UNIVERSIDADE ESTADUAL DE GOIÁS
UNIDADE UNIVERSITÁRIA DE ITABERAÍ

WANDERSON KAIQUE DE JESUS SANTOS

ANÁLISE E DESENVOLVIMENTO DE UM GESTOR DE SERVIÇOS PARA A FÁBRICA DE RAÇÕES SÃO SALVADOR ALIMENTOS.

Itaberaí

2018

Wanderson Kaique de Jesus Santos

Análise e desenvolvimento de um gestor de serviços para a fábrica de rações da São Salvador Alimentos.

Trabalho Final de Curso apresentado à Universidade de Goiás, Campus Itaberaí, como requisito parcial para a conclusão do curso de Bacharelado em Sistema de Informação, sob orientação da Professora especialista Juliana Vasconcelos Braga.

ITABERAÍ

2018

A Deus, por ter me dado saúde, força e paciência par superar as dificuldades.
A minha noiva pelo total apoio nessa caminhada vitoriosa.
Aos meus amigos e colegas, que me proporcionaram, até o momento, os melhores anos da minha vida.

Agradeço primeiramente a Deus por ter me dado força para dar continuidade neste trabalho, agradeço aos professores, em especial à professora Juliana, pela paciência na orientação e agradeço aos meus colegas que também me ajudaram com a elaboração deste trabalho de conclusão de curso.

“A verdadeira motivação vem de realização, desenvolvimento pessoal, satisfação no trabalho e reconhecimento.”

Frederick Herzberg

RESUMO

Este projeto descreve as etapas de documentação de um sistema *desktop* a ser desenvolvido com a finalidade de organizar as atividades diárias da fábrica de rações da São Salvador Alimentos (SSA). O projeto coloca em prática o conhecimento adquirido em quatro anos de curso em Sistemas de Informação na Universidade Estadual de Goiás Campus Itaberaí. Para a realização deste trabalho as metodologias usadas foram, uma revisão bibliográfica e o uso de ferramentas para a modelagem de dados. O uso da tecnologia da informação irá melhorar o gerenciamento das atividades que são propostas na empresa, prevenindo o acúmulo e o excesso de atividades em determinados colaboradores da empresa.

Palavras-chave: Documentação. Sistema desktop. Revisão bibliográfica. Modelagem de dados. Tecnologia da informação.

ABSTRACT

The project describes how to document documentation of a desktop system and installation of daily activities of the São Salvador Alimentos (SSA) feed mill. The project is in practice the knowledge acquired in four years of course in Information Systems at the State University of Goiás Campus Itaberaí. The date of this work was a methodology for data modeling, a bibliographical review and the use of tools. The use of technology is the performance of activities, avoiding the accumulation and excess of activities in certain employees of the company.

Keywords: Documentation. Desktop system. Literature review. Data modeling. Information Technology.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

ANSI	American National Atandart Institute (Instituto Nacional Americano de Padrões)
DDL	Data Definition Language (Linguagem de Definição de Dados)
DML	Data Manipulation Language (Linguagem de Manipulação de Dados)
IDE	Integrated Development Environment (Ambiente Integrado de Desenvolvimento)
SQL	Structured Query Language (Linguagem de Consulta Estutural)
TCC	Trabalho de Conclusão de Curso
TI	Tecnologia da Informação
UEG	Universidade Estadual de Goiás
UFSC	Universidade Federal de Santa Catarina
UML	Unified Modeling Language (Linguagem de modelagem Unificada)

SUMÁRIO

1. INTRODUÇÃO

Organizar o tempo, ou melhor, definir a gestão do tempo é a base de desenvolvimento desse trabalho, como organizar melhor a gestão do tempo? Como gerir o tempo? É na busca de respostas a esses questionamentos que se objetiva o estudo, a pesquisa e o desenvolvimento do software capaz de auxiliar no desenvolvimento dos serviços de uma empresa.

1.1 Descrição do Problema

A fábrica de rações possui áreas cujas atividades não são realizadas diariamente, mas sim de acordo com suas necessidades. As atividades dessas áreas são realizadas por qualquer um dos colabores, eles não são responsáveis apenas por uma determinada atividade e sim pelas atividades que o encarregado passa para ele.

Como as atividades dessas áreas são indicadas aos colaboradores, alguns colaboradores podem se sobrecarregar devido ao maior número de atividades estabelecidas para o ele e ao mesmo tempo outros estarem com menos atividades a serem realizadas.

1.2 Proposta de Solução

Analisando os problemas encontrados na empresa, foi realizada uma proposta para a criação um sistema que propõe a organização das atividades entre os colaboradores dessas áreas. O objetivo do sistema, além da organização das atividades entre os colaboradores, informa se as mesmas foram iniciadas, se estão em andamento, se já foram encerradas ou canceladas e auxilia na rapidez da entrega desses serviços.

O uso da tecnologia da informação irá melhorar o gerenciamento dessas atividades, evitando que o colaborador fique sobrecarregado com muitas atividades a serem realizadas e evitando que outro colaborador fique com poucas atividades, assim dividindo de forma igualitária essas atividades.

1.3 Objetivos

O estudo de caso mostra as necessidades da empresa, servindo de embasamento para o desenvolvimento da documentação de um sistema que ajudará a organização das atividades a serem realizadas, dividindo as mesmas deforma igualitária entre os colaboradores.

O objetivo principal é o desenvolvimento da documentação de um sistema que possibilite a aplicação dos conhecimentos e técnicas adquiridas no decorrer de quatro anos de curso em Sistema de Informação na Universidade Estadual de Goiás (UEG), Campus Itaberaí. A documentação desse projeto será elaborada para oferecer melhor o esclarecimento sobre documentos e diagramas construídos.

Os objetivos específicos são: desenvolver uma análise detalhada das reais necessidades que precisarão ser supridas, apresentar a solução para o problema real, desenvolver uma documentação de qualidade.

O sistema que facilitará no gerenciamento das atividades entre os colaboradores. O sistema a ser desenvolvido fornecerá informações sobre as atividades inseridas no sistema, sobre os colaboradores ligados a empresa e na situação em que se encontram as atividades já iniciadas.

O trabalho está dividido em cinco capítulos e um apêndice que se resumem a seguir.

O primeiro capítulo (Introdução) apresenta um texto breve que antecede a obra escrita, e que serve para apresentar a documentação do sistema ao leitor como.

O segundo capítulo (METODOLIA) apresenta uma fundamentação teórica sobre as metodologias que serão utilizadas para o desenvolvimento de cada etapa do projeto.

O terceiro capítulo (ESTUDO DE CASO) apresenta os métodos e abordagens para investigar o problema.

O quarto capítulo (PROJETO) apresenta a descrição detalhada e a modelagem de dados do sistema a ser desenvolvido.

O quinto capítulo (CONCLUSÃO) apresenta a conclusão do que foi proposto para resolver o problema apresentado.

O apêndice (GLOSSÁRIO DE MENSAGENS) apresenta as mensagens do sistema a ser desenvolvido.

Por fim, em REFERÊNCIAS BIBLIOGRAFICAS, listam-se por ordem alfabética, todas as referências bibliotecas utilizadas ao longo do projeto.

2 METODOLOGIA

Neste capítulo será apresentado a metodologia da análise do presente trabalho, abordando os conceitos e as principais funções de cada um dos assuntos compostos na estrutura teórica necessária da coleta, análise e interpretação dos dados. A análise será dividida em duas partes, o primeiro trata da Engenharia de Software e o outro da Engenharia de Requisitos.

2.1 Engenharia de Software

A Engenharia de Software é a disciplina responsável pela produção de um software, ela tem como objetivo possibilitar aos profissionais desenvolverem um software de alta qualidade. De acordo com Sommerville (2007), a Engenharia de Software não se relaciona apenas com os processos, mas também com as atividades.

> A Engenharia de Software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software. (SOMMERVILLE, 2007, p. 5).

Mesmo com muitas propostas definidas, todas elas orientam ao uso da engenharia no desenvolvimento do software. São os engenheiros de software que fazem esse processo funcionar. Para os engenheiros a maneira mais eficaz de produzir um software com qualidade e adotando uma abordagem organizada assim facilitando o desenvolvimento do sistema.

2.1.1 Processo de Software

Para produzir um software de qualidade é necessário um bom projeto, mas para isso e preciso que essa criação passe por processos que permitiram o seu desenvolvimento de forma apropriada seguindo os padrões necessários. Para se obter um produto com qualidade e essencial que o seu processo seja de qualidade. Precisamos entender os processos de desenvolvimento para podermos entender melhor o desenvolvimento do software.

> Um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. Existem quatro atividades fundamentais de processo que são comuns a todos os processos de software: Especificação de software, desenvolvimento de software, validação de software e evolução de software. (SOMMERVILLE, 2007, p. 6).

Os processos citados por Sommerville (2007) são atividades bem utilizadas pelos criadores de software. Esses processos descrevem um método de organização das atividades,

modelando as atividades em fases e determinando o relacionamento dessas fases. Esse processo é importante para a inicialização do projeto, assim definindo como o mesmo será conduzido.

- **Especificação**

A especificação tem a finalidade de descrever o que o software deve fazer. Seu estudo descreve soluções de como o software conseguirá resolver os problemas levantados na análise. Para resolver problemas de domínio ela descreve quais as propriedades funcionais são necessárias, assim classificando as de forma organizada.

- **Desenvolvimento**

O desenvolvimento é onde a especificação do sistema se converte em um sistema executável, e nessa área onde se envolve a programação do software.

- **Validação**

O processo da validação de software e onde comprova, com base nos documentos, que o sistema esta cumprindo suas funções de acordo com o que foi estabelecido, garantia da segurança das informações do sistema e agindo conforme as especificações dos requisitos.

- **Evolução do Software**

A evolução de software vem com a finalidade de descrever as mudanças que são realizadas no sistema, para que o mesmo fique com as necessidades na empresa e se adaptando as novas tecnologias.

2.1.2 Modelos de processos de Software

De acordo com Sommerville (2007) um modelo de processo de software é uma representação abstrata de um processo de software, cada modelo de processo representa um processo sob determinada perspectiva e, dessa forma, fornece somente informações parciais sobre processo.

Esses modelos não são descrições definitivas de processo de software, ao contrário, são abstrações do processo que podem ser usadas para explicar diferentes abordagens para o desenvolvimento de software e podem ser considerados como frameworks de processo que podem ser ampliadas e adaptadas para criar processos mais específicos de engenharia de software (SOMMERVILLE, 2007).

São três os modelos de processo citados por Sommerville (2007). São eles:

- **Modelo em cascata**

É caracterizado pela sequência que as etapas do desenvolvimento seguem, o início da quinta etapa só funcionara com o término da quarta, a quarta etapa só funcionará com o término da terceira e assim sucessivamente. Por tanto cada etapa do modelo só terá início com o término a anterior.

De acordo com Sommerville (2007) os principais estágios do modelo demonstram as atividades fundamentais de desenvolvimento, são eles: Análise e definição de requisitos, projeto de sistema e software, implementação e teste da unidade, integração e teste do sistema e operação e manutenção.

Sua vantagem e com a parte de documentação dividida por etapa, mas a sua desvantagem está na divisão inflexível do projeto dessas etapas. Portanto, o modelo em cascata deve ser usado apenas quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudanças radicais durante o desenvolvimento do sistema (SOMMERVILLE, 2007).

- **Desenvolvimento evolucionário**

Para Sommerville (2007) o desenvolvimento evolucionário baseia-se na ideia de desenvolvimento de uma implementação inicial, expondo o resultado aos comentários do usuário e refinando esse resultado por meio de várias versões até que seja desenvolvido um sistema adequado.

Para Sommerville (2007) existem dois tipos fundamentais de desenvolvimento evolucionário. São eles:

- Desenvolvimento exploratório.

Tem como principal objetivo o trabalho junto ao cliente, assim explorando os requisitos e entregando o sistema final.

- Prototipação *throwaway*.

Tem como finalidade analisar os requisitos do cliente, para selecionar quais deles são viáveis. Compreendendo os requisitos obtidos pelos clientes, tem por finalidade desenvolver uma melhor definição desses requisitos.

> Uma abordagem evolucionária para o desenvolvimento de software é frequentemente mais eficaz do que a abordagem em cascata na produção de sistemas que atendem às necessidades imediatas dos clientes. A vantagem de um processo de software baseado na abordagem evolucionária é que a especificação pode ser desenvolvida de forma incremental. (SOMMERVILLE, 2007, p. 45).

- **Engenharia de software baseada em componentes.**

Tem como foco a reutilização de algum software. Segundo Sommerville (2007) a maioria dos projetos existe algum reuso de software. De maneira informal, quando os profissionais que trabalham com o projeto e já conhecem os códigos, eles o modificam e os incorporam ao sistema.

Para Sommerville (2007) a abordagem orientada a reuso depende de uma grande base de componentes de software reusáveis e alguns framework de integração desses componentes.

Segundo Sommerville (2007) o estágio de especificação inicial de requisitos e o estágio de validação sejam comparáveis a outros processos, os estágios intermediários de um processo orientado a reuso são diferentes, esses estágios são o de análise de componente, modificação de requisitos, projeto de sistema com reuso e desenvolvimento de integração.

A engenharia de software baseada em componentes tem a vantagem de diminuir os custos e os riscos, pois ela trabalha com o reuso do software a ser desenvolvido. Por esse reuso a entrega do software acontece de forma rápida.

No entanto para Sommerville (2007), a formulação dos requisitos ainda é necessária, pois isso pode levar a um sistema que não atenda às reais necessidades dos usuários e algum controle sobre a evolução do sistema será perdido se novas versões dos componentes reusáveis não estiverem sob o controle da organização que a utiliza.

2.2 Engenharia de Requisitos

Para Sommerville (2007) os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais, eles refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema, por exemplo, controlar um dispositivo, enviar um pedido ou encontrar informações. Engenharia de requisitos tem como função descobrir, analisar, documentar e verificar.

Esses requisitos são classificados em:

- Requisitos funcionais

Os requisitos funcionais descrevem como o software deve funcionar. De acordo com Sommerville (2007) esse requisitos dependem do tipo do software que está sendo desenvolvido, dos usuários a que o software se destina e da abordagem geral considerada pela organização ao redigir os requisitos.

- Requisitos não Funcionais

Para Sommerville (2007) os requisitos não funcionais podem estar relacionados às propriedades emergenciais do sistema, com confiabilidade, tempo de resposta e espaço de armazenamento.

Os requisitos irão fornecer ao software referencias que iram validar o produto final. Para Sommerville (2007) o objetivo de processo de engenharia de requisitos é criar e manter um documento de requisitos de sistema. Esse processo inclui quatro subprocessos. São eles:

- **Estudo de viabilidade**

O estudo de viabilidade tem como função avaliar a parte financeira e econômica, ela procura determinar se a empresa terá um bom desempenho financeiro e econômico com um eventual projeto.

> Em todos os sistemas novos, o processo de engenharia de requisitos deve começar com um estudo de viabilidade. A entrada para o estudo de viabilidade consiste de um conjunto preliminar de requisitos de negócio, um esboço da descrição do sistema e com o sistema pretende apoiar os processos de negócio. (SOMMERVILLE, 2007, p. 97).

De acordo com Sommerville (2007), a realização de um estudo de viabilidade envolve a avaliação de informações, sua coleta e a elaboração de um relatório. Para Sommerville (2007) nesse estudo você pode consultar fontes de informações como os gerentes de departamento em que o sistema será usado, engenheiros de software familiarizados com o tipo de sistema proposto, especialistas em tecnologia e usuários finais do sistema.

- **Elicitação e análise de requisitos**

A elicitação e analise de requisitos tem como função de entrevistar os clientes e usuários finais do sistema, assim aprendendo mais sobre a aplicação do sistema, suas funcionalidades e seu desempenho. Esse processo pode envolver varias pessoas de uma organização, que podem ser representadas pelo termo *stakeholder* que se refere a qualquer pessoa ou grupo que estão ligados ao sistema.

De acordo com Sommerville (2007) cada organização terá sua própria versão ou modelo desse modelo geral, dependendo de fatores locais, como o nível de conhecimento da equipe, o tipo de sistema que está sendo desenvolvido e os padrões usados.

> Novamente, você pode pensar nessas atividades como um espiral, de modo que as atividades se intercalem à medida que o processo progrida da parte interior da espiral para a exterior. (SOMMERVILLE, 2007, p. 99)

As atividades desse processo são:

- Obtenção de requisitos

Interação com pessoas que estão ligadas ao sistema, assim coletando as informações necessárias para a elaboração dos requisitos.

- Classificação e organização de requisitos

Atividade que envolve a organização dos requisitos que foram coletados na atividade anterior.

- Priorização e negociação de requisitos

Essa atividade envolve a priorização de requisitos e propõe, por meio de negociação, resolver os conflitos entre os requisitos.

- Documentação de requisitos

A documentação de requisitos tem com função armazenar os requisitos formais ou informais.

- **Validação de requisitos**

A validação de requisitos determina se os requisitos obtidos estão atendendo as necessidades que o sistema precisa para atender o usuário. Para Sommerville (2007) a validação de requisitos é importante porque os erros em um documento de requisitos podem levar a custos excessivos de retrabalho quando são descobertos durante o desenvolvimento ou depois que o sistema está em operação.

A validação de requisitos necessita de verificações nos requisitos do documento de requisitos durante seu processo, essas verificações são de validade, consistência, completeza, realismo e facilidade de verificação.

Sommerville (2007) cita três técnicas de validação de requisitos que podem ser usadas em grupo ou individual. São ela:

- Revisão de requisitos

Uma equipe que envolve pessoas de ambas as organizações para determinar algum problema com o documento de requisitos.

- Prototipação

De acordo com Sommerville (2007) prototipação, usando a abordagem de validação, e um modelo executável do sistema é apresentado para usuários finais e clientes.

- Geração de casos de teste

Para descobrir se existem problemas com os requisitos, os testes realizados deveram ser criados como parte do processo de validação. Caso houver teste difíceis de serem realizados, provavelmente os requisitos serão difíceis de serem implementados.

Sommerville (2007) cita que não se deve subestimar os problemas de validação de requisitos. Para ele e difícil demostrar que o conjunto de requisitos atende às necessidades do usuário, os usuários devem imaginar o sistema em operação e avaliar sua adequação ao trabalho.

> É difícil para profissionais de informática habilidosos realizarem esse tipo de análise abstrata e é ainda mais difícil para os usuários do sistema. Como resultado você raramente encontra todos os problemas de requisitos durante o de validação. (SOMMERVILE, 2007, p. 106).

- **Gerenciamento de requisitos**

De acordo com Sommerville (2007) os requisitos de sistema estão sempre mudando, um motivo para isso é que esses sistemas são geralmente criados para lidar com problemas perversos.

Após a instalação do sistema outros requisitos surgiram. Para os usuários e os clientes é difícil prever quais serão aos efeitos ocorridos após a implementação do sistema na organização.

Para Sommerville (2007), o gerenciamento de requisitos é um processo para compreender e controlar as mudanças dos requisitos de sistema, e preciso manter o acompanhamento dos requisitos individuais e manter as ligações entre os requisitos dependentes, de modo que seja possível avaliar o impacto das mudanças de requisitos.

> O processo de gerenciamento de requisitos deve se iniciar assim que uma versão inicial do documento de requisitos esteja disponível, mas você deve iniciar o planejamento das mudanças de requisitos durante o processo de elicitação de requisitos. (SOMMERVILLE, 2007, p.106).

3 Estudo de Caso

Neste capitulo será descrito a análise e documentação de um sistema para a fábrica de rações da empresa São Salvador Alimentos (SSA). Nele contém uma breve descrição da empresa, do problema e uma proposta para a melhoria desse problema. Também será descrito os objetivos do sistema e a documentação para o seu desenvolvimento.

3.1 Apresentação da Empresa

A fábrica de rações da São Salvador Alimentos fundada na cidade de Itaberaí, estado de Goiás trabalha com a produção e distribuição de rações. A fábrica de rações é toda informatizada e automatizada, funcionando dentro dos níveis mais exigentes de bioseguridade.

Tem capacidade para produzir até 130 toneladas por hora e está dividida em duas linhas de produção independentes, o que permite flexibilizar a fabricação já que há diferentes formulações de acordo com o sexo e a idade das aves. Cada lote de ração produzida tem uma amostra coletada para rastreabilidade da cadeia produtiva e análise da qualidade (SSA, 2016).

3.2 Organograma de Empresa

O organograma apresentado abaixo apresenta, de forma especificada, a principal divisão da fabrica de rações. As áreas apresentadas são responsáveis por determinadas funções dentro da empresa, mas a principal é a área da operação que é responsável pelo funcionamento da outras áreas.

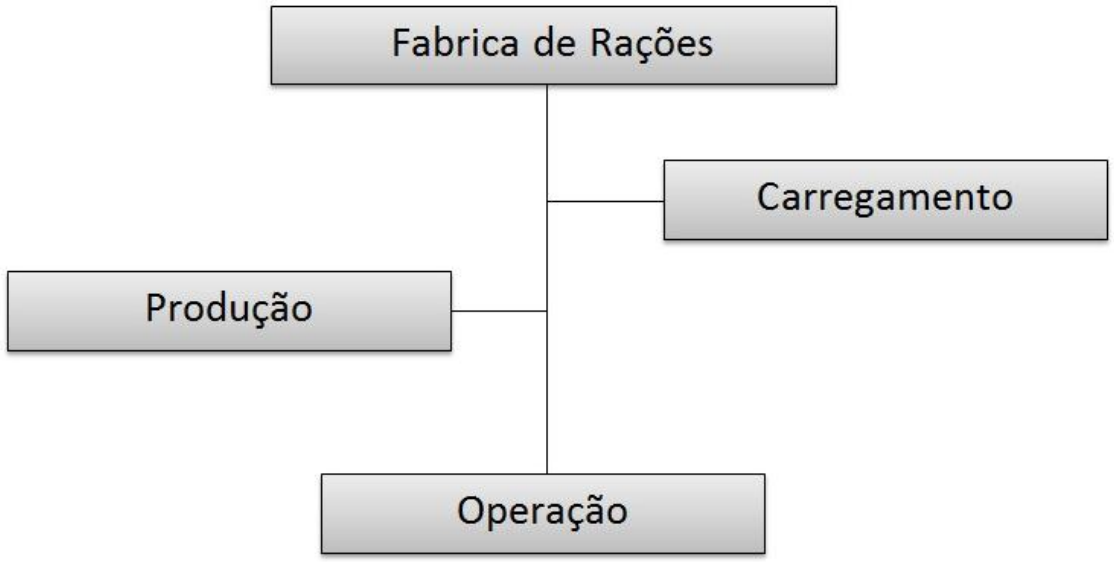

Figura 1: Organograma da fábrica de Rações da SSA

Fonte: Acervo do autor

3.3 Considerações Finais

Utilizando o estudo de caso com a aplicação da Engenharia de Software e da Engenharia de Requisitos no desenvolvimento do trabalho de conclusão de curso (TCC), todos os conhecimentos adquiridos no período acadêmico serão aplicados na prática.

Este trabalho contribuiu para um melhor aprendizado de tudo o que foi transmitido através de teorias com a aplicação prática no decorrer do desenvolvimento da documentação do projeto.

4 Projeto

Este capítulo retrata o desenvolvimento do projeto, apresentando uma descrição geral e a modelagem de dados do sistema a ser desenvolvido. A metodologia apresentada será dividida por requisitos.

4.1 Lista de Requisitos Funcionais

Id	Nome	Descrição
RF-01	Realizar cadastro	Módulo do sistema onde será realizado os cadastros.
RF-02	Abrir ordem de serviço	Módulo do sistema onde será realizado o cadastro da O.S.
RF-03	Encaminhar O.S.	Módulo do sistema onde o encarregado deverá encaminhar a O.S. para o colaborador que irá executar.
RF-04	Acompanhar O.S.	Módulo do sistema onde o encarregado deverá acompanhar o andamento da O.S. podendo alterá-la de necessário.
RF-05	Fechar O.S.	Módulo do sistema onde o encarregado deverá encerrar a O.S.
RF-06	Gerar Relatórios	Consultar as atividades que estão sento executadas ou as atividades que já foram executadas.

Quadro 1: Lista de requisitos funcionais

4.1.1 Lista de Requisitos Não Funcionais

Id	Nome	Descrição
RNF-01	Compatibilidade com o sistema operacional.	O sistema será executado em plataforma Windows 7.
RNF-02	Interação com o usuário	O sistema deverá ser de fácil manuseio, para facilitar ao usuário a utilização dos recursos funcionais.

Quadro 2: Lista de requisitos não funcionais

4.2 Diagrama de Caso de Uso

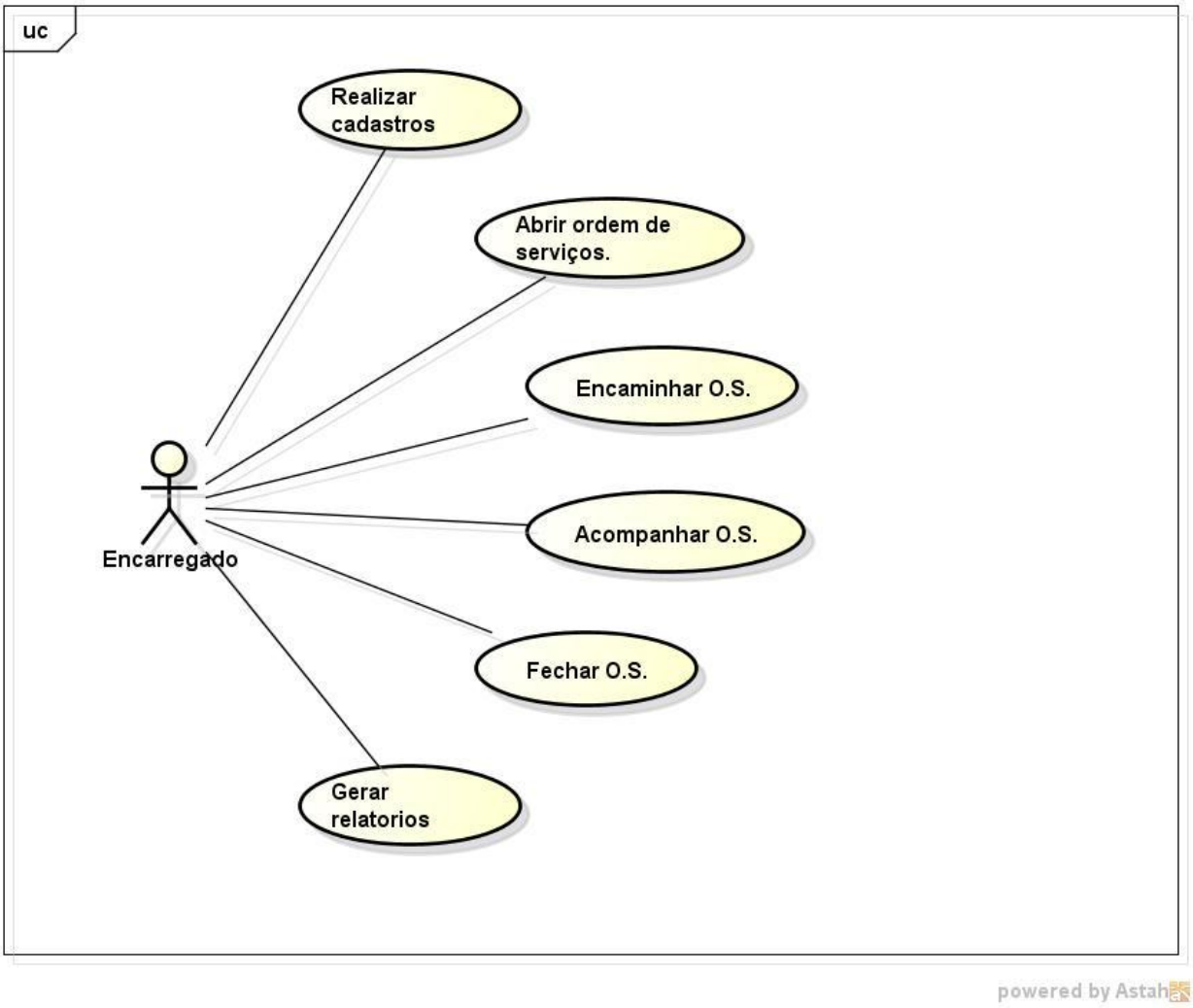

Figura 2: Diagrama de caso de uso

Caso de Uso: UC01 Realizar Cadastros	
Ator principal	Encarregado.
Atores secundários	Não há.
Resumo	Esse caso de uso descreve as etapas percorridas pelo encarregado para inserir, alterar ou excluir os cadastros realizados.
Pré-Condição	Deve estar logado no sistema para alterar os dados. Para alterar ou excluir, uma busca deve ser realizada com sucesso.

Fluxo Principal	
Ações do ator	**Ações do Sistema**
1.1 O encarregado solicita o cadastro ou alteração dos dados.	1.2 O sistema exibe o formulário para preenchimento ou alteração.
2.1 O encarregado insere ou altera os campos desejados e clica em inserir ou alterar.	2.2 O sistema valida as informações.
	2.3 O sistema exibe a mensagem (MSG.01) ou (MSG.02).
Fluxo alternativo	
Ações do ator	**Ações do sistema**
Não há.	Não há.
Exceção	
Ações do ator	**Ações do sistema**
Não há.	Não há.

Quadro 3: Descrição de caso de uso – Realizar cadastros

Caso de Uso: UC02 Abrir ordem de serviço	
Ator principal	Encarregado.
Atores secundários	Não há.
Resumo	Esse caso de uso descreve as etapas percorridas pelo encarregado para criar uma nova ordem de serviço.
Pré-Condição	Deve estar logado no sistema para alterar os dados.
Fluxo Principal	
Ações do ator	**Ações do Sistema**
1.1 O encarregado solicita uma nova ordem de serviço.	1.2 O sistema exibe o formulário para preenchimento.
2.1 O encarregado insere os campos desejados e clica em salvar.	2.2 O sistema valida as informações.
	2.3 O sistema exibe a mensagem (MSG.01).
Fluxo alternativo	
Ações do ator	**Ações do sistema**

Não há.	Não há.
Exceção	
Ações do ator	**Ações do sistema**
Não há.	Não há.

Quadro 4: Descrição de caso de uso – Abrir ordem de serviços

Caso de Uso: UC03 Encaminhar O.S.	
Ator principal	Encarregado.
Atores secundários	Não há.
Resumo	Esse caso de uso descreve as etapas percorridas pelo encarregado para selecionar um colaborador responsável pela O.S.
Pré-Condição	Deve estar logado no sistema para alterar os dados.
Fluxo Principal	
Ações do ator	**Ações do Sistema**
1.1 O encarregado solicita a O.S.	1.2 O sistema exibe o formulário da O.S..
2.1 O encarregado solicita o colaborador responsável pela O.S. e clica em salvar.	2.2 O sistema valida as informações.
	2.3 O sistema exibe a mensagem (MSG.01).
Fluxo alternativo	
Ações do ator	**Ações do sistema**
Não há.	Não há.
Exceção	
Ações do ator	**Ações do sistema**
Não há.	Não há.

Quadro 5: Descrição de caso de uso – Encaminhar O.S.

Caso de Uso: UC04 Acompanhar O.S.	
Ator principal	Encarregado.
Atores secundários	Não há.
Resumo	Esse caso de uso descreve as etapas

	percorridas pelo encarregado para acompanhar o andamento das O.S.
Pré-Condição	Deve estar logado no sistema para alterar os dados.
Fluxo Principal	
Ações do ator	**Ações do Sistema**
1.1 O encarregado solicita uma ordem de serviço já existente.	1.2 O sistema exibe o formulário da O.S.
2.1 O encarregado insere alguma alteração campos desejados e clica em salvar.	2.2 O sistema valida as informações.
	2.3 O sistema exibe a mensagem (MSG.01).
Fluxo alternativo	
Ações do ator	**Ações do sistema**
Não há.	Não há.
Exceção	
Ações do ator	**Ações do sistema**
Não há.	Não há.

Quadro 6: Descrição de caso de uso – Acompanhar O.S.

Caso de Uso: UC05 Fechar O.S.	
Ator principal	Encarregado.
Atores secundários	Não há.
Resumo	Esse caso de uso descreve as etapas percorridas pelo encarregado para encerrar o O.S.
Pré-Condição	Deve estar logado no sistema para alterar os dados.
Fluxo Principal	
Ações do ator	**Ações do Sistema**
1.1 O encarregado solicita uma ordem de serviço já existente.	1.2 O sistema exibe o formulário da O.S.
2.1 O encarregado encerra a O.S. e clica em salvar.	2.2 O sistema valida as informações.

	2.3 O sistema exibe a mensagem (MSG.01).
Fluxo alternativo	
Ações do ator	**Ações do sistema**
Não há.	Não há.
Exceção	
Ações do ator	**Ações do sistema**
Não há.	Não há.

Quadro 7: Descrição de caso de uso – Fechar O.S.

Caso de Uso: UC06 Gerar relatório.	
Ator principal	Encarregado.
Atores secundários	Não há.
Resumo	Esse caso de uso descreve as etapas percorridas pelo encarregado para emitir um relatório das O.S.
Pré-Condição	Deve estar logado no sistema para alterar os dados.
Fluxo Principal	
Ações do ator	**Ações do Sistema**
1.1 O encarregado solicita uma ordem de serviço já existente.	1.2 O sistema exibe o formulário da O.S.
2.1 O encarregado clica na opção gerar relatório.	2.2 O sistema valida as informações.
	2.3 O sistema exibe o relatório.
Fluxo alternativo	
Ações do ator	**Ações do sistema**
Não há.	Não há.
Exceção	
Ações do ator	**Ações do sistema**
Não há.	Não há.

Quadro 8: Descrição de caso de uso – Gerar relatório.

4.3 Requisitos de Dados

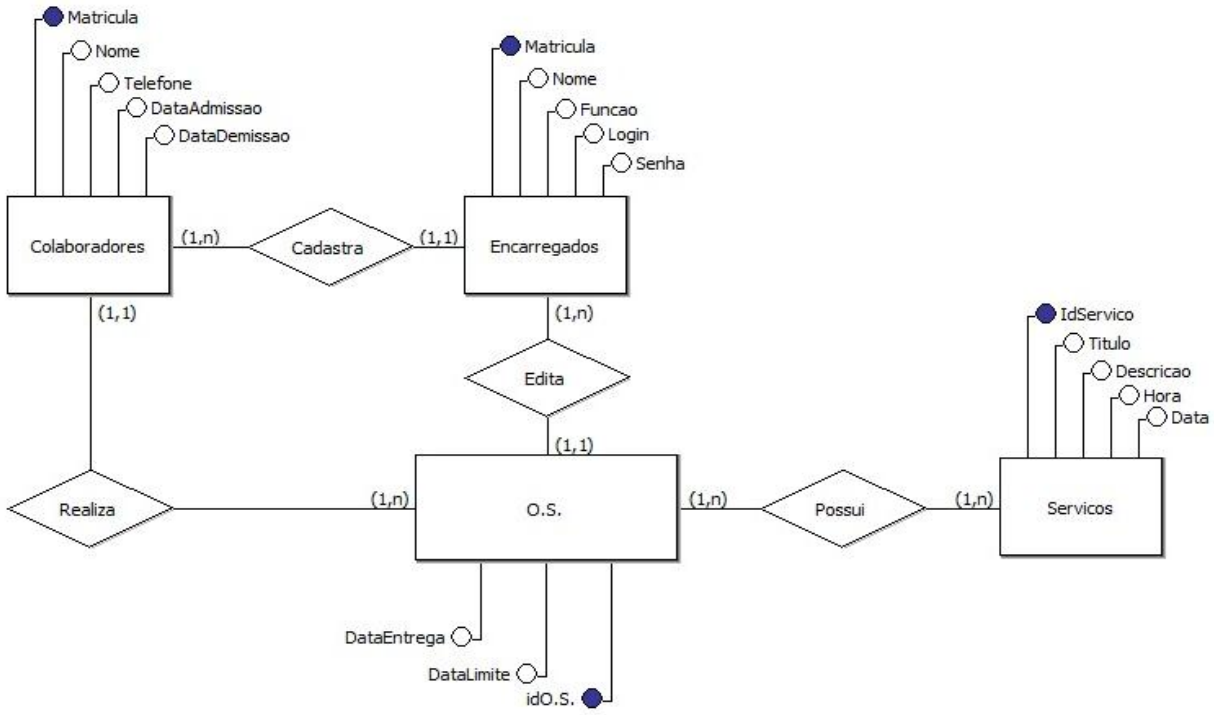

Figura 3: Diagrama MER

Entidade: Encarregados				
Atributos	**Descrição**	**Tipo**	**Tamanho**	**Observações**
matricula	Matricula de identificação do encarregado.	Int		Chave primaria, não nulo.
nome	Nome do encarregado	Varchar	45	Não nulo.
funcao	Função desempenhada pelo encarregado.	Varchar	45	Não nulo.
senha	Senha para acesso ao sistema	Varchar	8	Não nulo.
login	Nome de usuário para acesso ao sistema	Varchar	20	Não nulo.

Quadro 9: Diagrama de dados – Entidade encarregado

Entidade: Colaboradores				
Atributos	**Descrição**	**Tipo**	**Tamanho**	**Observações**
matricula	Matricula de identificação do colaborador.	Int		Chave primaria, não nulo.

Nome	Nome do colaborador.	Varchar	45	Não nulo.
Telefone	Número de telefone do colaborador	Varchar	12	Não nulo.
dataAdmissao	Data de admissão do colaborador.	Varchar	8	Não nulo.
dataDemissao	Data de demissão do colaborador.	Varchar	8	Não nulo.

Quadro 10: Diagrama de dados – Entidade Colaborador

Entidade: Servicos				
Atributos	**Descrição**	**Tipo**	**Tamanho**	**Observações**
idServico	Código de identificação do serviço.	Int	Sequencial	Chave primaria, não nulo.
titulo	Titulo do serviço.	Varchar	45	Não nulo.
descricao	Descrição do serviço que será realizado.	Varchar	45	Não nulo.
hora	Hora em que o serviço foi inserido no sistema.	Varchar	6	Não nulo.
data	Dia, mês e ano em que o serviço foi inserido no sistema.	Varchar	8	Não nulo.

Quadro 11: Diagrama de dados – Entidade Servicos

Entidade: O.S.				
Atributos	**Descrição**	**Tipo**	**Tamanho**	**Observações**
idO.S.	Código de identificação do agendamento.	Int	Sequencial	Chave primaria, não nulo.
dataEntrega	Data estabelecida para a entrega do O.S..	Varchar	8	Não nulo.
dataLimite	Data limite estabelecida para			

	a entrega da O.S.			

Quadro 12: Diagrama de dados – O.S.

4.4 Diagrama de Classes

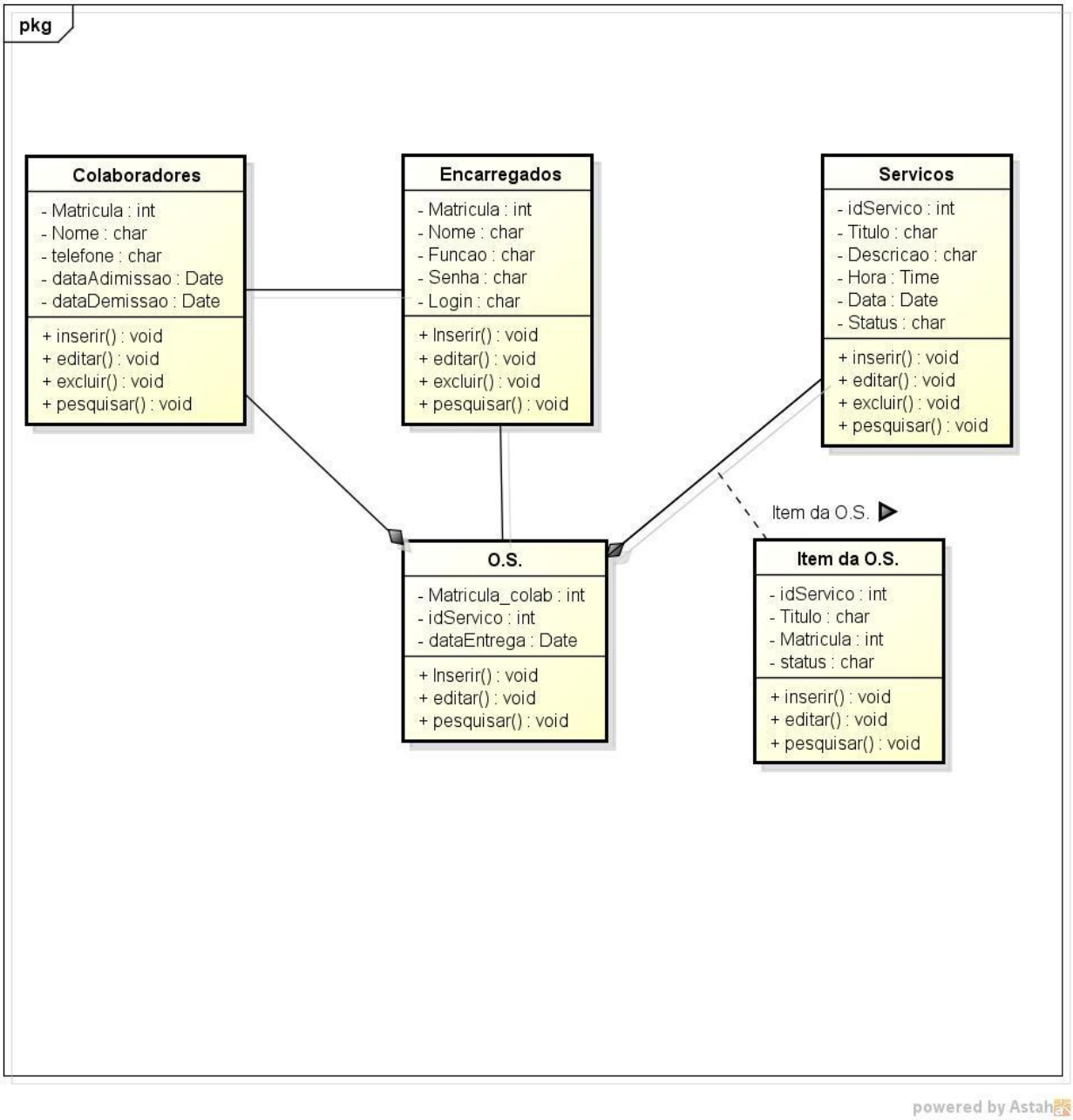

Figura 4: Diagrama de Classes

4.5 Diagrama de Sequência

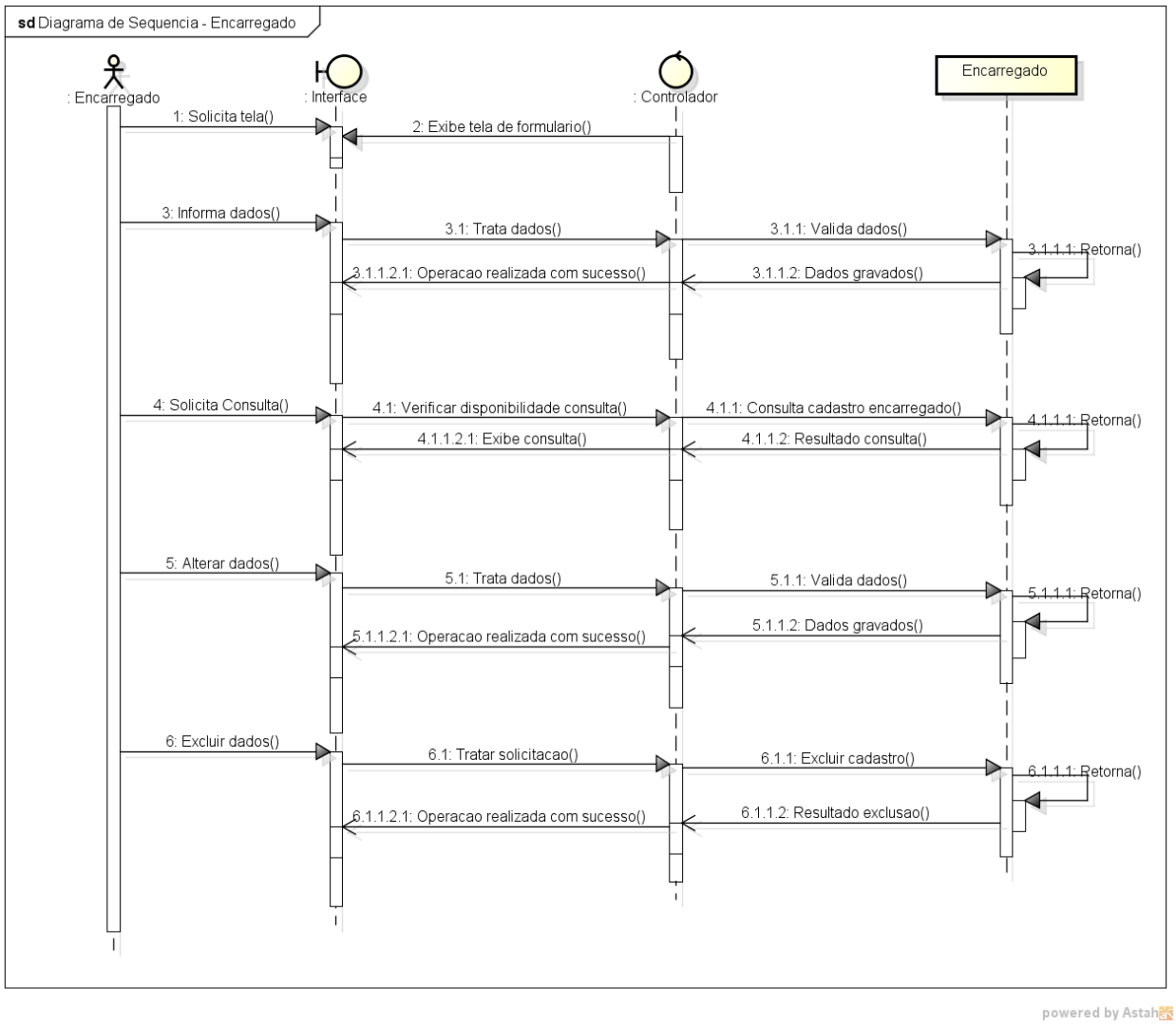

Figura 5: Diagrama de Sequência – Encarregado

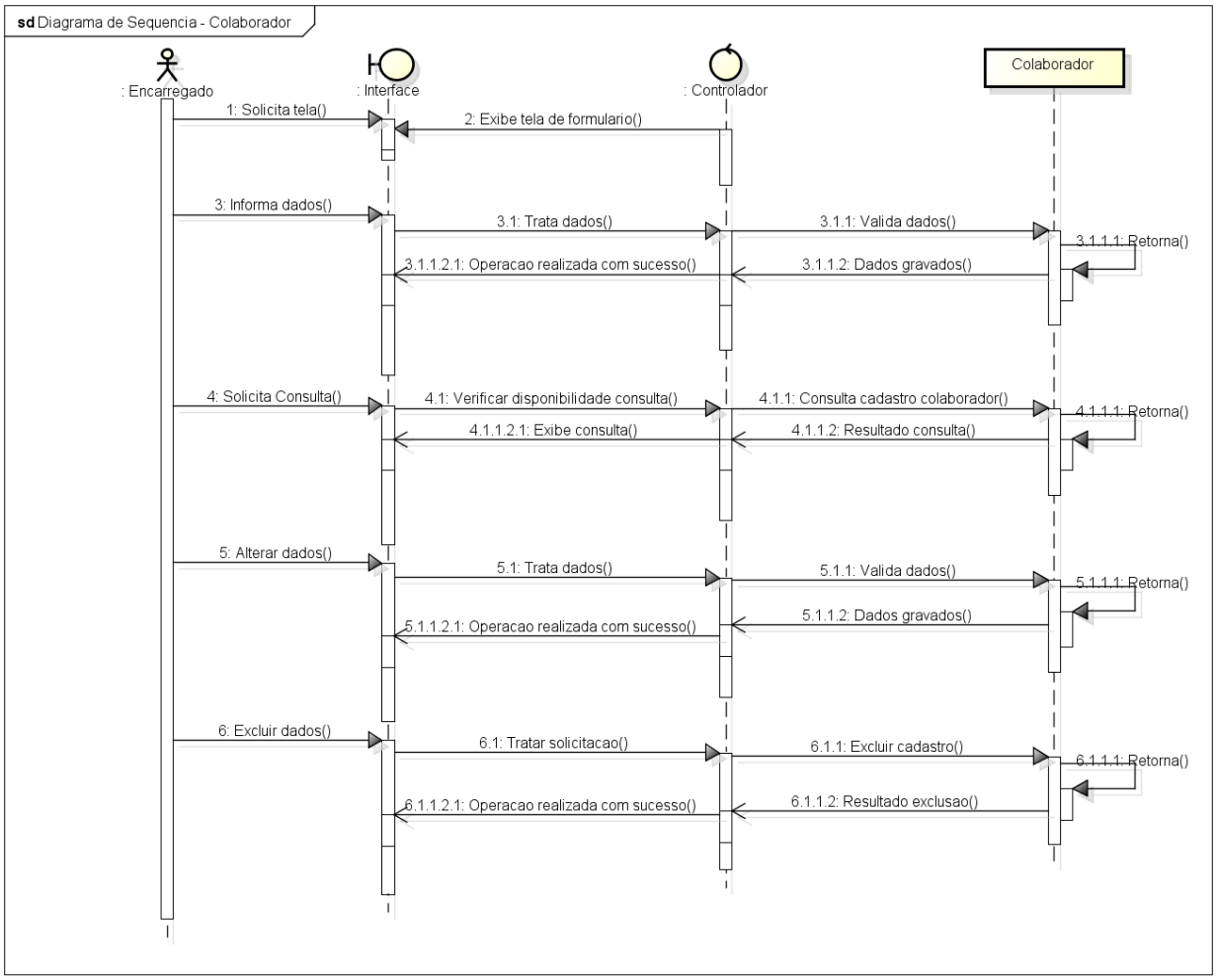

Figura 6: Diagrama de Sequência – Colaborador

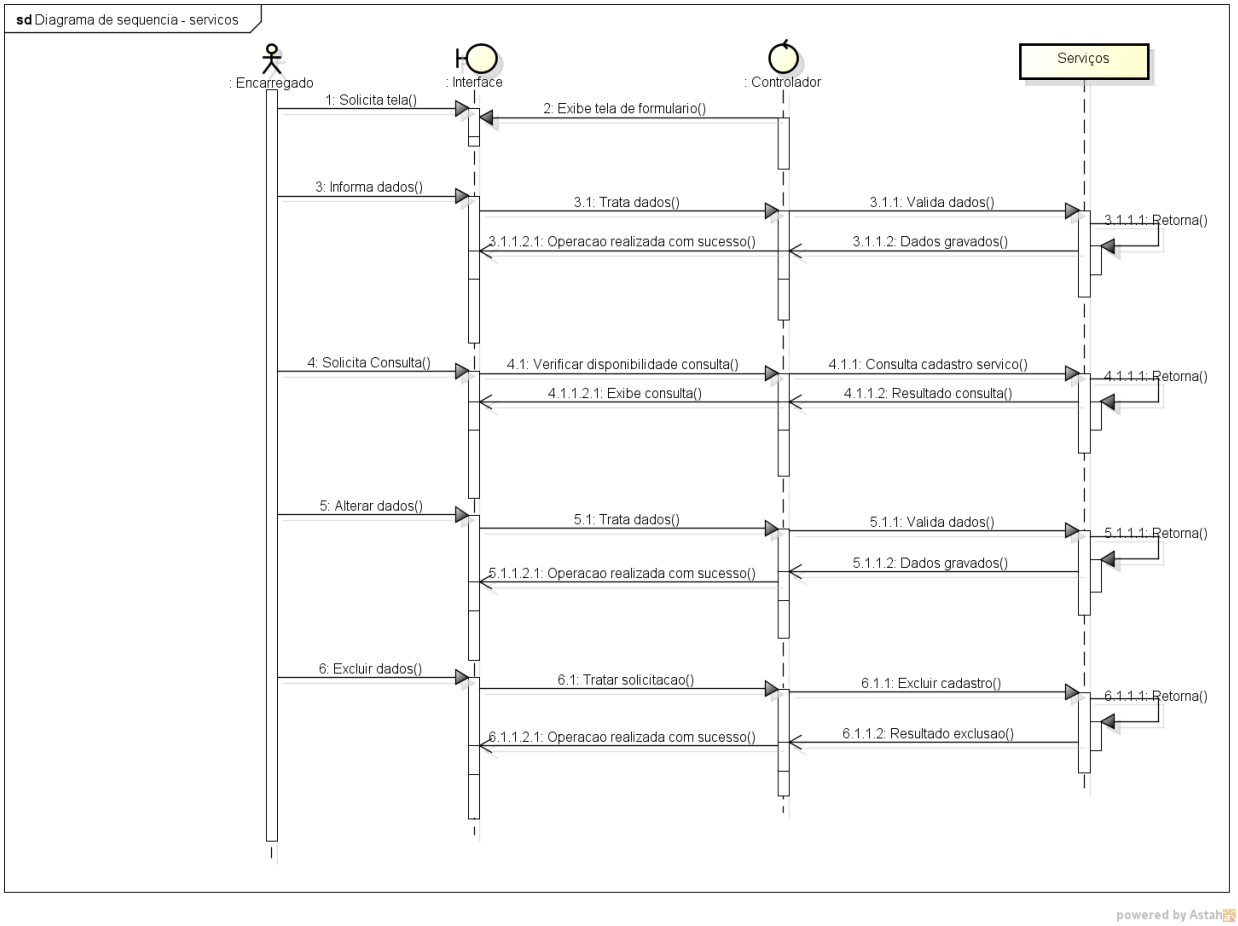

Figura 7: Diagrama de Sequência – Serviços

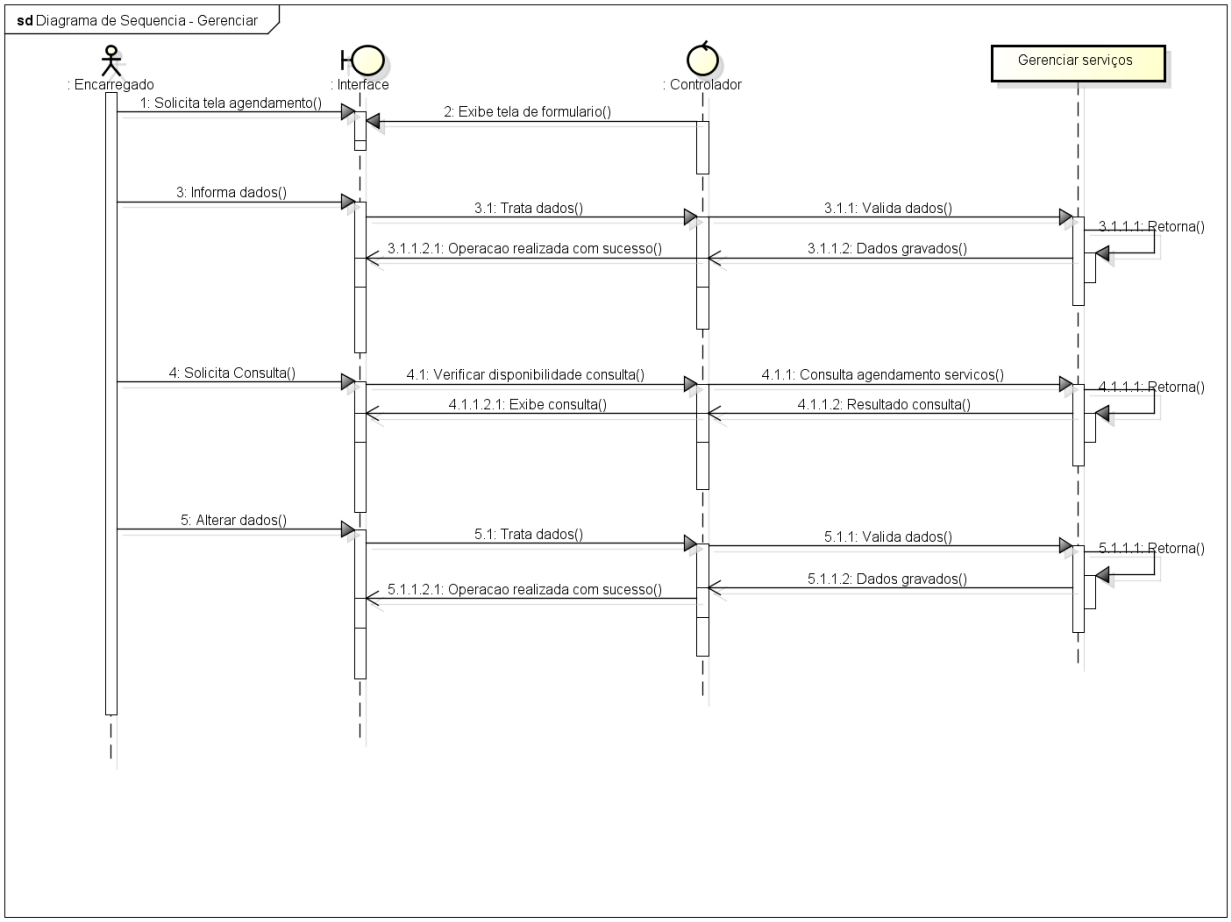

Figura 8: Diagrama de Sequência – O.S.

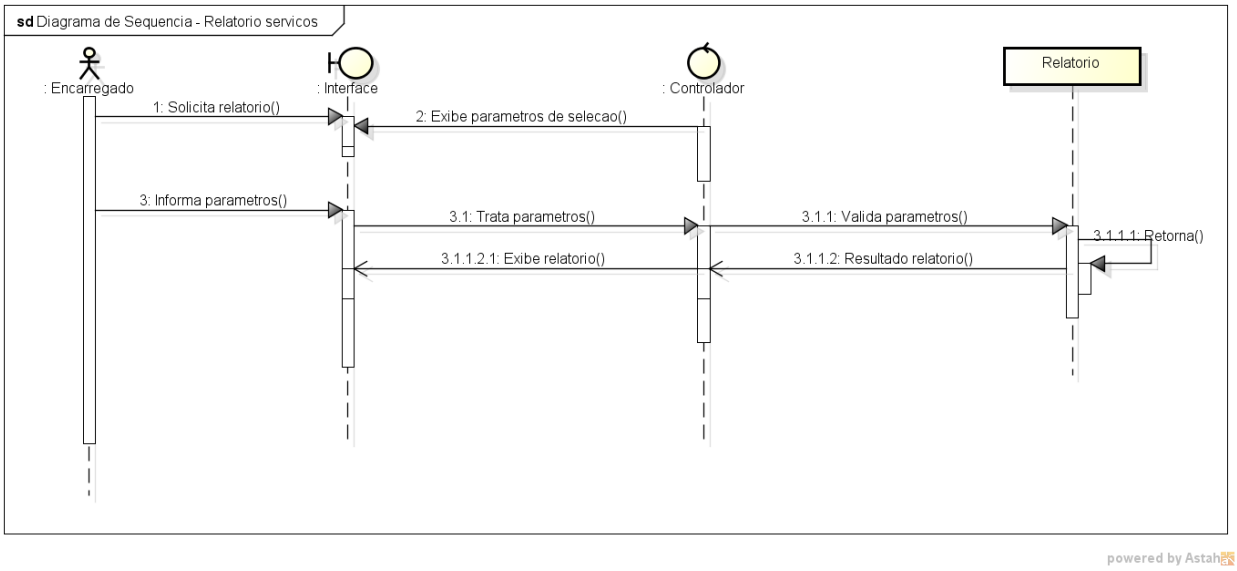

Figura 9: Diagrama de Sequência – Relatório Serviços

4.6 Metodologia e ferramentas utilizadas

Neste capitulo é descrito as principais metodologias e ferramenta de desenvolvimento a serem utilizada no desenvolvimento da documentação.

4.6.1 UML

A UML (Unified Modeling Language) que significa Linguagem de Modelagem Unificada surgiu da união de três grandes metodologias, conhecidas pelos nomes: Booch Method, de Grady Booch, OMT, de Ivar Jacobson e OOSE, de James Rumbaugh.

De acordo com Booch, Rumbauch e Jacobson (2012) a UML e uma linguagem padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software.

> A UML é apenas uma linguagem e, portanto, é somente uma parte de um método para o desenvolvimento de software. A UML é independente do processo, apesar de ser perfeitamente utilizada em processo orientado a casos de usos, centrado na arquitetura, iterativo e incremental. (BOOCH, RUMBAUGH, JACOBSON, 2012, p.14).

4.6.2 Diagramas da UML

Para Booch, Rumbauch e Jacobson (2012) um diagrama é a apresentação gráfica de um conjunto de elementos, geralmente representadas como gráficos de vértices (itens) e arcos (relacionamentos). De acordo com esses autores esses diagramas são desenhados para permitir a visualização de um sistema sob diferentes perspectivas; nesse sentido, um diagrama constitui uma projeção de um determinado sistema.

Nesta pesquisa serão descritas apenas alguns desses diagramas que são os Diagramas de Caso de Uso, Diagrama de Classes é o Diagrama de Sequência.

4.6.3 Diagrama de Caso de Uso

De acordo com De acordo com Booch, Rumbauch e Jacobson (2012) os diagramas de caso de uso são um dos diagramas disponíveis na UML para a modelagem de aspectos dinâmicos de sistemas.

O papel do diagrama de caso de uso tem por finalidade a modelagem do comportamento do sistema, de um subsistema ou de uma classe. O caso de uso é aplicado para

realizar a modelagem da visão de caso de uso do sistema. Em sua maior parte envolvendo a modelagem do conteúdo do sistema é subsistema.

> Os diagramas de casos de uso são importantes para visualizar, especificar e documentar o comportamento de um elemento. Esses diagramas fazem com que sistemas, subsistemas e classes fiquem acessíveis e compreensíveis, por apresentarem uma visão externa sobre como esses. () elementos podem ser utilizados no contexto. (BOOCH, RUMBAUGH, JACOBSON, 2012, p.263).

Para Booch, Rumbauch e Jacobson (2012) o caso de uso descreve a sequência das ações realizadas pelo sistema que proporciona resultados observáveis de valor para um determinado ator. Realizado por uma colaboração o caso de uso e representado por uma eclipse que contem linhas contínuas que, geralmente, contem apenas o seu nome.

Figura 10: Exemplo de caso de uso

Fonte: BOOCH, RUMBAUGH, JACOBSON, 2012.

O ator representa os papéis que os usuários de casos de uso representam quando se interagem com o caso de uso. Os atores de casos de uso, mesmo sendo utilizados na modelagem, não fazem parte do sistema, eles residem fora do sistema. Os atores são representador como figuras esquematizadas como mostra a figura abaixo.

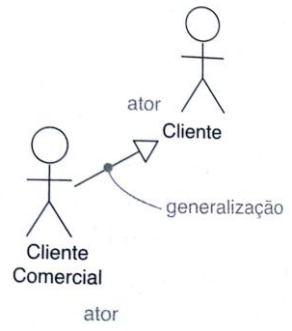

Figura 11: Exemplo de atores

Fonte: BOOCH, RUMBAUGH, JACOBSON, 2012.

O diagrama de caso de uso nada mais é que a interação entre o caso de uso é os atores e a interação do caso de uso com outro caso de uso, de modo que atribua o ator a sua função. Os relacionamentos entre casos de uso mais utilizados são a inclusão e a extensão (*include e extend*).

O relacionamento de inclusão tem como função transferir o comportamento de um caso de uso base para outro caso de uso, assim o caso de uso incluído nunca ficara sozinho. Para Booch, Rumbauch e Jacobson (2012), o relacionamento de inclusão e utilizado para evitar a repetição da escrita do fluxo de eventos.

O relacionamento de extensão tem como função a transferência de comportamento de um caso de uso base para outro caso de uso, mas o caso de uso base poderá ficar isolado. De acordo com Booch, Rumbauch e Jacobson (2012), o caso de uso base pode ser estendido somente em momentos específicos.

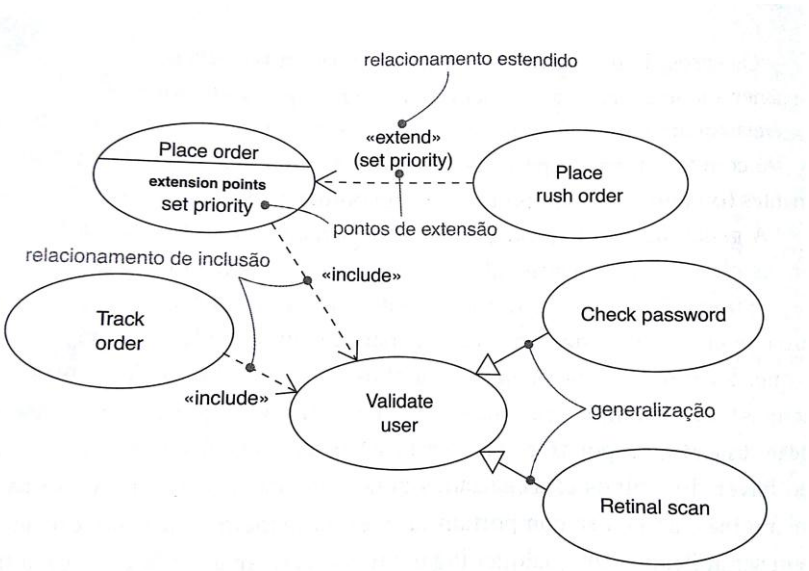

Figura 12: Exemplo de relacionamento de inclusão “includ” extensão “extend”

Fonte: BOOCH, RUMBAUGH, JACOBSON, 2012.

4.6.4 Diagrama de Classes

Diagramas de classes não elementos usados para a modelagem de sistemas orientados a objeto. São neles que mostram o conjunto de classes, interface e colaboração e os relacionamentos.

> Use os diagramas de classes para fazer a modelagem da visão estática do projeto de um sistema. A maioria dos casos, isso envolve a modelagem do

> vocabulário do sistema, a modelagem de colaborações ou a modelagem de esquemas. (BOOCH, RUMBAUGH, JACOBSON, 2012, p.115).

Nos sistemas orientados a objeto as classes são os elementos mais importantes. A classe descreve um conjunto de objetos que utilizam os mesmos atributos que são as operações, os relacionamentos e a semântica. O modelo de representação das classes e mostrado no formato de um retângulo.

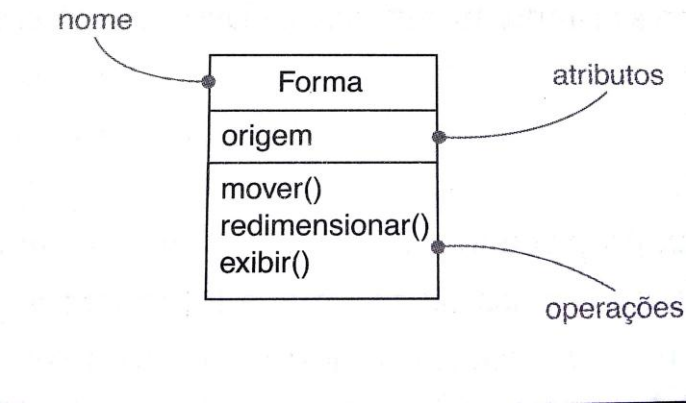

Figura 13: Exemplo de uma classe
Fonte: BOOCH, RUMBAUGH, JACOBSON, 2012.

. De acordo com Booch, Rumbauch e Jacobson (2012), cada classe deve conter um nome diferente das demais classes, o nome das classes é uma sequência de caracteres. As classes também possuem atributos que nada mais são que as propriedades do item que está sendo representado pela classe. Por exemplo, todo carro possui motor, cor e rodas, por tanto os atributos são as especificações dos itens da classe.

As operações também fazer parte de uma classe, de acordo com Booch, Rumbauch e Jacobson (2012, p.57), elas são a implementação de um serviço que pode ser solicitado por algum objeto da classe para modificar o comportamento, ou seja, uma operação é uma abstração de algo que pode ser feito com um objeto e que é compartilhado por todos os objetos dessa classe.

4.6.5 Diagrama de sequência

. De acordo com Booch, Rumbauch e Jacobson (2012, p.273), o diagrama de sequência e utilizado para a modelagem dos aspectos dinâmicos de sistemas, ele e um diagrama de interação que da ênfase à ordenação temporal das mensagens.

> Os diagramas de interação são utilizados para fazer a modelagem dos aspectos dinâmicos do sistema. Na maior parte, isso envolve a modelagem de instancias concretas ou prototípicas de classes interfaces, componentes e nós juntamente com as mensagens que são tocadas entre eles, tudo isso no contexto de um cenário que ilustra um comportamento. (BOOCH, RUMBAUGH e JACOBSON, 2012, p.273).

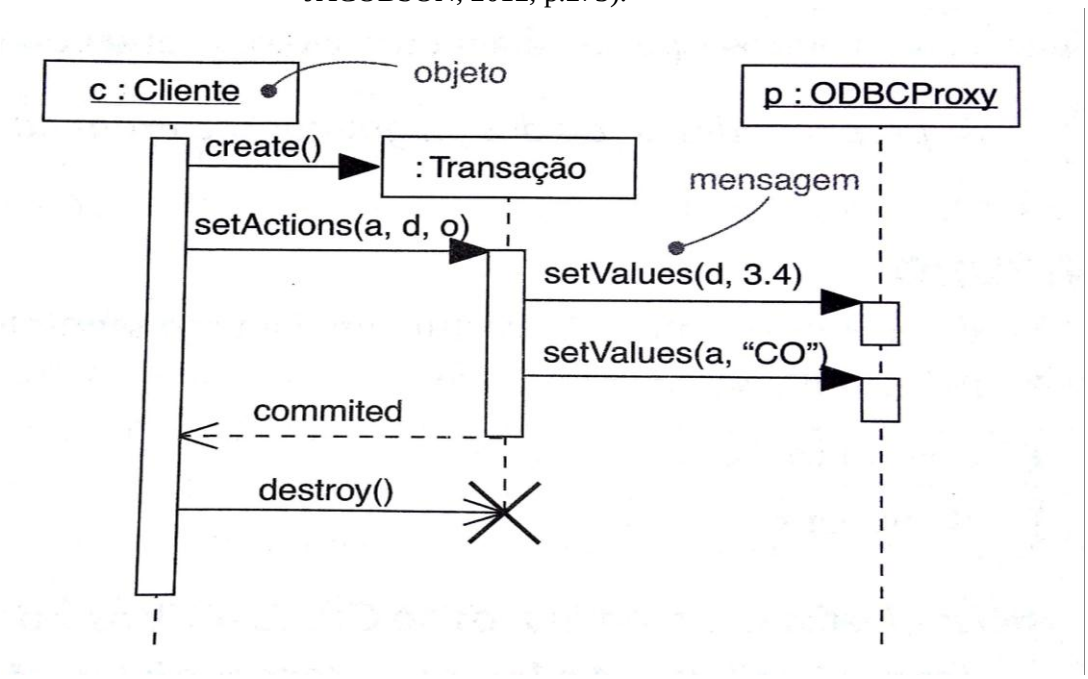

Figura 14: Exemplo de um diagrama de sequência

Fonte: BOOCH, RUMBAUGH, JACOBSON, 2012.

4.6.6 Astah

De acordo com Carneiro (2015) o Astah foi desenvolvido no Japão na plataforma java, o que garante sua portabilidade para qualquer plataforma que possuía máquina virtual java. Astah e uma ferramenta utilizada na modelagem de um sistema, assim permitindo a elaboração de modelos dos diagramas de UML.

4.6.7 Linguagem Java

A linguagem de programação Java começou a ser desenvolvido no ano de 1991 pelo grupo Green. A linguagem Java foi um projeto gerenciado pela Sun Microsystems, cujo foco principal era a criação de uma nova geração de computadores mais avançados, assim melhorando seu uso.

De acordo com JUNIOR.(2015, p.17) esta plataforma em um ambiente completo de desenvolvimento e execução de programas que reúne um conjunto ímpar de facilidades: uma linguagem completamente orientada a objeto, robusta, muito portátil, que permite operação em

rede (com destaque à internet), a distribuição de aplicações e que incorpora diversas características voltadas à segurança.

> O Java é uma linguagem de programação de propósito geral, concorrente, baseada em classes e orientada a objetos. Projetada para ser simples o bastante para que a maioria dos programadores de torne fluente na linguagem. Java em relação com C e C++, porém é organizada de forma diferente, com vários aspectos de C e C++ omitidos e algumas ideias de outras linguagens incluídas. (JUNIOR, 2015, p.17).

Atualmente a linguagem Java e dividida em três:

- **JavaME (*Java Micro Edition*):** Ferramenta utilizada nos dispositivos móveis como celulares. De acordo com Jandl Jr. (2015, p.18), essa segmento e composto por maquinas virtuais otimizadas para ambientes mais restritos.
- **JavaSE (*Java Standard Edition*)**: Integra os elementos padrão da plataforma e permite o desenvolvimento de aplicações de pequeno e médio porte. (JANDL JR, 2015, p.19).
- **JavaEE (*Java Enterprise Edition*):** Voltada para aplicações mais complexas.

> Java é uma linguagem puramente orientada a objeto e atende a todos os requisitos necessários para isso: oferece mecanismos de abstração, encapsulamento e hereditariedade. Com exceção dos seus tipos primitivos de dados, tudo em Java são classes ou instâncias de classes. (JUNIOR, 2015, p. 18).

De acordo com Junior (2015, p. 21) Java possui mecanismos de segurança em que podem, no caso de *applets*, evitar, por exemplo, qualquer operação no sistema de arquivos da máquina-alvo, minimizar riscos.

> Mesmo com tudo isso, Java ainda é uma linguagem sintática e estruturalmente simples, o que a torna uma linguagem de programação única. Exatamente por isso tornou-se a linguagem de programação orientada a objeto mais utilizada mais utilizada no mundo. (JUNIOR, 2015, p. 21).

4.6.8 NetBeans IDE

Segundo Furgeri (2013, p. 266) o NetBeans IDE (*Integrated Development Environment),* isto é, um ambiente integrado para o desenvolvimento de aplicações Java. Esse sistema permite ao usuário escrever, compilar, depurar, e instalar programas.

De acordo com Manzano (2011), a grande característica dessa IDE é o fato de ser escrita em Java, o que permite executá-la em diversas plataformas como Windows, Linux entre outros. Sua distribuição é realizada sob as condições da SPL (Sun Public License), isso ocorre por ter um código fonte aberto.

4.6.9 Linguagem SQL

De acordo com Cardoso, G. Cardoso, V. (2013) a linguagem SQL e declarativa que, no processamento importa apenas o resultado, o executável não importa. A linguagem SQL é muito flexível, ela permite a definição da estrutura dos dados, a manipulação dos dados e das regras e restrições de integridade.

> Na década de 1970, a IBM, dentro do projeto R, desenvolveu a linguagem Sequel, cujo objetivo era implementar o modelo relacional proposto por Codd, a qual evoluiu e mudou de nome para SQL. Com o sucesso e a divulgação, foi necessário padronizar essa linguagem. Isso foi feito em 1986 pelo ANSI (American National Standards Institute) e pelo ISO (International Organization for Standardization), que publicaram o padrão SQL-86. (CARDOSO, V. CARDOSO, G. 2013, p.13 e 14).

Para Cardoso, V. Cardoso, G. (2013, p. 14) para que a SQL forneça tantos recursos, é formada por várias partes, cada uma com seu propósito específico. Nessa composição da SQL temos a DDL (Data Definition Language – linguagem de definição de dados), DML (Data Manipulation Language – linguagem de manipulação de dados).

De acordo com Cardoso, V. Cardoso, G. (2012, p.122) a DDL, parte responsável pela definição de dados deve ser iniciada com a criação das tabelas do projeto. Essa parte de comandos da SQL possibilita a criação, exclusão e modificação de definições de relações ou tabelas.

Para criar tabelas no banco de dados utilizamos o comando CREATE TABLE. Para alterar a tabela o comando utilizado e o ALTER TABLE utilizado para modificar a estrutura de uma tabela. Para a exclusão dos dados e utilizado o comando DROP. Tendo definido o banco de dados, e iniciado as operações de manipulação da SQL: SELECT, INSERT, UPDATE e DELETE.

O comando SELECT e utilizado para escolher os atributos de uma tabela que será apresentado em consulta por meio da especificação do alvo. O comando INSERT acrescenta dados na tabela, o comando UPDATE é usado para modificar os dados das tabelas do banco de dados e o comando DELETE e usado para excluir os dados da tabela do banco de dados.

4.6.10 MySQL

De acordo com Milani (2006, p. 22) o MySQL é um servidor e gerenciador de banco de dados (SGBD) relacional, de licença (sendo uma delas de software livre), projetado inicialmente para trabalhar com aplicações de pequeno e médio porte, mas hoje atendendo a aplicações de grande porte e com mais vantagens do que seus concorrentes.

O surgimento do MySQL se originou na década de 90. Seu surgimento ocorreu pela necessidade de uma interface SQL que fosse compatível com o ISAM. Os responsáveis pela sua criação foram David Axmark, Allan Larsson e Michael Monty Widenius.

> A maioria dos sistemas operacionais existentes no mercado suporta a execução do MySQL. Por seu um programa escrito em C e C++, isto torna extremamente fácil a sua portabilidade entre diferentes plataformas, Dentre as principais, pode-se destacar: Linux, Unix, FreeBSD, Mac OS X Server e Windows (2000, 2003 e XP). (MILANI, 2006, p. 25).

Uma de suas características e a segurança que de acordo com Milani (2006) possui um gerenciador de conexões que trabalha com criptografia em suas senhas e com o uso de autenticação baseado em sanha, ele possui uma firewall responsável por ligar as conexões necessárias para os domínios específicos em sua linha de acesso.

4.6.11 MySQL Workbench

O MySQL Workbench é uma ferramenta visual unificada para arquitetos de banco de dados, desenvolvedores e administradores de banco de dados. Segundo Vespa (2010), a ferramenta possui opções de abrir conexão e scripts SQL, novo modelo de dados modelo de dados de um script SQL, criação de servidor, importação/exportação de base, entre outros.

Em síntese, o MySQL Workbensch fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração para configuração do servidor, de usuários, backup e muito mais. É importante destacar que a aplicação é oferecida gratuitamente e comercialmente e está disponível nas plataformas Windows, Linux e Mac OS X.

4.6.12 PhpMyAdmin

De acordo com Milani (2006, p. 85) o PhpMyAdmin é uma ferramenta de uso gráfico simples e prático para auxiliar no gerenciamento e administração do seu servidor e de suas bases de dados MySQL. Ele disponibiliza ao usuário os processos de criação de bases de dados, criando e modificando as tabelas que e um dos principais processos do MySQL.

> Deve-se ficar muito claro para o usuário que o PhpMyAdmin apenas automatiza algumas das linhas de comando mais frequentes do MySQL. Não existe nenhum recurso exclusivo do PhpMyAdmin que não possa ser feito a partir do terminal cliente do MySQL. Muito pelo contrário, nem tudo pode ser feito via terminal, poderá ser feito utilizando o PhpMyAdmin. (Milani, 2006, p. 85).

4.7.13 O Modelo Entidade-Relacionamento

De acordo com Cardoso, V. Cardoso, G. (2012, p. 27) o modelo entidade-relacionamento (MER) foi criado por Peter Chen em 1976 e trata-se de uma modelagem conceitual.

> O modelo entidade-relacionamento permite a representação da estrutura lógica do projeto com uma visão genética. Sua estrutura é feita de forma clara e simples, possibilitando representar os dados do mundo real como objetos denominados entidades ou conjunto de entidades. (CARDOSO, V. CARDOSO, G. 2012, p, 28).

- **Entidades**

Cardoso, V. Cardoso, G. (2012, p. 28) afirma que a entidade é reconhecida como conjunto, pois representa um conjunto de objetos e não um objeto individualmente, que se for preciso apresentar esse objeto individualmente, denominamos ocorrência ou instancia de entidades.

> Para a conceituação de entidade, entende-se como coisa ou objeto do mundo real que pode ser separada, distinguível de outro objeto. Nesse contexto, exemplificamos com uma entidade Livro, Carro e Pessoa como entidade concreta, mas podemos encontrar entidades abstratas como Viagens ou um Aluguel, Compra ou um Empréstimo. Mesmo com essa classificação, a representação é a mesma. (CARDOSO, V. CARDOSO, G. 2012, p. 28).

Figura 15: Exemplo de entidades.

Fonte: CARDOSO, V. CARDOSO, G. 2012.

- **Atributos**

De acordo com Cardoso, V. Cardoso, G. (2012, p. 28) o modelo de entidade-relacionamento (MER) fornece a opção de descrever a entidade, ou seja, ela não fica representada somente com um retângulo, é possível colocar suas qualidades, que formalmente são denominadas atributos.

Figura 16: Exemplo de atributos.
Fonte: CARDOSO, V. CARDOSO, G. 2012.

Cada entidade possui seus respectivos atributos, cada atributo e ligado a entidade por uma linha reta.

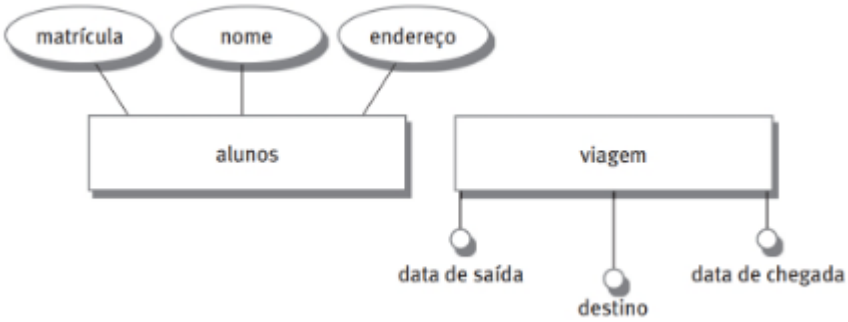

Figura 17: Exemplo das entidades e seus atributos.
Fonte: CARDOSO, V. CARDOSO, G. 2012.

- **Relacionamento**

> A fim de completar o modelo, as entidades não podem ficar isoladas, pois isso denotaria falha, uma vez que as informações estarão organizadas futuramente para o acesso de forma integrada. Para essa organização sem perda de conteúdo, as entidades devem estar associadas, ligadas entre si. No MER, não é permitido ligar uma entidade diretamente à outra. Quando há uma associação, ela é representada por um relacionamento. O relacionamento no diagrama é apresentado na forma de um losango e, para a associação entre entidades, deve-se seguir a notação básica, que são entidades ligadas ao relacionamento por linhas retas. (CARDOSO, V. CARDOSO, G. 2012,p. 32).

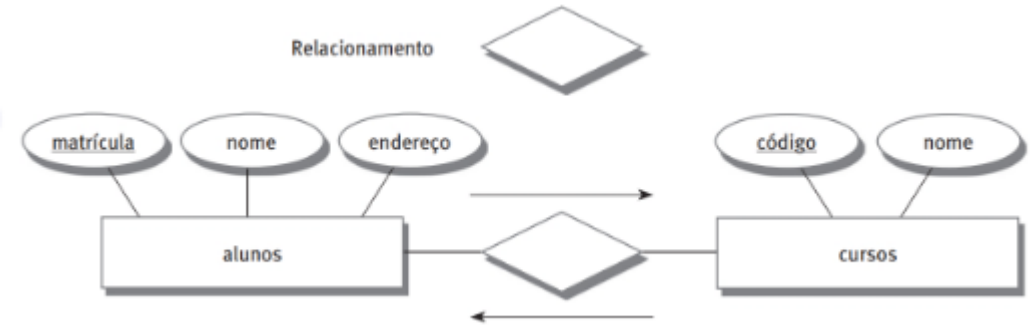

Figura 18: Exemplo de relacionamento entre entidades.
Fonte: CARDOSO, V. CARDOSO, G. 2012.

4.6.13 br Modelo

O br modelo é uma sistema para a criação de diagramas de uma base de dados. Esse sistema foi desenvolvido por Carlos Henrique Cândido que foi auxiliado pelo seu orientador professor Dr. Ronaldo dos Santos Mello (UFSC). Esse sistema foi apresentado trabalho de conclusão de curso de pós-graduação em banco de dados.

Segundo Cândido (2005), a ideia principal da aplicação é a modelagem em banco de dados relacionais. Esse sistema possibilita o desenvolvimento de um modelagem mais conceitual e também de uma modelagem lógica.

5. CONCLUSÃO

Este projeto abrange o desenvolvimento da documentação de um sistema desktop para a empresa São Salvador Alimento (SSA). A proposta para a implementação do software, tem como objetivo gerenciar as atividades entre os colaboradores, assim não permitindo uma quantidade elevada de atividades para apenas um colaborador.

Sobre a metodologia utilizada, foi realizada um levantamento bibliográfica a respeito da Engenharia de Software e Engenharia de Requisitos com base na obra de Sommerville (2007); a pesquisa de campo foi realizada no decorrer dos dias de trabalho.

A revisão bibliográfica proporcionou o embasamento necessário para a análise e intepretação dos dados do tema abordado, permitindo ainda a elaboração da documentação do projeto. Foi explicada a análise de requisitos, modelagem de dados, diagramas, banco de dados entre outros.

A dificuldade encontra foi em utilizar a ferramenta MySQL Workbench para fazer a modelagem de dados do sistema a ser desenvolvido e outra dificuldade foi a programação orientada a objeto.

REFERÊNCIAS BIBLIOGRÁFICAS

BOOCH, G.; RUMBAUCH, J.: JACOBSON I. **UML: guia do usuário**. 12ª reimpressão. Rio de Janeiro: Elsevier, 2005.

CÂNDIDO, C.H. **Aprendizagem em Banco de Dados: Implementação de Ferramenta de Modelagem E.R,** 2005. Disponível em: http://www.inf.ufsc.br/~r.mello/bdnc/Especializacao-CarlosCandido-FerramentaModelagemER-2005.pdf. Acesso em 11 nov. 2018.

CARDOSO, V.: CARDOSO, G. **Linguagem SQL: Fundamentos e práticas.** Saraiva: São Paulo 1ª Edição, 2013.

CARDOSO, G.: CARDOSO, V. **Sistema de Banco de Dados.** Saraiva: São Paulo 1ª Edição, 2012.

CARNEIRO, B.S. **O que é o Astah?,** 2015. Disponível em: <https://www.startupsstars.com/2015/10/o-que-e-o-astah-posttecnico-por-bruno-seabra/>. Acesso em 10 de nov. 2018.

FURGERI, Sérgio. **Java 7: Ensino Didático.** Érica: São Paulo 2ª Edição, 2013.

JUNIOR, Peter Jandl. **Java: guia do programador.** Novatec: São Paulo 3ª Edição, 2015.

MANZANO, J. A. N. G.; COSTA JUNIOR, R. A.: **Java 7: programação de computador: guia prático de introdução, orientação e desenvolvimento.** Érica: São Paulo 1ª Edição, 2011.

MILANI, André. **MySQL: guia do programador.** Novatec: São Paulo 1ª Edição, 2006.

SOMMERVILLE, I. **Engenharia de Software**. 8. Ed. São Paulo: Pearson Addison Wesley, 2007.

VESPA. T.G. **MySQL Workbench,** 2010. Disponível em: <https://thiagovespa.com.br/blog/2010/09/18/mysql-workbench/>. Acesso em 11 nov. 2018.

APÊNDICE – GLOSSÁRIO DE MENSAGENS

Glossário de Mensagens do Sistema		
Código da MSG	**Descrição da mensagem**	**Botão MSG**
(MSG.01)	**Inserido com Sucesso.**	**OK**
(MSG.02)	**Alterado do Sucesso.**	**OK**
(MSG.03)	**Deseja aditar o usuário?**	**SIM/NÃO/CANCELAR**
(MSG.04)	**Excluído com sucesso.**	**OK**
(MSG.05)	**Deseja excluir o item?**	**SIM/NÃO/CANCELAR**

APÊNDICE B – PROTÓTIPO

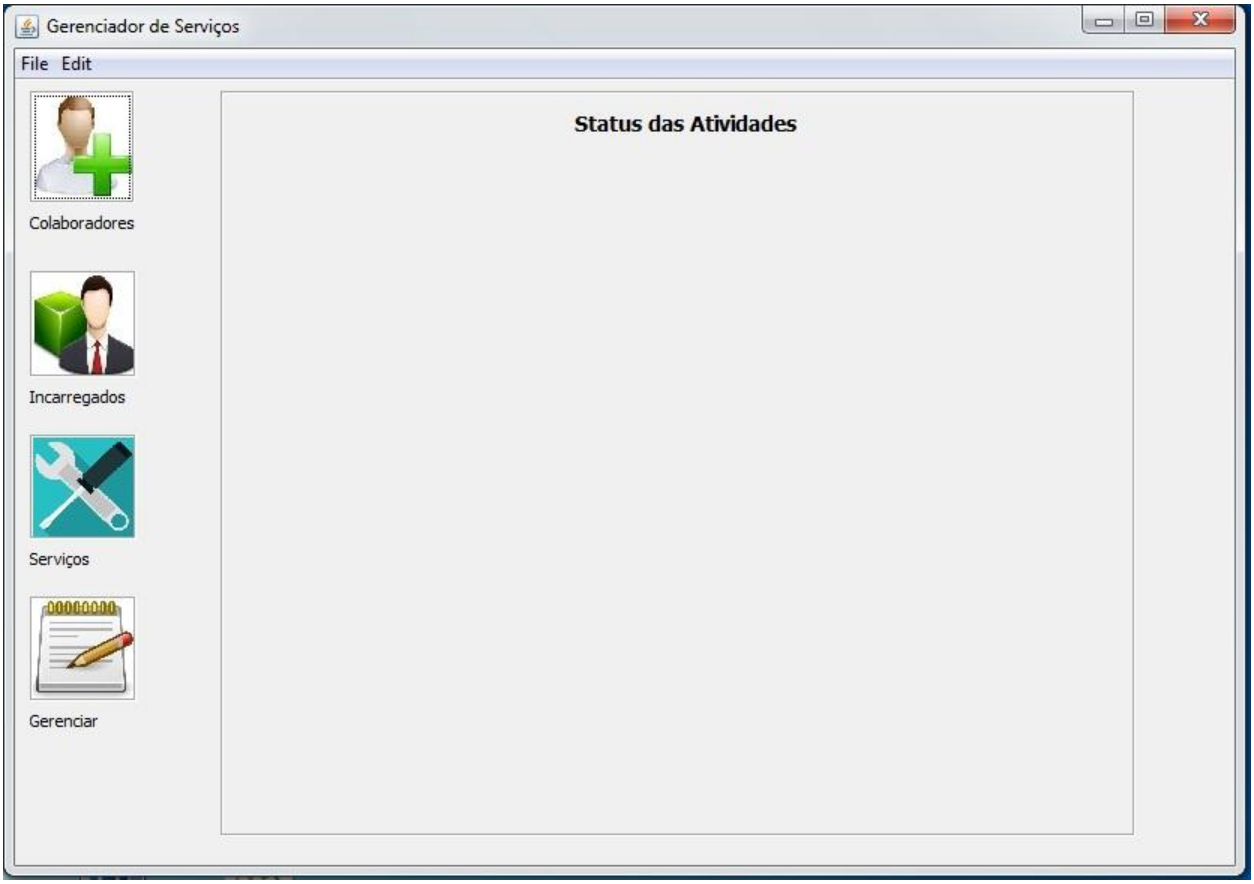

Figura 19: Tela Inicial do sistema
Fonte: Protótipo do sistema

Figura 20: Tela de cadastro do encarregado
Fonte: Protótipo do sistema

Figura 21: Tela de cadastro dos serviços.
Fonte: Protótipo do sistema.

Figura 22: Tela cadastro colaborador.
Fonte: Protótipo do sistema.

Figura 23: Tela de gerenciamento do sistema.
Fonte: Protótipo do sistema.

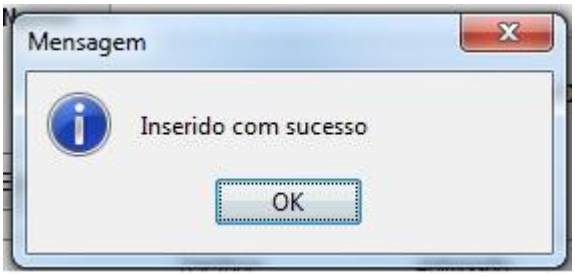

Figura 24: Tela de mensagem do sistema (MSG.01).
Fonte: Protótipo do sistema.

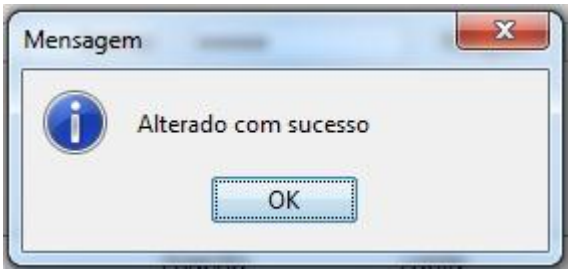

Figura 25: Tela de mensagem do sistema (MSG.02).
Fonte: Protótipo do sistema.

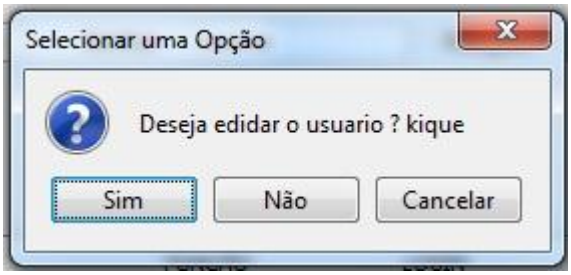

Figura 26: Tela de mensagem do sistema (MSG.03).
Fonte: Protótipo do sistema.

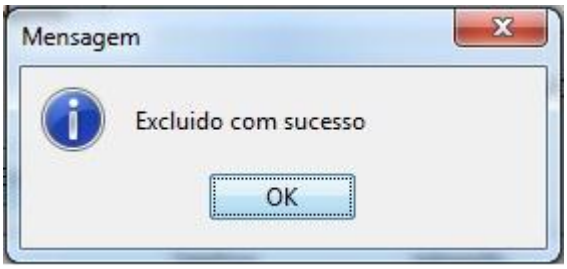

Figura 27: Tela de mensagem do sistema (MSG.04).
Fonte: Protótipo do sistema.

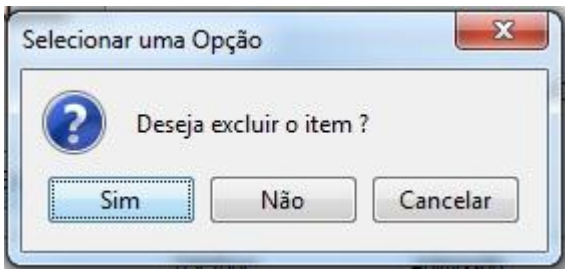

Figura 28: Tela de mensagem do sistema (MSG.05).
Fonte: Protótipo do sistema.