INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Campus São João da Boa Vista

Trabalho Final de Curso

4º ano – Curso Técnico em Informática

Prof. Breno Lisi Romano e Prof. Luiz Angelo Valota Francisco

PROCESSO DE ELABORAÇÃO DE CASOS DE TESTE DO MÓDULO DE CONTROLE ADMINISTRATIVO DO PROJETO GERAÇÕES

Geovanna Gabrício Pessoa

Bv1620053

São João da Boa Vista – SP

2019

Resumo

Devido à evidente necessidade da informatização das Instituições de Longa Permanência para auxiliar na administração do local, elaborou-se o *software* Gerações. Acoplado ao desenvolvimento dos códigos desse *software*, existe o desenvolvimento dos casos de teste, atividade a qual possui suma importância perante o Projeto, pois que garante a qualidade do referido, corrigindo erros e conferindo as suas funcionalidades. Esse trabalho tem como objetivo apresentar a atividade de casos de teste na concepção do módulo de Controle Administrativo do Projeto Gerações – módulo responsável por desenvolver o planejamento semanal de limpeza da instituição e, além disso, gerenciar o controle financeiro – expondo não somente a execução dessa atividade, mas também, todo o processo que levou à sua realização.

Palavras – chave: *software*, casos de teste, gerenciar.

Sumário

1 Introdução 7
1.1 Contextualização/Motivação 7
1.2 Objetivo Geral da Pesquisa 9
1.3 Objetivos Específicos 9
1.4 Estrutura do Documento 9
2 Desenvolvimento 11
2.1 Levantamento Bibliográfico 11
2.1.1 Engenharia de *Software* 11
2.1.2 Ciclo de desenvolvimento de *software* 11
2.1.2.1 Modelo cascata (Waterfall) 11
2.1.3 Teste de *software* 13
2.1.3.1 Caso de teste 13
2.1.3.2 Tipos de Teste e Teste de Caixa-Preta 13
2.1.4 Requisitos 14
2.1.4.1 Levantamento de requisitos de um *software* 14
2.1.4.2 Qualidade de requisitos 14
2.1.4.3 Especificação da análise 15
2.1.4.4 Tipos de requisitos 16
2.1.5 A influência dos requisitos no caso de uso 16
2.1.6 Casos de uso 17
2.2 Estrutura do Desenvolvimento 18
2.2.1 Requisitos no Controle Administrativo 18
2.2.2 Diagrama de Casos de Uso no Controle Administrativo 21
2.2.3 Casos de Teste no Controle Administrivo 22
3. Conclusões e Recomendações 27
3 Referências Bibliográficas 29

Índice de Figuras

Figura 1 - Descrição dos módulos 1 ao 6, mostrando suas principais funções 8
Figura 2 – Descrição dos módulos 7 ao 9, mostrando suas principais funções 8
Figura 3 - Processo de Desenvolvimento de *Software*: Modelo Cascata 12
Figura 4 - Ilustração representando o impacto de erros na análise de requisitos 16
Figura 5 - Exemplo simples de Diagrama de Caso de Uso 17
Figura 6 - Diagrama de Casos de Uso do módulo de Controle Administrativo 22

Índice de Tabelas

Tabela 1: Documentação para o Diagrama de Caso de Uso referente à figura (5) 17
Tabela 2: Documentação do Levantamento de Requisitos.....18
Tabela 3: Criar um planejamento semanal de limpeza da instituição.....22
Tabela 4: Marcar como realizada uma tarefa do planejamento semanal de limpeza na instituição..23
Tabela 5: Exibir ficha do funcionário.....23
Tabela 6: Criar uma despesa.....23
Tabela 7: Inserir dados do pagamento do paciente.....24
Tabela 8: Exibir tabelas de fluxo de caixa.....24
Tabela 9: Resultados dos Casos de Testes.....25
Tabela 10: Descrição das falhas reportadas.....26

Lista de Abreviaturas e Siglas

IDL	Índice de Desenvolvimento Urbano para Longevidade
IFSP	Instituto Federal de Educação, Ciência e Tecnologia São Paulo
PDS	Prática de Desenvolvimento de Sistemas
RF	Requisito Funcional
RNF	Requisito Não-Funcional
CT	Caso de Teste
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
CMMI	Modelo de Capacidade e Maturidade Integrado

1 Introdução

1.1 Contextualização/Motivação

Esta seção foi elaborada com base nos conceitos das referências [1][2][3][4]

São João da Boa Vista é uma pequena cidade localizada no interior do estado de São Paulo, a qual o IDL (Índice de Desenvolvimento Urbano para Longevidade) foi bem avaliado em relação a outros municípios, indicando a cidade como um bom lugar para pessoas da terceira idade.

Devido ao crescente envelhecimento populacional observado, a cidade se preocupa cada vez mais em providenciar recursos que assegurem à terceira idade uma vida tranquila e, por essa razão, há várias localidades conhecidas como Instituições de Longa Permanência que tem o intuito de oferecer uma ambiência agradável, segura e interativa entre os idosos.

Sob esse viés, acompanhando o crescimento tecnológico da cidade, o IFSP tem como propósito promover um estudo científico que seja gratuito e profissionalizante. O Instituto de São João da Boa Vista possui diversos cursos técnicos, integrados, de graduação e pós-graduação, sendo os cursos integrados ao ensino médio Informática e Eletrônica, cursos os quais os alunos precisam desenvolver um projeto voltado à sociedade como parte de sua finalização. Tais projetos, visam não só aprimorar e colocar em prática os conhecimentos adquiridos dos alunos como também beneficiar a cidade onde se localiza o campus, criando uma conexão de interação entre o cidadão e a tecnologia.

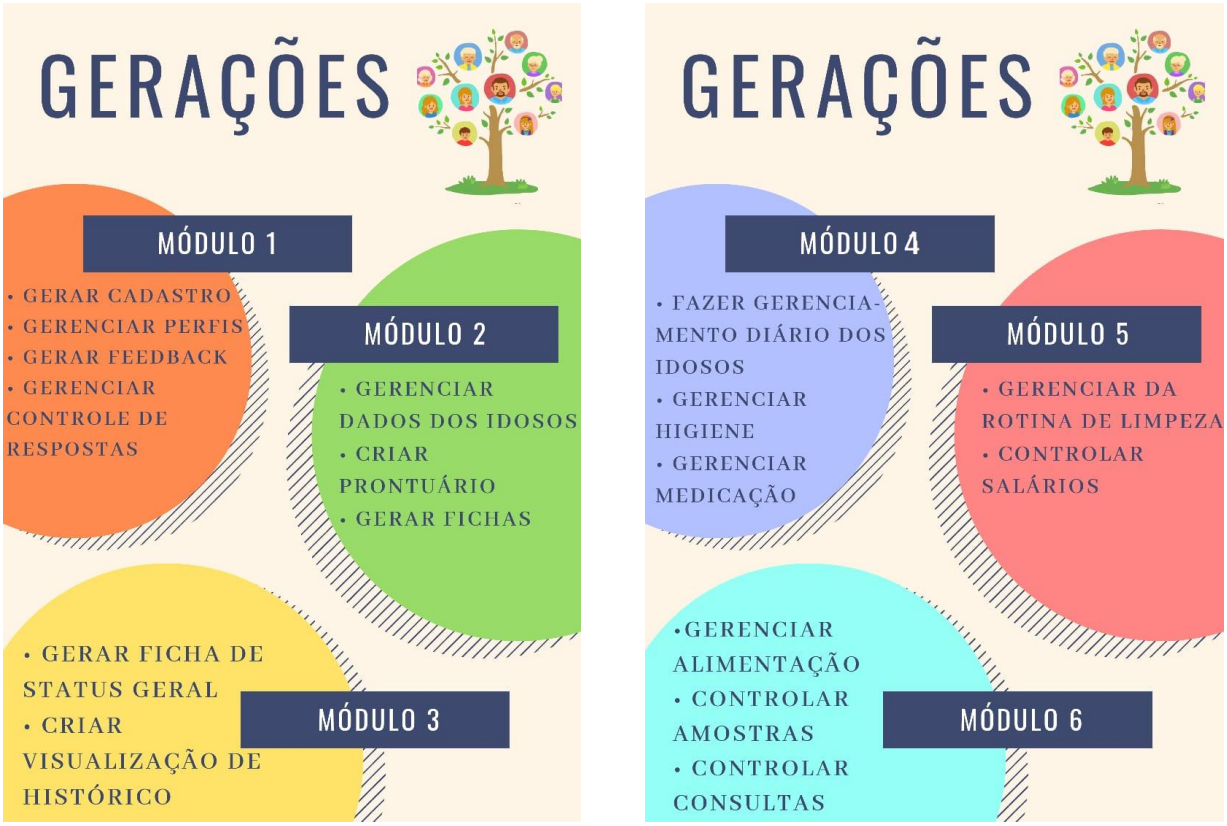
O Projeto referente às turmas de Informática da Instituição constitui a matéria de PDS (Prática de Desenvolvimento de Sistemas). Nessa, os professores responsáveis pela matéria apresentam uma proposta de sistema que será discutida em classe, dessa forma, os alunos assumem a responsabilidade de finalizar esse sistema até o fim do ano letivo.

No ano de 2019, o Projeto apresentado, denominado “Gerações”, concerne em desenvolver um sistema voltado às Instituições de Longa Permanência da cidade, buscando melhorar a qualidade e a agilidade da administração desta Instituição. O sistema referido auxiliará em todos setores da organização, desde o controle de atividades físicas até o pagamento de despesas.

À vista disso, o projeto é separado em subsistemas e estudos de casos que irão proporcionar o tema base para cada módulo. Em cada módulo, há alunos com funções diferentes, sendo elas: Analista/Testador, Desenvolvedor e Desenvolvedor de Banco de Dados.

Na imagem a seguir, é possível analisar a divisão dos módulos, assim como a descrição de cada módulo:

Figura 1 - Descrição dos módulos 1 ao 6, mostrando suas principais funções



Fonte: Elaboração própria

Figura 2 – Descrição dos módulos 7 ao 9, mostrando suas principais funções

Fonte: Elaboração própria

No subsistema de Controle Gerencial do Projeto, há o módulo de Controle Administrativo que é responsável pela administração geral da instituição pelo sistema, planejando e desenvolvendo funcionalidades referentes aos gestores e à equipe de limpeza. Dessarte, esse Trabalho de Finalização de Curso, irá adentrar na análise do módulo referido, apresentando a importância dos Casos de Testes do módulo de Controle Administrativo.

1.2 Objetivo Geral da Pesquisa

O objetivo geral da seguinte pesquisa é apresentar a importância do desenvolvimento analítico da atividade de Casos de Teste no módulo de Controle Administrativo, além colocar em pauta o processo que levou a isso. A atividade referente aos Casos de Teste, consiste em testar as iterações desenvolvidas pelos desenvolvedores do módulo, reportando os erros encontrados e, além disso, expondo as descrições deles.

Essa atividade perdura durante quase 3 bimestres, desde a documentação até a finalização das verificações. A pesquisa acompanhará esse processo até o término do ano letivo, com a finalização do Projeto.

1.3 Objetivos Específicos

- Apresentar os conceitos teóricos de *Software* e de Casos de Testes;
- Elaborar os principais requisitos do módulo de Controle Administrativo;
- Diagramar os Casos de Uso do módulo de Controle Administrativo;
- Projetar os Casos de Teste do módulo de Controle Administrativo;
- Executar os Casos de Teste do módulo de Controle Administrativo.

1.4 Estrutura do Documento

No Capítulo 1, é apresentada a introdução da pesquisa realizada, expondo o contexto no qual envolve o Projeto, desde a cidade onde é localizada a Instituição responsável pelo desenvolvimento do *software*, até a estruturação do tema especificado. Neste caso, é contextualizado ao leitor, que o Projeto Gerações o qual é desenvolvido no ano de 2019 pelos alunos do último ano de Informática do IFSP, servirá como auxilio para a administração geral das Instituições de Longa Permanência de São João da Boa Vista e região. A partir deste *software*, será possível que os responsáveis pelos pacientes acompanhem as atividades físicas de seus dependentes, as dietas aplicadas a eles, dentre

outras funcionalidades. O Projeto possui um intuito social que mobilizou os alunos para desenvolverem o *software* referido com o maior nível de qualidade possível.

O capítulo seguinte é referente ao desenvolvimento do tema da pesquisa. Para tal, é necessário explicar os conceitos em modelo de linha do tempo, analisando primeiramente os conceitos primordiais para então prosseguir até o desenvolvimento da atividade dos Casos de Teste. Dessa forma, é possível compreender melhor o trabalho realizado durante todo o ano pelos educandos e, com mais precisão, o trabalho realizado por um Analista de Sistemas.

O capítulo 3 é referente as considerações finais da pesquisa, retomando aos conceitos apresentados no decorrer da referida.

2 Desenvolvimento

2.1 Levantamento Bibliográfico

2.1.1 Engenharia de *Software*

Esta seção foi elaborada com base nos conceitos das referências [5][6][7]

A definição de Engenharia de *Software* abrange um conjunto de três elementos fundamentais, sendo eles: métodos, ferramentas e procedimentos. Tais fundamentos têm como intuito principal controlar o processo de desenvolvimento do *software* que está sendo produzido, pois dessa forma é possível oferecer ao profissional uma base para a construção do *software* de alta qualidade produtivamente.

Esse procedimento pode ser visualizado de forma semelhante em outros contextos. Como exemplo, para o processo de criação de uma propriedade, é necessária a criação e aprovação de uma planta antes de dar início a qualquer desenvolvimento prático. Análogo a esse processo, o profissional da Engenharia de *Software* possui o caráter de trabalhar na construção de um instrumento que, anteriormente à sua construção prática, necessita de uma documentação para garantir sua estabilidade no decorrer de seu desenvolvimento.

A Engenharia de *Software* se preocupa com o *software* somente quando este é um produto. Portanto, nessa engenharia, são descartados *softwares* que não serão utilizados futuramente por terceiros, ou seja, *softwares* os quais não são criados a partir do interesse de um cliente físico ou jurídico.

Como qualquer outro produto, um *software* possui uma vida útil e é concebido através da percepção de suprir uma necessidade tecnológica, o referido estará sujeito a manutenções ocasionais e será retirado de operação quando chegar ao limite de sua vida útil. Em síntese, a Engenharia de *Software* irá especificar, projetar, implementar e manter um sistema.

2.1.2 Ciclo de desenvolvimento de *software*

Esta seção foi elaborada com base nos conceitos da referência [5]

É usado para a construção do Projeto, uma prototipação que tem como base fundamentos como: a natureza do projeto e da aplicação, os métodos e as ferramentas a serem usadas, os controles e os produtos que precisam ser entregues ao cliente.

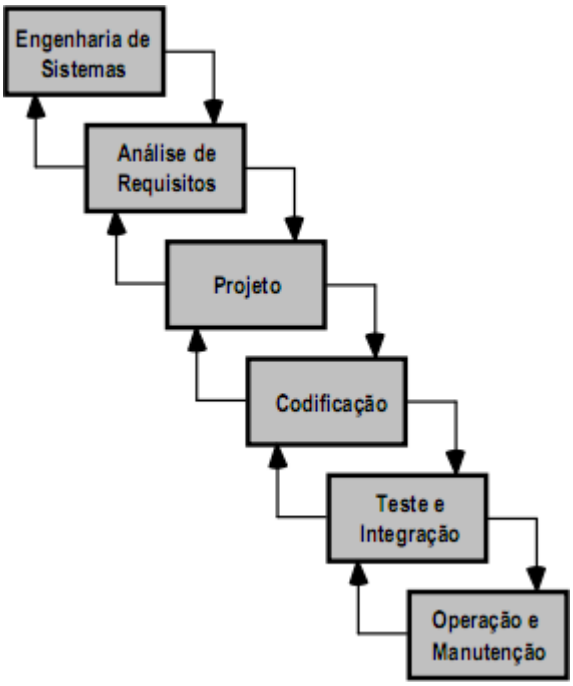
2.1.2.1 Modelo cascata (Waterfall)

Esta seção foi elaborada com base nos conceitos da referência [5]

O modelo de paradigma mais usado na Engenharia de *Software* é o Modelo Cascata, ele abrange uma sequência de ciclos que devem ser seguidos pela equipe de desenvolvimento.

Na imagem a seguir é possível observar as fases desse paradigma:

Figura 3 - Processo de Desenvolvimento de *Software*: Modelo Cascata



Fonte: https://felipelirarocha.wordpress.com/2012/04/15/diversos-modelos-de-desenvolvimento-de-software-resumo/

- Engenharia de Sistemas: Também chamado de "Estudo de viabilidade", refere a capacidade de coletar dados em nível de sistema para evitar futuras necessidades de manutenção.
- Análise de Requisitos: Essa etapa garante que o produto esteja de acordo com as requisições do cliente, pois para isso, é necessário criar uma documentação especificando o que o sistema irá realizar e como irá realizar.
- Projeto: Pode ser sintetizado em: estrutura de dados, arquitetura de *software*, detalhes procedimentais e caracterização de interface.
- Codificação: Neste momento, o projeto é transformado em código, tomando forma de programa.
- Teste e Integração: O programa deve ser testado diversas vezes em busca de erros, para garantir que qualquer defeito seja reparado antes da entrega.
- Operação e Manutenção: Etapa em que o sistema é instalado e colocado em uso. Após isso, ocorrerão manutenções no programa e essas podem ocorrer por motivos diferentes, como

para reparar erros não detectados anteriormente; para adaptar às mudanças do ambiente; ou então para adicionar características e funcionalidades com o intuito de aumentar a qualidade do produto.

As vantagens desse modelo é que ele permite controle departamental e gerencial; a desvantagem é que não há margem para muitas revisões, pois, no estágio de teste em que ocorre a busca de erros não há flexibilidade para alterar conceitos iniciais do Projeto.

2.1.3 Teste de *software*

Faz parte do Modelo Cascata (ver 2.1.2.1) de desenvolvimento de *software* a etapa de "Teste e Integração". Tal etapa é essencial para a conclusão do *software*, já que ela irá garantir que o código está sendo desenvolvido tendo como referência os critérios estabelecidos previamente na documentação do Projeto.

2.1.3.1 Caso de teste

Esta seção foi elaborada com base nos conceitos das referências [8][7]

Como é possível observar, a construção de um *software* depende das habilidades, interpretação e execução daqueles que os constroem, portanto, há sempre a possibilidade de erros no que tange as ações humanas. Caso ocorram erros e, os referidos perdurem durante o decorrer do projeto, a entrega final do *software* estará comprometida, pois estará fora dos padrões requisitados pelo cliente.

Sob essa análise, para que erros não perdurem durante o decorrer de um projeto, existe a atividade de desenvolvimento de testes de *software*. Essa atividade, de responsabilidade do analista, não acontece somente em uma etapa do desenvolvimento, mas sim, sempre que houver atualizações no desenvolvimento. Consiste em executar uma série de testes e, dessa forma, garantir que o *software* está sendo desenvolvido de acordo com os critérios documentados anteriormente.

Em outras palavras, o Teste é uma série de críticas voltadas ao sistema. É conveniente que tais críticas sejam realizadas pois, dessa maneira, além de complementar revisões realizadas, é capaz de aferir o nível de qualidade do produto em questão.

2.1.3.2 Tipos de Teste e Teste de Caixa-Preta

Esta seção foi elaborada com base nos conceitos da referência [9]

Existem diversos tipos de testes diferentes, cada qual com suas vantagens e limitações, portanto, para escolher o tipo de teste que será realizado em cada *software*, a equipe de desenvolvimento deverá analisar qual é mais conveniente dentro de suas necessidades.

No Projeto Gerações, o tipo de teste aderido foi o de caixa-preta, assim, essa pesquisa se aprofundará no desenvolvimento do referido. É considerado relativamente o teste mais simples, pois usa o mesmo ponto de vista do usuário ao invés de lidar com operações de programas.

2.1.4 Requisitos

Esta seção foi elaborada com base nos conceitos da referência [7]

Segundo os padrões IEEE e a terminologia CMMI, as seguintes definições de requisitos podem ser usadas para que estes sejam aplicáveis:

- "Condição ou potencialidade de que um usuário necessita para resolver um problema ou atingir um objetivo";
- "Condição ou potencialidade que um sistema, componente ou produto deve possuir para que seja aceito (isto é, satisfaça a um contrato, padrão, especificação ou outro documento formalmente imposto)";
- "Expressão documentada dessa característica".

2.1.4.1 Levantamento de requisitos de um *software*

Esta seção foi elaborada com base nos conceitos da referência [10]

O levantamento de requisitos deve ocorrer logo no início do desenvolvimento do *software*, afinal, os requisitos possuem papel fundamental para o andamento das demais atividades.

Para a formulação da especificação dos requisitos, é necessário o entendimento de todas as partes envolvidas do desenvolvimento (cliente, analista, desenvolvedor...), para que dessa forma, todos trabalhem de forma equivalente e não ocorram desentendimentos futuros.

Estudos afirmam que, a qualidade da especificação dos requisitos é capaz de garantir melhor desempenho no *software* como um todo, logo, é necessário que os membros envolvidos ao processo de desenvolvimento do Projeto estejam de fato convencidos de que a análise dos requisitos foi suficiente; sob o mesmo ponto de vista da citação de Crosby, um empresário mundialmente conhecido, "*Qualidade é conformidade com os requisitos*".

2.1.4.2 Qualidade de requisitos

Esta seção foi elaborada com base nos conceitos da referência [9]

A busca e exigência por qualidade estão sempre presente do cotidiano de qualquer pessoa. Ao adquirir um produto, por exemplo, o cliente busca saber se essa mercadoria possui requisitos correspondentes às suas necessidades, caso esses requisitos sejam supridos pelo item, ele irá realizar a aquisição; caso contrário ele irá buscar por um novo. Tais requisitos, nem sempre serão voltados a

termos técnicos, pois os requisitos do cliente se basearão em sua vivência e em sua análise pessoal; é função do responsável da empresa que ele interprete esta análise.

A relação do *software* e o cliente funcionam da mesma maneira. Analogamente à frase de E. Berard, "*Caminhar sobre a água e desenvolver software a partir de uma especificação de requisitos é fácil se ambos estão congelados*."; portanto, é necessário que o cliente estabeleça previamente ao desenvolvedor suas necessidades, desejos e afins, pois, dessa forma, o primeiro passo para desenvolvimento desse *software* será que o Analista fique a par dos critérios de seu cliente e assim crie requisitos a partir de suas necessidades e vontades.

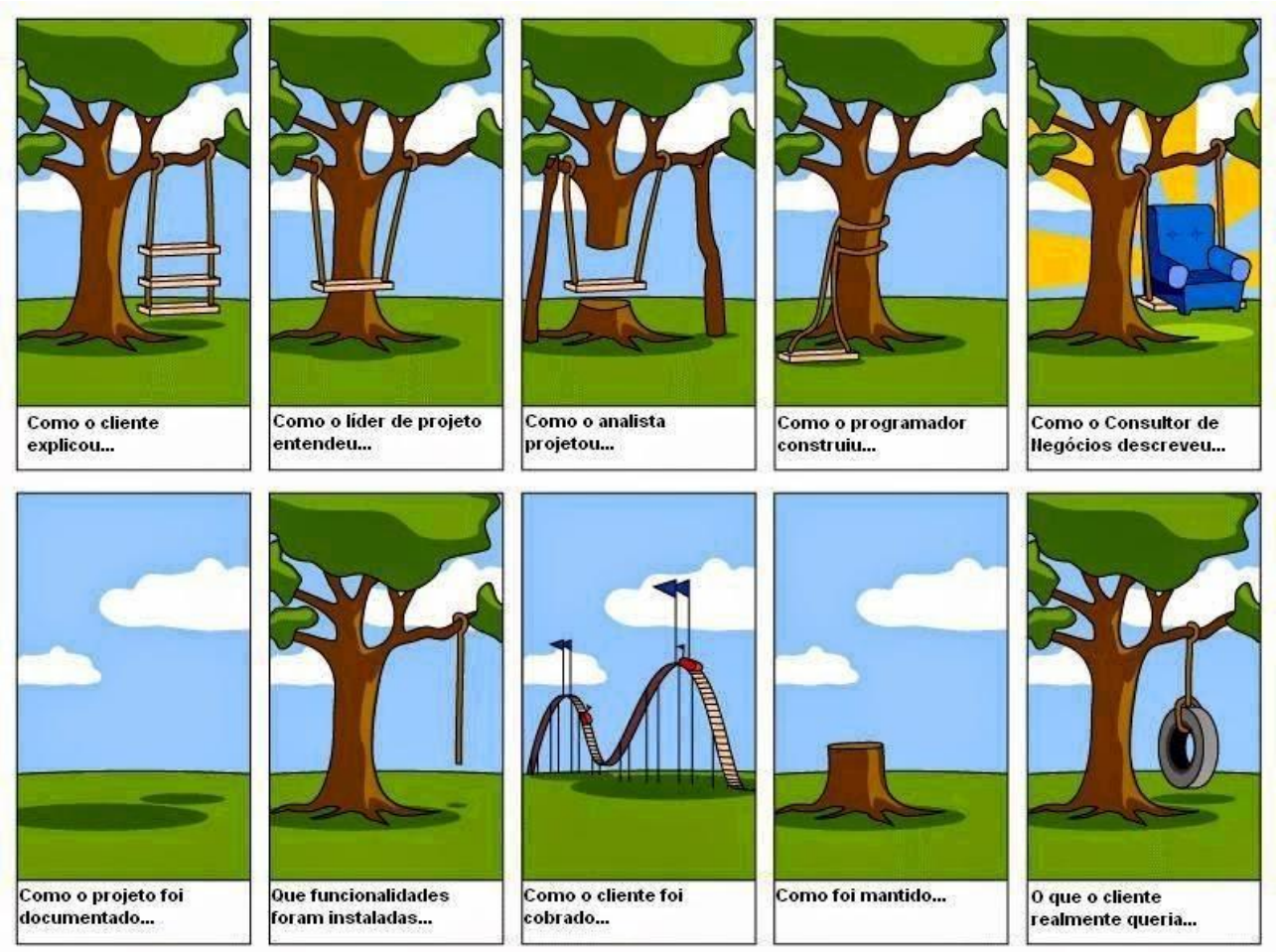
2.1.4.3 Especificação da análise

Esta seção foi elaborada com base nos conceitos da referência [9]

Para garantir a qualidade no processo do levantamento de requisitos é primordial a boa comunicação entre o Analista e o cliente, caso contrário, podem ocorrer desentendimentos que acarretarão em problemas futuros, deixando o cliente descontente com a entrega.

Uma representação clássica desse problema é a imagem a seguir, que demonstra de maneira ilustrativa modificações não requeridas que foram realizadas no decorrer do desenvolvimento do *software* e afetaram a entrega final.

Figura 4 - Ilustração representando o impacto de erros na análise de requisitos



Fonte: https://www.euax.com.br/2018/08/o-que-e-escopo-de-projeto-escopo-do-produto/

2.1.4.4 Tipos de requisitos

Esta seção foi elaborada com base nos conceitos da referência [10][9]

Sob esse viés, o levantamento dos requisitos de um *software* consiste em dividi-los em dois segmentos, sendo esses: Requisitos Funcionais (RF), os quais são referentes às funcionalidades de um *software*; e também os Requisitos Não-Funcionais (RNF), os quais envolvem custo, confiabilidade, entre outras características.

Uma vez que RF definem as funcionalidades do *software* em questão, os RNF definem as características dessas ações, uma vez que essas também terão influência dentro do sistema. Por exemplo, “A autenticação deverá ocorrer em um espaço de tempo de quatro segundos”.

2.1.5 A influência dos requisitos no caso de uso

Esta seção foi elaborada com base nos conceitos da referência [11]

Como relatado no tópico 2.1.2, os requisitos são fundamentais para o desenvolvimento de outras atividades, incluindo o Caso de Uso.

Os casos de uso não representam funcionalidades internas do sistema, mas sim funcionalidades completas para o usuário. Eles podem servir como contrato entre o cliente e o desenvolvedor já que, por possuírem uma compreensão simples e dinâmica, acabam por manter uma conexão ainda maior entre o cliente, o usuário e o desenvolvedor do sistema.

2.1.6 Casos de uso

Esta seção foi elaborada com base nos conceitos da referência [11]

Um caso de uso é um diagrama que demonstra todas as formas pelas quais o ator pode utilizar o programa. O ator pode ser qualquer usuário ou até mesmo outro sistema, porém, um nome deve ser atribuído ao autor para especificar com mais clareza sua função. Exemplo: “Autor: administrador”; “Autor: bibliotecário”.

Na imagem a seguir, é possível analisar um exemplo de caso de uso, onde é possível que a secretária agende uma consulta para um paciente:

Figura 5 - Exemplo simples de Diagrama de Caso de Uso

Fonte: Elaboração própria.

Todavia, o caso de uso se trata apenas de um diagrama visual, portanto, para analisa-lo melhor é necessário elaborar uma documentação. A documentação do caso de uso é feita de forma padronizada através de uma tabela, onde deve-se especificar algumas informações sobre os autores e relações que envolvem o diagrama e também detalhar a relação das ações de usuário/sistema.

Tabela 1: Documentação para o Diagrama de Caso de Uso referente à figura (5)

Nome do caso de uso	Marcar consulta de um paciente
Descrição	Caso de Uso que permite que a profissional com função de secretária marque uma consulta para um paciente do consultório.
Ator Envolvido	Secretária; paciente
Pré-condições	-

Fluxo Básico	-
Usuário	Sistema
1. O paciente faz uma requisição para marcar uma consulta, logo, a secretária acessa o sistema de calendário;	
	2. O sistema exibe uma tela com os dias e os horários disponíveis para marcar a consulta;
3. Após o paciente escolher a data e hora disponível desejada, a secretária seleciona o dia e marca a consulta.	
	4. O sistema cadastra a consulta no banco de dados do *software* do consultório e altera o horário selecionado para indisponível. Mais nenhuma consulta pode ser agendado nesse horário, portanto, o sistema não deve permitir sobreposições de datas/horários.
	5. Fim de caso de uso.

Fonte: Elaboração Própria

2.2 Estrutura do Desenvolvimento

2.2.1 Requisitos no Controle Administrativo

A seguir, é possível analisar a tabela do Levantamento de Requisitos do módulo de Controle Administrativo. Estes requisitos são referentes às funcionalidades inclusas no planejamento semanal e gestão financeira da instituição. É válido ressaltar que, no decorrer do Projeto, alguns requisitos sofreram alterações devido a necessidade de uma nova compreensão dos referidos, sendo esta a versão da tabela atual e resumida:

Tabela 2: Documentação do Levantamento de Requisitos

Identificador	Descrição do Requisito
RF #01	O sistema deve permitir que o representante da equipe de limpeza possa criar, editar e excluir tarefas no planejamento semanal da limpeza. Este usuário também poderá fazer uma marcação nas tarefas que já foram realizadas através de uma caixa de seleção que estará localizada logo no início da descrição. O planejamento semanal deverá contar com:

	• Nome da tarefa*: Varchar (50); • Descrição da tarefa*: Varchar (250); • Data de realização: Date-time; • Status da tarefa* (concluída ou não): Boolean (deverá indicar se está selecionado ou não). • Funcionário responsável pelo setor*: Varchar (100); • Nome do setor*: Varchar (50); • Observações: Varchar (500);
RF #02	Planejamento semanal: Deverá contar com: • Nome do setor*: Varchar (50); • Data de limpeza: Date; • Descrição de como deve ser realizada a limpeza*: Varchar (250); • Funcionário responsável pelo setor: Varchar (100). O gestor tem acesso para visualizar todos os planejamentos semanais.
RF #03	O sistema deverá notificar o gestor quando todas as alas forem selecionadas para que um novo planejamento seja feito. A mensagem deve aparecer como um pop-up na tela, quando ele acessar a aba de planejamento. Para que o gestor e o representante da limpeza saibam que existe um aviso, deverá aparecer uma bolinha de notificação na frente do nome da ala que teve o planejamento finalizado.
RF #04	O sistema deverá exibir a ficha de todos os funcionários previamente cadastrados, decorrente a sua categoria; essa ficha deverá conter: • Prontuário: Integer e autoincremento (USADO ID NOS MODELOS MER) • Nome do funcionário*: Varchar (100); • CPF* (XXX.XXX.XXX-XX): Inteiro (11), pontos e hífen automáticos; • Data de Nascimento* (DD/MM/AAAA): Date; • Salário*: Float; • Cargo*: Varchar (80); • Data de entrada na empresa* (DD/MM/AAAA): Date; • Status* (ativo ou inativo): Boolean; • Bônus salarial: Float; • Desconto salarial: Float. O sistema também deverá permitir que o gestor ordene a busca dos funcionários de forma que os inativos (quem já não trabalha mais na empresa) apareçam primeiro. O usuário também poderá ordenar a busca (select na tabela de Funcionários) por ordem alfabética (A-Z / Z-A).
RF #05	O sistema deverá conter o item ‘Orçamentos’ no menu. Ao selecionar esse item, o usuário deverá ser direcionado a uma página exclusiva para o orçamento dos funcionários. O gestor deverá criar e/ou editar os salários dos funcionários. Ele poderá adicionar um bônus aos seus salários (variável presente no RF#04) ou realizar um desconto em caso de falta não justificada. Se ele pesquisar o nome do funcionário por CPF ou Nome Completo, deverá aparecer uma ficha do funcionário com suas informações. A ficha deverá ser em formato de tabela, contendo nome*, CPF*, salário*, data de entrada na empresa*, cargo*, data de nascimento*, status* (se está trabalhando, de férias, de licença ou afastado). Sempre que forem alterados o Status, Salário, o Bônus Salarial ou o Desconto Salarial, deve ser registrado a data/hora destas alterações, pois isto impacta diretamente na geração do fluxo de caixa mensal da instituição. O histórico deve ficar acessível para pesquisa do gestor.
RF #06	O sistema deverá permitir que o gestor possa criar, editar e buscar os pagamentos mensais ocorridos na instituição (água, energia, gastos com alimentação, entre outros). Além disso, o

	gestor deverá adicionar a data de vencimento de cada pagamento, dessa forma ele poderá ser avisado caso não pague a conta dentro do prazo. A mensagem de aviso deverá aparecer como um pop-up e a bolinha de notificação deverá aparecer seguindo o mesmo esquema da notificação da tarefa descrito no RF#03. • Tabela: Contas Mensais; • Nome da conta*: Varchar (50); • Descrição: Varchar (100); • Valor*: Float; • Data de vencimento* (DD/MM/AAAA): Date; • Status* (pago ou não): Boolean. O sistema deve permitir que o gestor busque pelos pagamentos dos serviços da instituição.
RF #07	O gestor deve cadastrar o prontuário do funcionário, o sistema deverá permitir que o gestor crie, edite e busque o pagamento dos pacientes da instituição, pesquisando por prontuário ou nome completo do idoso. Ao pesquisar um paciente, deverá aparecer sua ficha completa, apresentando todas as informações que foram salvas no cadastro, incluindo o seu histórico de pagamento. O sistema deve permitir que o gestor busque pelos pagamentos de estadia dos pacientes da instituição. Quando estiver na página do planejamento, o sistema deve permitir que o gestor faça uma busca ordenada por 'Planos Semanais', 'Planos Mensais' e 'Planos Anuais', aparecendo assim os pacientes que se encaixam em cada categoria.
RF #08	O sistema deverá ter um terceiro item de menu que receberá o nome de doações, contendo o registro de todas as doações realizadas na instituição. O registro deverá conter o tipo da doação, sendo eles: doação monetária (float), alimentícia ou agasalhos. • Tabela: Doações; • Nome da doação*: Varchar (50); • Tipo* (roupa, comida, dinheiro): Varchar (50); • Descrição: Varchar (300); • Valor (em caso de doação monetária): Float;
RF #09	A quarta opção do menu retrátil deverá abrir uma aba com todas as notificações que o gestor receber, independente do fato dele ter fechado elas, deverão ficar armazenadas na opção 'Notificações' do menu retrátil, na frente deverá estar o número de notificações que o gestor possui. No canto superior direito, logo acima das notificações deve ter uma opção de "Limpar" na qual se o gestor clicar deverá excluir todas as notificações desta aba. Se o gestor não quiser excluir tudo, mas apenas algumas específicas ele tem a opção de clicar em um "X" ao lado de cada notificação. Se o gestor limpar todas as notificações o número ao lado da palavra "Notificações" deverá aparecer como "0" (zero). As notificações que serão enviadas ao gestor são referentes a contas pendentes da instituição (sendo elas contas mensais, salários de funcionários e atualização do pagamento dos idosos), conclusão do planejamento semanal.
RF #10	O sistema deverá imprimir na tela duas tabelas relacionadas ao fluxo de caixa. A primeira deverá apresentar todas as receitas da empresa e o valor do todas elas, o gestor NÃO vai digitar esses dados, o sistema irá traze-los do banco de dados. Ao final da tabela o sistema deve fazer um auto soma de todos os valores e imprimir o valor total de receitas na tela. A segunda tabela segue o mesmo esquema, apresentando todas as despesas da instituição e o valor de todas elas. Ao final da tabela o sistema deve fazer um auto soma e imprimir o valor total de despesas na tela. Os valores das despesas devem estar na cor vermelha e os valores das receitas devem estar na cor verde. Ao final das duas tabelas o sistema deve imprimir na tela o valor total de dinheiro na instituição. Se o valor for positivo a cor deve ser verde, mas se o valor for negativo a cor

	deve ser vermelha.
RF #11	O sistema deverá conter um menu retrátil, de forma que quando o usuário posicionar o mouse sobre este botão deverá aparecer um menu com 4 opções: ‘Limpeza’, ‘Orçamento’, ‘Doações’ e “Notificações”.

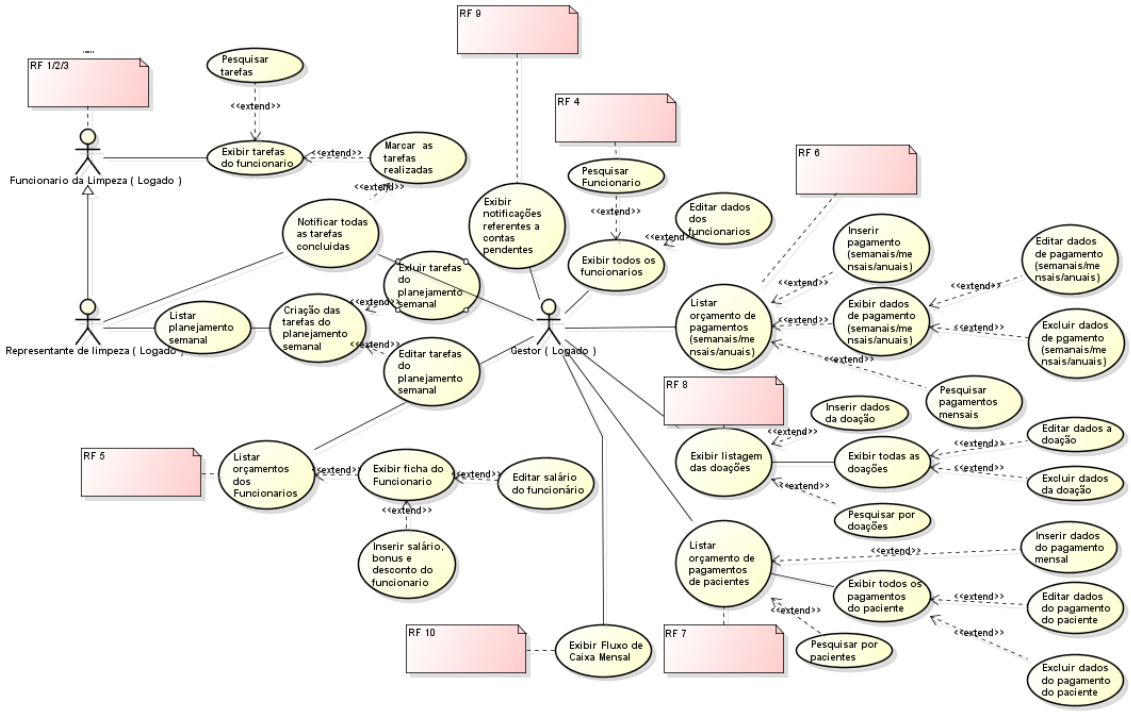
2.2.2 Diagrama de Casos de Uso no Controle Administrativo

Como dito anteriormente, o diagrama de casos de uso do módulo de Controle Administrativo que foi elaborado com o programa *Astah Community*. Na documentação dos Casos de Uso do Controle Administrativos, é possível encontrar casos de uso simples (com menos de 3 fluxos alternativos), médios (entre 3 e 7 fluxos alternativos).

O diagrama foi elaborado tendo como atores o Funcionário da Limpeza e o Representante da Limpeza, que são responsáveis por manusear o planejamento semanal, e também o Gestor, o qual tem acesso a todas as funcionalidades do Controle Administrativo além de ser responsável por manusear algumas dessas, tais como criação de despesas, ficha de funcionário, entre outras. Em suma, os relacioamentos principais do caso de uso da figura 6 são:

- Criação das tarefas do planejanto semanal: O funionário da limpeza acessa o planejamento semanal e inclui as tarefas que serão realizadas na respectiva semana;
- Marcar tarefas realizadas: Quando uma tarefa que foi inclusa no planejamento semanal for concluída, o funcionário deve assinalar essa tarefa para que no sistema conte que ela já foi realizada;
- Exibir ficha do funcionário: O sistema apresenta na interface, as informações sobre o funcionário que o gestor desejar ver;
- Criação de uma despesa: As despesas da instituição devem ser inseridas pelo gestor no sistema;
- Inserir dados do pagamento mensal: O gestor deve inserir o pagamento dos pacientes da instituição no sistema;
- Exibir tabelas fluxo de caixa: O sistema irá exibir as receitas, as despesas e o total de dinheiro da instituição a partir das informações inseridas pelo gestor.

Figura 6 - Diagrama de Casos de Uso do módulo de Controle Administrativo



Fonte: Elaboração própria

2.2.3 Casos de Teste no Controle Administrivo

Esta seção foi elaborada com base nos conceitos da referência [9]

O foco principal desta pesquisa é apresentar o desenvolvimento analítico dos casos de testes modelo caixa-preta referente ao módulo de Controle Administrativo do *software*. No teste de caixa-preta, o analista vai fazer uso dos requisitos (ver 2.2.1) e dos casos de uso (ver 2.2.2) para analisar quais saídas são esperadas em cada funcionalidade, como apresentado nas tabelas a seguir:

Tabela 3: Criar um planejamento semanal de limpeza da instituição

Casos de Teste	**Descrição do Caso de Teste**	**Resultado Esperado**
CT #01	Verifica se o usuário seleciona a opção de criar um novo planejamento, dentro da página de planejamento semanal.	O sistema deve responder ao comando, apresentando uma tela que contenha os campos a serem preenchidos pelo usuário: • Nome da tarefa*; • Descrição da tarefa; • Data de realização da tarefa*; • Status da tarefa • Funcionário responsável pelo setor*; • Nome do setor*; • Observações da tarefa; *Seguido do botão “cadastrar”.
CT #02	Verifica se o usuário preencheu os campos	O sistema apresenta um alerta nos casos

	de maneira incorreta ou incompleta, e seleciona a opção “Criar planejamento”. O preenchimento incorreto ou incompleto do planejamento pode ocorrer caso: • O campo de nome da tarefa não seja preenchido; • A data adicionada seja de um dia anterior ao dia da data de criação do planejamento; • O nome do funcionário responsável pelo setor não seja selecionado; • O nome do setor não seja selecionado.	citados na descrição do teste.

Tabela 4: Marcar como realizada uma tarefa do planejamento semanal de limpeza na instituição

Casos de Teste	Descrição do Caso de Teste	Resultado Esperado
CT #09	Verifica se, na página da tarefa, o usuário clica na caixa de seleção para marcar uma tarefa como realizada.	O sistema atualiza a informação no banco de dados.
CT #10	Verifica se, na página da tarefa, o usuário desmarca a caixa de seleção para colocar uma tarefa como não realizada.	O sistema atualiza a informação no banco de dados.

Tabela 5: Exibir ficha do funcionário

Casos de Teste	Descrição do Caso de Teste	Resultado Esperado
CT #14	Verifica se o gestor selecionou a opção “Pesquisar funcionário” no menu do gestor.	O sistema deve exibir uma tela com uma barra de pesquisa (no cabeçalho da tabela), que de início mostre todas as fichas de todos os funcionários previamente cadastrados. O usuário poderá ordenar os campos de busca como desejar, e os dados irão aparecer em forma de uma tabela com os respectivos campos: • Prontuário; • Nome do funcionário; • CPF. Se o usuário não alterar a ordenação, o sistema irá exibir os funcionários na ordem alfabética, independente se está ativo ou inativo.

Tabela 6: Criar uma despesa

Casos de Teste	Descrição do Caso de Teste	Resultado Esperado
CT #30	Verifica se o usuário selecionou a opção “Criar despesa” no menu do gestor.	O sistema deve exibir um formulário contendo os seguintes campos a serem preenchidos:

		• Nome do serviço*; • Descrição; • Status (pago ou pendente) *; • Data de vencimento*. *Seguido do botão "ok".
CT #31	Verifica se os dados inseridos pelo usuário tem o formato compatível com o formato do código.	O sistema não deve permitir que o usuário digite uma variável incompatível.
CT #32	Verifica se o usuário preencheu todos os campos obrigatórios.	Caso todos os campos obrigatórios tenham sido preenchidos, o salário será validado e o atualizado no banco de dados.

Tabela 7: Inserir dados do pagamento do paciente

Casos de Teste	**Descrição do Caso de Teste**	**Resultado Esperado**
CT #42	Verificar se o gestor clica em "inserir" para inserir as informações do pagamento do paciente.	O sistema deve apresentar uma tela com os campos que devem ser preenchidos pelo usuário.
CT #43	Verifica se os dados inseridos pelo usuário tem o formato compatível com o formato do código.	O sistema não deve permitir que o usuário digite uma variável incompatível.
CT #44	Verifica se o usuário preencheu todos os campos obrigatórios.	Caso todos os campos obrigatórios tenham sido preenchidos, o salário será validado e o atualizado no banco de dados.

Tabela 8: Exibir tabelas de fluxo de caixa

Casos de Teste	**Descrição do Caso de Teste**	**Resultado Esperado**
CT #54	O gestor deverá selecionar a opção de visualizar o fluxo de caixa.	O sistema deverá exibir duas tabelas seguidas que contenham as receitas da empresa e o valor do todas elas, além das despesas, a partir dos valores armazenados no banco de dados. Nessas tabelas devem aparecer: 1ª tabela: • Valores de entrada na empresa • Soma de todos os valores. 2ª tabela: • Valores de saída na empresa (contas de água, luz, etc.); • Soma das despesas. Ao final das duas tabelas, deve ser exibido o saldo total da empresa. Se o valor for positivo a cor deve ser verde, mas se o valor for negativo a cor deve ser vermelha.

Em suma, o teste consiste em concentrar na análise do funcionamento visual do *software*, percebendo e revelando problemas como na interface, funções incorretas ou omitidas, erros de comportamento ou então erros de iniciação e término.

Deve ser reportado na documentação, os resultados dos casos de teste, como é possível observar na tabela a seguir:

Tabela 9: Resultados dos Casos de Testes

	02/08/2019	09/08/2019	23/08/2019	06/09/2019	20/09/2019	24/09/2019	25/09/2019
CT #01	**Sucesso**				**Falha**	**Falha**	**Sucesso**
CT #02	**Falha**		**Sucesso**				
CT #03	**Falha**		**Sucesso**				
CT #04	**Falha**		**Sucesso**				
CT #05	**Falha**		**Sucesso**				
CT #06	**Falha**		**Sucesso**				
CT #07	**Falha**		**Sucesso**				
CT #08	**Sucesso**			**Falha**		**Sucesso**	
CT #09					-		
CT #10					-		
CT #11		**Sucesso**					
CT #12					-		
CT #13					-		
CT #14		**Sucesso**				**Falha**	
CT #15					-		
CT #16					-		
CT #17					-		
CT #18					-		
CT #19					-		
CT #20					**Sucesso**		
CT #21					**Sucesso**		
CT #22					**Sucesso**		
CT #23					**Sucesso**		
CT #24					**Sucesso**		
CT					**Sucesso**		

#25							
CT #26					**Sucesso**		
CT #27					**Sucesso**		
CT #28					**Sucesso**		
CT #29					**Sucesso**		
CT #30					**Sucesso**		
CT #31					**Sucesso**		
CT #32					**Sucesso**		
CT #33					**Sucesso**		
CT #34					**Sucesso**		
CT #35					**Sucesso**		
CT #36					**Sucesso**		
CT #37					**Sucesso**		
CT #38					**Sucesso**		
CT #39							-
CT #40							**Sucesso**
CT #41							**Sucesso**

Nos casos de falha, é necessário descrever o motivo da falha em uma tabela separada, dessa forma, os desenvolvedores sabem exatamente qual o problema que eles devem corrigir:

Tabela 10: Descrição das falhas reportadas

Casos de Teste	Data da Execução	Responsável da Atividade de Teste	Status do Teste	Motivo da Falha	Responsável pela Correção da Falha	Data Prevista para Novo Teste
CT #03 ao CT #07	02/08/2019	Geovanna e Thalyson	**Falha**	As funções do "alterar" e do "excluir" estavam vazias.	Heloisa e Matheus	09/08/2019
CT #08	06/09/2019	Geovanna e Thalyson	**Falha**	O requisito não foi bem desenvolvido, dado que os *stakeholders* e outras informações não foram adicionados.	Heloisa e Matheus	25/09/2019
CT #01 e CT #02	20/09/2019	Thalyson e Thomaz	**Falha**	Esse requisito está sendo desenvolvido novamente, dado que foi editado pelo	Matheus	25/09/2019

				desenvolvedor e apresentou variados bugs.		
CT #14	24/09/2019	Geovanna e Thalyson	**Falha**	Esse requisito está sendo refeito.	Matheus	26/09/2019

3. Conclusões e Recomendações

Como dito na contextualização da pesquisa, todos os anos, o IFSP busca proporcionar maior conexão entre a comunidade da cidade de São João da Boa Vista e a tecnologia trabalhada no campus. Dessa forma, visto o grande crescimento da população da terceira idade na cidade, esse Projeto teve como intuito corroborar com a informatização das Instituições de Longa Permanência da cidade e região, auxiliando na administração da empresa. Dito isso, é válido ressaltar a relevância do módulo de Controle Administrativo, o qual seu trabalho foi focado em proporcionar um planejamento organizado e simples para a equipe de limpeza da instituição, visando mais produtividade. Além disso, com o intuito de diminuir a papelada e controlar a área financeira da instituição, o módulo planejou e desenvolveu funcionalidades que controlam a entrada e saída de dinheiro no local.

O objetivo geral da pesquisa consistia em apresentar o processo de desenvolvimento analítico da atividade de Casos de Teste do módulo de Controle Administrativo. O desenvolvimento dessa atividade teve a duração de 3 bimestres e foi crucial para a garantia do funcionamento correto das funcionalidades, afinal a fase de testes é uma fase do modelo de desenvolvimento “Cascata”.

À vista disso, realizar testes na interface do *software* foram de suma importância para a qualidade do referido, pois, dessa forma, o usuário não encontrará problemas no momento de utilizar as funcionalidades. A correção de funcionalidades que apresentaram falha durante o teste desenvolvido pelo analista, proporcionou maior segurança de que o produto final entregue será satisfatório ao cliente.

O Levantamento de Requisitos é sempre importante pois se trata do planejamento do sistema, a qualidade desse foi imprescindível para melhor qualidade no processo de desenvolvimento do projeto, visto que outras atividades se basearão nele. A documentação dos requisitos do Controle Administrativo foi elaborada de forma detalhada e extensa, para evitar desentendimentos futuros entre os membros do módulo, por isso, nessa documentação, foi explicado não somente o que seria feito, mas também, a linguagem que seria usada e outros termos técnicos. Todos participaram dessa atividade que foi verificada pelos professores da disciplina de PDS.

Após concluir a documentação do Levantamento de Requisitos, a próxima atividade foi iniciar o desenvolvimento dos Casos de Uso. Para a elaboração dessa atividade, foi necessário consultar o documento de requisitos, para assim, criar fluxos alternativos condizentes. O diagrama do módulo conta com 35 casos de usos. O desenvolvimento dessa atividade foi necessário para a documentação dos Casos de Teste.

A documentação dos Casos de Teste consistiu em transformar os fluxos alternativos do Caso de Uso em Casos de Testes, sendo que cada um desses foi testado. Além de testar, os resultados dos testes foram reportados e as falhas foram especificadas em uma outra tabela. Essa documentação gerou ao todo 54 casos de testes, sendo que essa pesquisa apresentou os principais desses.

A execução dos Casos de Teste ocorreu conforme as funcionalidades eram desenvolvidas e inseridas pelos desenvolvedores na plataforma SVN. Esse processo gerou um documento com 11 versões que estão relatadas no Histórico de Revisão da documentação.

Ao final, o objetivo do trabalho que consistia em apresentar o processo de desenvolvimento de casos de teste do módulo de Controle Administrativo foi concluído, contudo, não foi possível apresentar com detalhes seu processo completo devido ao fato de que o Projeto Gerações seguirá sendo desenvolvido até sua conclusão total.

Após isso, é válido expor que esse processo, como qualquer outro, teve pontos negativos. Um desses, é o fato de que o Projeto Gerações foi desenvolvido por uma equipe total de 58 pessoas (sendo 56 alunos e 2 professores), portanto, gerenciar essa equipe grande foi um desafio. Contudo, esse desafio resultou em mais pontos positivos do que negativos, afinal a experiência adquirida nesse Projeto fez efeito na vida de todos os participantes.

3 Referências Bibliográficas

[1] **Pesquisa Aponta São João Da Boa Vista Co/mo Melhor Cidade Para Idosos**. G1, 2017. Disponível em: <http://g1.globo.com/sp/sao-carlos-regiao/noticia/2017/03/pesquisa-aponta-sao-joao-da-boa-vista-como-melhor-cidade-para-idosos.html> Acesso em: 16 ago. 2019

[2] **Qualidade De Vida Em Instituições De Longa Permanência Para Idosos: Considerações A Partir De Um Modelo Alternativo De Assistência**. UFSJ, 2010. Disponível em: <https://ufsj.edu.br/portal2-repositorio/File/mestradopsicologia/2010/Dissertacoes/Dissertacao_Gleicimara%20.pdf>Acesso em: 25 ago. 2019

[3] **Agência Nacional De Vigilância Sanitária**. MINISTÉRIO DA SAÚDE, 2005. Disponível em: <https://svn.sbv.ifsp.edu.br/svn/pds2019vespertino/documentacao/comum/Documentos%20de%20Institui%c3%a7%c3%b5es%20de%20Longa%20Perman%c3%aancia%20-%20Estudo/04%20-%20RDC_283_2005_COMP.pdf> Acesso em: 23 ago. 2019

[4] **Missão, Visão E Valores**. IFSP, 2017. Disponível em: <http://www.ifs.edu.br/sobre-o-campus-propria/missao-visao-e-valores> Acesso em: 16 ago. 2019

[5] JÚNIOR, W. M. P. **Apostila Engenharia De Software**. Disponível em: <http://www.waltenomartins.com.br/ap_es_v1.pdf> Acesso em: 08 out. 2019

[6] GUDWIN, R. R. **Engenharia de Software: Uma Visão Prática**. Disponível em: <http://faculty.dca.fee.unicamp.br/gudwin/sites/faculty.dca.fee.unicamp.br.gudwin/files/ea975/ESUVP2.pdf> Acesso em: 08 out. 2019

[7] FILHO, W. P. P. **Engenharia de Software – Fundamentos, Métodos e Padrões**: 3. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora, 2009

[8] DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao Teste de Software**: 4. ed. Rio de Janeiro: Editora Campus, 2007.

[9] KOSCIANSKI, A.; SOARES, M. S. **Qualidade de Software**: 2. ed. São Paulo: Editora Novatec, 2010.

[10] FRANCETO, Simone. **ESPECIFICAÇÃO E IMPLEMENTAÇÃO DE UMA FERRAMENTA PARA ELICITAÇÃO DE REQUISITOS DE SOFTWARE BASEADA NA TEORIA DA ATIVIDADE**, 2005. Disponível em: <https://www.unimep.br/phpg/bibdig/pdfs/2006/VMEPWGALFNMD.pdf> Acesso em: 26 set. 2019

[11] AUGUSTO TACLA, Cesar. **ANÁLISE E PROJETO OO & UML 2.0**. Disponível em: <http://www.dainf.ct.utfpr.edu.br/~tacla/UML/Apostila.pdf> Acesso em: 26 set. 2019