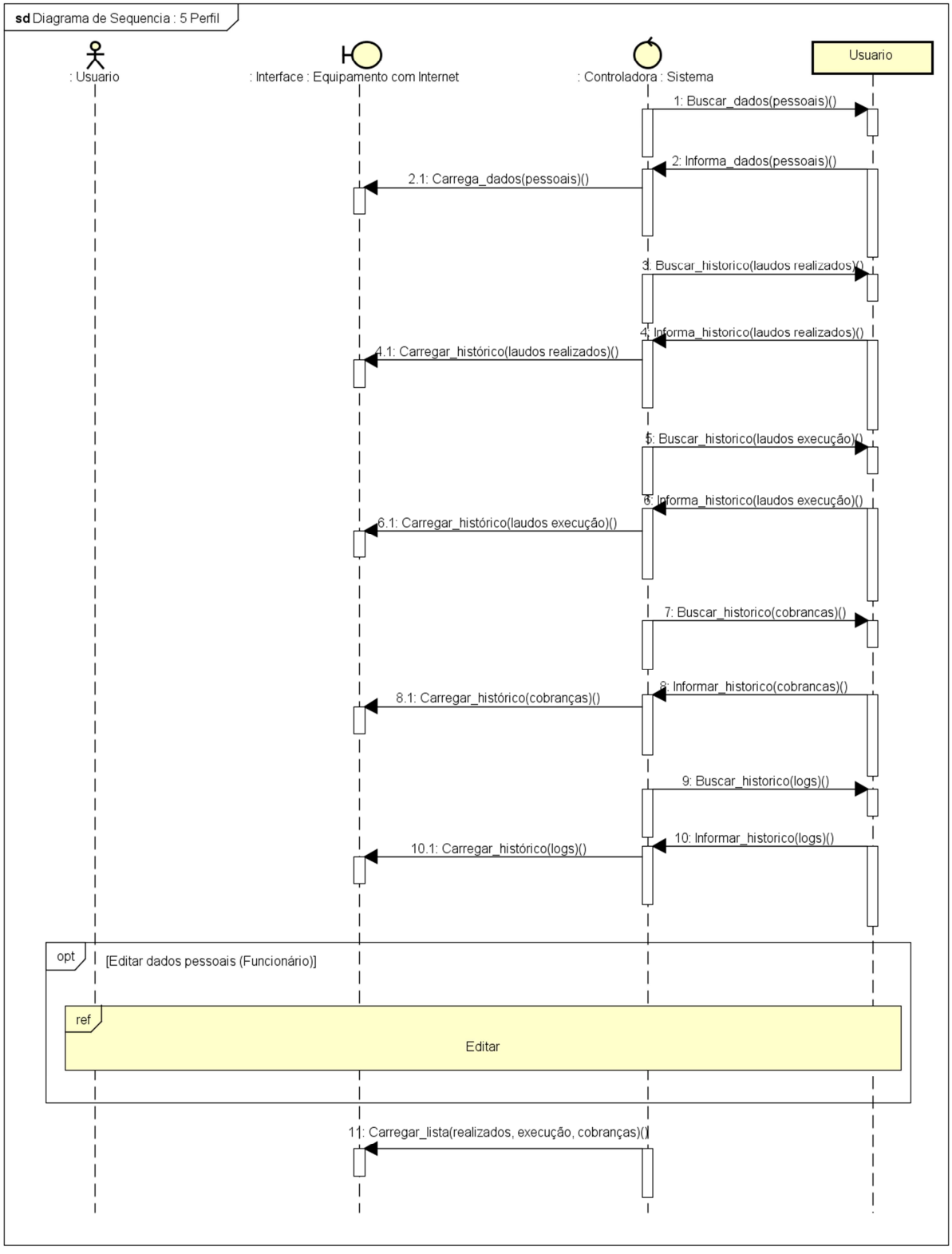
Fonte: Próprio Autor.

ANEXO H - Diagrama de Sequência referente à ativação do contrato do estagiário.



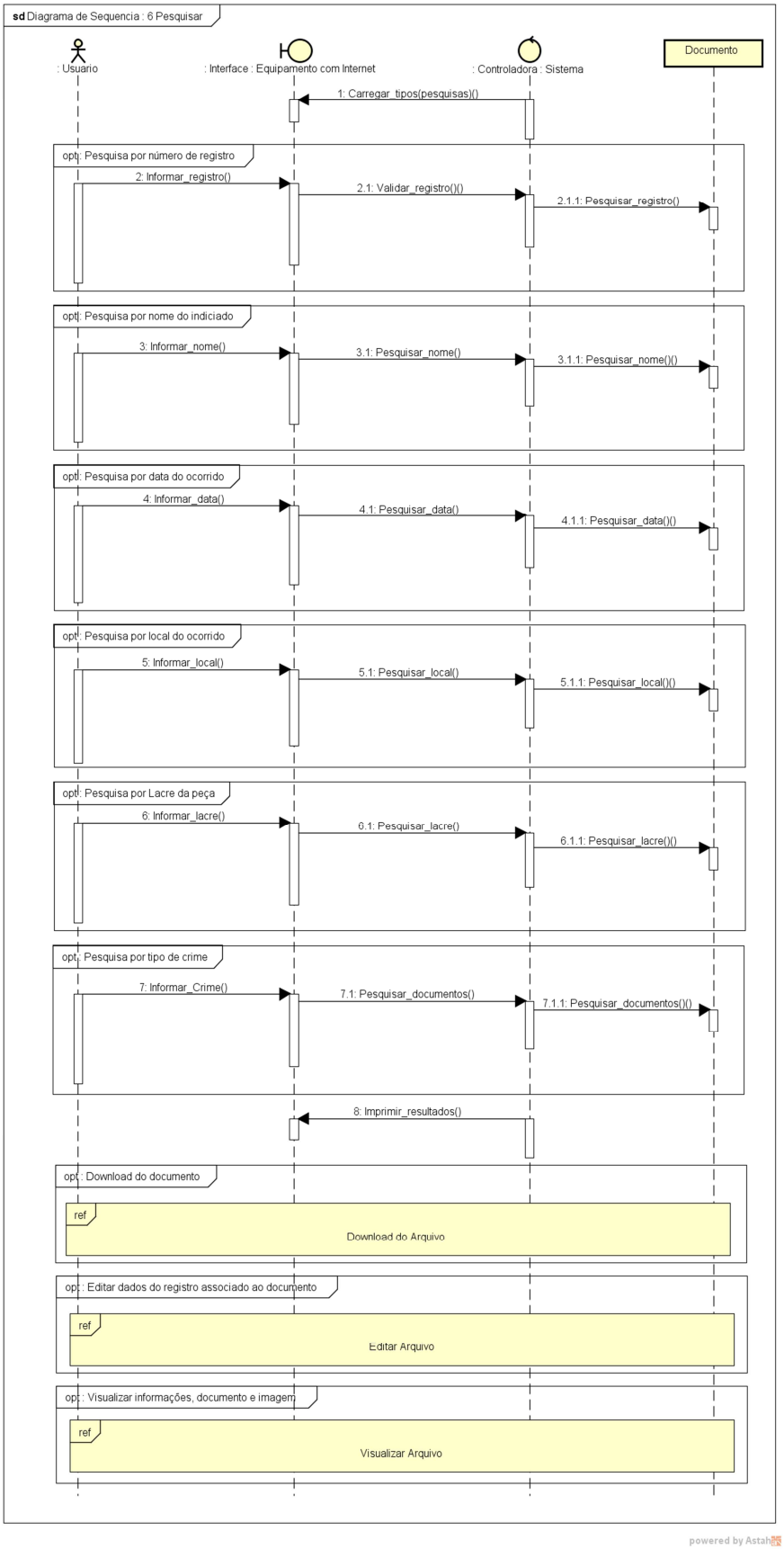
Fonte: Próprio Autor.

ANEXO I - Diagrama de Sequência referente à página de consulta de perfil.



Fonte: Próprio Autor.

ANEXO J - Diagrama de Sequência referente à pesquisa.



Fonte: Próprio Autor.