**UNIVERSIDADE REGIONAL DE BLUMENAU**

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**

**CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO**

# PORTAL SGMS: SISTEMA WEB DE APOIO AO PROCESSO DE GERÊNCIA E CONTROLE NA MANUTENÇÃO DE SOFTWARE

**THIAGO RIBEIRO VIEIRA**

**BLUMENAU**
**2016**

**THIAGO RIBEIRO VIEIRA**

# PORTAL SGMS: SISTEMA WEB DE APOIO AO PROCESSO DE GERÊNCIA E CONTROLE NA MANUTENÇÃO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Profª. Luciana Pereira de Araújo, Mestra - Orientadora

**BLUMENAU**
**2016**

# PORTAL SGMS: SISTEMA WEB DE APOIO AO PROCESSO DE GERÊNCIA E CONTROLE NA MANUTENÇÃO DE SOFTWARE

Por

**THIAGO RIBEIRO VIEIRA**

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

| | |
|---|---|
| Presidente: | Prof. Luciana Pereira de Araújo, Mestra – Orientador, FURB |
| Membro: | Prof. Gilvan Justino, Mestre – FURB |
| Membro: | Prof. Simone Erbs da Costa, Especialista – FURB |

Blumenau, 01 de dezembro de 2016

Dedico este trabalho aos familiares, amigos, professores, colegas de trabalho e especialmente aqueles que me ajudaram diretamente na realização deste.

## AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que sempre esteve presente.

À minha orientadora, Profª. Luciana Pereira de Araújo, por ter me orientado, colaborado e acreditado na realização deste trabalho.

Aos meus amigos, aqueles que, direta ou indiretamente estiveram ao meu lado durante esta caminhada.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

José de Alencar

# RESUMO

Este trabalho tem como objetivo o desenvolvimento de um sistema web para atender os segmentos que visam planejar e controlar a manutenção de software em uma organização com vários produtos ou projetos. O sistema promove a troca de informações entre os envolvidos no processo de manutenção, permitindo que as solicitações sejam gerenciadas, acompanhadas e controladas até a sua conclusão. A aplicação foi desenvolvida em linguagem C# utilizando banco de dados SQL Server, no qual foram publicados em uma rede local em uma empresa do setor de software da região. Com os indicadores foi possível acompanhar a performance da manutenção nos seus processos dentro do ciclo de vida do software e ao final foi aplicado uma avaliação heurística com três membros desta empresa para se chegar aos resultados e conclusões finais.

Palavras-chave: Processo de manutenção. Ciclo de vida do software. Solicitações.

# ABSTRACT

This work aims to develop a web system to serve the segments that aim to plan and control the maintenance of software in an organization with several products or projects. The system promotes the exchange of information among those involved in the maintenance process, allowing the requests to be managed, monitored and controlled until their completion. The application was developed in C # language using SQL Server database, in which they were published in a local network in a software company in the region. With the indicators it was possible to track the performance of the maintenance in its processes within the software life cycle and at the end a heuristic evaluation with three members of this company was applied to reach the final results and conclusions.

Key-words: Maintenance process. Life cycle of the software. Requests.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

DCU - Diagrama de Caso de Uso

EA - Enterprise Architect

HTML - HyperText Markup Language

IEC - International Electrotechnical Comission

ISO - International Organization for Stadarzation

MER - Modelo de Entidade e Relacionamento

MVC - Model View Controller

OO - Orientação a Objetos

RF - Requisito Funcional

RNF - Requisito Não Funcional

SGMS – Sistema de Gerência na manutenção de software

SLA - Service Level Agreement

SQL - Structured Query Language

UC - Casos de Uso

# SUMÁRIO

# 1 INTRODUÇÃO

A manutenção de software é uma das fases em Engenharia de Software do ciclo do desenvolvimento de um software, a qual engloba as etapas de correção, adaptação e evolução (MAGELA, 2006 apud WEBER, 2014, p. 12). Por sua vez, um dos problemas que a equipe de manutenção (suporte, desenvolvimento, teste e documentação) enfrenta está relacionada ao gerenciamento das atividades da mesma (WEBER, 2014).

Conforme Magela (2006 apud WEBER, 2014, p. 15), "a manutenção de software custa no mínimo 90% do valor original do projeto, sem ter um limite para o valor máximo". A manutenção de software também é reconhecida como a atividade que demanda o maior volume de esforço dentre todas as atividades de Engenharia de Software (MAGELA, 2006 apud WEBER, 2014, p. 12). Ela é definida como a modificação de um produto de software depois de sua entrega (ao cliente) para corrigir erros, melhorar sua performance ou qualquer outro atributo, ou para adaptar o produto a um ambiente modificado (IEEE, 1998). Conforme ISO (1999), "este processo normalmente é desencadeado por uma solicitação do cliente ou por algum relatório de problemas gerado pelo usuário".

Sendo assim, "o problema do alto custo referente a manutenção não passa de uma má comunicação entre a equipe responsável por realizar a manutenção e o cliente" (APRIL e ALAN, 2008 apud SANTOS, 2015, p. 34). Dessa forma, há a necessidade de uma ferramenta que auxilie e organize as solicitações ou demandas encaminhadas ao processo de manutenção de software, bem como disponibilize ao solicitante informações de onde e como anda a execução da sua solicitação.

Diante deste cenário, este trabalho apresenta o desenvolvimento de um portal web para atender os segmentos que visam controlar a manutenção de software em uma empresa com vários produtos ou projetos. De um modo geral, pretende-se garantir o andamento e integridade na gestão das solicitações em determinada área de serviço. O portal proporciona o trabalho em equipe, através de um mecanismo de controle de solicitações, que notifica via e-mail e autoriza o registro das demais informações aplicáveis a cada etapa do processo. Este mecanismo busca assegurar a agilidade e o compromisso com o cumprimento dos prazos em todas as etapas do processo da manutenção de software.

Sendo assim, este trabalho aborda a automatização e controle de um processo de manutenção de software, desde o registro da falha até a sua entrega com a respectiva correção. O trabalho se dedica justamente na integração e execução de todos os papéis durante o ciclo

de manutenção do software, no qual cada membro do processo poderá exercer seu papel e assim melhorar a eficiência e a eficácia dos processos no ciclo de manutenção.

## 1.1 OBJETIVOS

O objetivo geral do trabalho é o desenvolvimento de um sistema web para atender os segmentos que visam planejar, controlar e gerenciar a manutenção de software em uma organização com vários produtos ou projetos.

Os objetivos específicos do trabalho são:

a) oferecer um módulo para o acompanhamento e gerenciamento dos processos no ciclo de vida da manutenção de software por cada um dos papéis;
b) permitir a centralização de indicadores de performance e atividade da manutenção nos seus processos;
c) garantir o andamento e a integridade da gestão das solicitações ou demandas nas determinadas áreas de serviço, como suporte, desenvolvimento, teste e documentação.

## 1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos. O primeiro capítulo é composto pela introdução, os objetivos definidos para o mesmo e a apresentação de sua estrutura.

O segundo capítulo apresenta a fundamentação teórica, ciclo de vida do desenvolvimento de software, processos na manutenção do software e os trabalhos correlatos. No terceiro capítulo é apresentado o desenvolvimento da ferramenta, listados os requisitos principais, diagramas de casos de uso, descrição dos casos de uso, diagrama de classe, diagrama de atividade, modelo conceitual de dados, sua implementação, assim como técnicas e ferramentas utilizadas, operacionalidade da implementação e os resultados e discussões. Por fim, o quarto capítulo apresenta as conclusões e as sugestões para trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Esse capítulo aborda assuntos a serem apresentados nas seções a seguir, tais como o ciclo de vida do desenvolvimento de software, processos na manutenção do software e os trabalhos correlatos.

## 2.1 CICLO DE VIDA DO DESENVOLVIMENTO DE SOFTWARE

O ciclo de vida do software descreve como um software deve ser desenvolvido, onde está envolvido desde o início da ideia de sua criação até a sua descontinuação. Como o processo de software deve atender todo o ciclo de vida de um software, ele estará em todas as etapas de desenvolvimento, chegando ao conceito de modelo de Ciclo de Vida de um Processo (MAGELA, 2006 apud WEBER, 2014, p. 14). Basicamente, o ciclo de vida define a ordem das atividades envolvidas em um projeto de software.

O ciclo de vida é a estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término ou descontinuação do sistema. De acordo com Associação Brasileira de Normas Técnicas (1998, p. 1):

> Esta norma estabelece uma estrutura comum para os processos de ciclo de vida e software desde a concepção de ideias até a descontinuação do software com uma terminologia bem definida e é composta de processos, atividades e tarefas que servem para ser aplicada durante a aquisição de um sistema que contém software, de um produto de software independente ou de um serviço de software, e durante o fornecimento, desenvolvimento, operação e manutenção de produtos de software (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 1998, p. 1).

O modelo de ciclo de vida é a primeira escolha a ser feita no processo de software. A partir desta escolha define-se a maneira mais adequada de obter as necessidades do cliente (MACÊDO; SPÍNDOLA, 2016). Exemplos de ciclos de vida de software são: Espiral, Cascata, Ciclo de Vida Iterativo, entre outros.

O modelo de clico de vida espiral apresenta características do modelo Cascata e prevê da Prototipação para obter um maior controle sobre os riscos do projeto, ou seja, tornando o processo de construção de um produto mais seguro. A prototipação consiste na construção de um sistema de forma rápida e que o usuário possa avaliá-lo. O protótipo é uma versão funcional do sistema ou de parte dele (TEIXEIRA, 2007).

Laudon (2002, p. 7) enfatiza que “uma vez disponibilizado, o protótipo vai sendo refinado até chegar numa versão final que atenda completamente às necessidades do usuário”. A vantagem da abordagem da prototipação fica evidenciada quando existe alguma incerteza

sobre os requerimentos, sendo útil para projetar a interface final do usuário ou a parte final do sistema com a qual o usuário interage: tela, relatório ou página na web.

> A metodologia espiral foi concebida para englobar as melhores práticas tanto do ciclo de vida clássico quanto da prototipação. Essa metodologia inovou ao trazer também um novo elemento, a análise de riscos. Além disso, foi uma das primeiras metodologias a adotar o conceito de iteração. Sucessivas iterações moldam às poucas soluções mais completas do software (PRESSMAN, 2006).

Na primeira iteração do modelo espiral, os objetivos, alternativas e restrições são definidos e os riscos são identificados e analisados. O cliente avalia o resultado da iteração e baseado nos apontamentos do mesmo, inicia-se a próxima iteração. Isso possibilita ao cliente e ao desenvolvedor perceber e reagir a riscos em cada uma das etapas evolutivas. Entretanto, a metodologia espiral exige considerável experiência para avaliar os riscos e obter sucesso. Encara-se que se um grande risco não for detectado, ocorrerão problemas futuramente (LAUDON, 2002).

Por sua vez, o modelo de ciclo de vida em cascata tem suas etapas bem definidas e estruturadas. Esse modelo possui um conceito básico, no qual uma etapa forneça informações e fundamentação para que sejam usadas como entradas para a etapa seguinte. Portanto, o processo de desenvolvimento do software é simples de conhecer e controlar (RAMOS et al., 2010).

Já o modelo iterativo apresenta um ciclo de vida iterativo baseado no aumento e no refinamento sucessivo de um sistema através de múltiplos ciclos de desenvolvimento de análise, de projeto, de implementação e de teste (LARMAN, 2000 apud SIMAS et al., 2014, p. 2). Sendo assim, Simas et al. (2014, p. 3) enfatiza que “a equipe fica focada com os objetivos de cada incremento, trabalhando de maneira mais eficiente”.

Nas seções a seguir são apresentados dois modelos do ciclo de vida do desenvolvimento de software, sendo eles o modelo cascata e o modelo iterativo. Assim como, os processos na manutenção do software e a definição de cada atividade em todo o seu ciclo.

### 2.1.1 Modelo Cascata

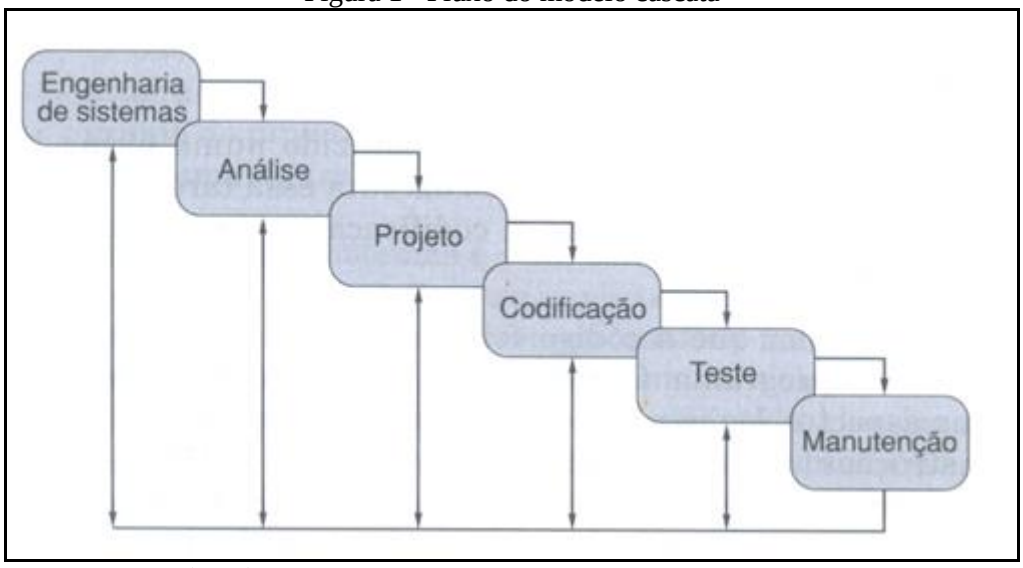
O modelo cascata consiste de fases e atividades que são realizadas sequencialmente, de forma que uma atividade deve ser iniciada após o término do seu antecessor.

> Na primeira vez que uma fase de desenvolvimento é completada, o desenvolvimento prossegue para a próxima fase e não há retorno. A vantagem do desenvolvimento cascata é que ele permite controle departamental e gerencial. Um planejamento pode ser atribuído com prazo final para cada estágio de desenvolvimento e um produto pode prosseguir no processo de desenvolvimento, teoricamente ser entregue no prazo. O desenvolvimento move do conceito, através do projeto (design), implementação, teste, instalação, descoberta de defeitos e termina com a operação e manutenção. Cada fase de desenvolvimento prossegue em uma ordem estrita, sem

qualquer sobreposição ou passos iterativos (PRESSMAN, 2006).

A Figura 1 demonstra as etapas discutidas por Pressman (2006), sendo que esse modelo sugere uma abordagem sequencial para o desenvolvimento de software. O fluxo é iniciado na engenharia de sistema e finalizado na manutenção. Um dos problemas identificados neste ciclo de vida, segundo Pressman (2006), "é que os projetos reais raramente seguem o fluxo sequencial que o modelo propõe. Alguma iteração sempre ocorre e traz problemas na aplicação do paradigma".

Figura 1 - Fluxo do modelo cascata




Fonte: Pressman (2006).

O primeiro estágio representado pela engenharia de sistemas é uma abordagem sistemática e disciplinada para o desenvolvimento de software (PRESSMAN, 2006). A base da engenharia de software é formada por conjuntos de atividades para o processo de desenvolvimento de software. Portanto, o ciclo em cascata é um modelo de engenharia projetado para ser aplicado no desenvolvimento do software, onde assim, a ideia principal que o dirige é que as diferentes etapas de desenvolvimento seguem uma sequência (RAMOS et al., 2010).

O segundo estágio do modelo cascata é a análise de requisitos. É neste momento que é efetuado o conhecimento do problema para o desenvolvimento do software (JALOTE, 2005). O terceiro estágio é o projeto, que, segundo Ramos et al. (2010, p. 1), "é um processo de vários passos que se centraliza em quatro atributos diferentes do sistema: estrutura de dados, arquitetura do software, detalhes procedais e caracterização das interfaces".

A codificação é a quarta etapa do ciclo, onde é transformado o problema em código fonte, ou seja, em uma linguagem de programação que foi definida nas etapas anteriores. Já o

teste, que ocupa a quinta etapa, tem o papel de medir o controle da qualidade do software durante o desenvolvimento (JALOTE, 2005).

Por fim, a sexta etapa é a manutenção. Essa é a etapa final do ciclo do modelo cascata que consiste na correção de erros que não foram previamente detectados. Também pertence a esta etapa melhorias funcionais e outros tipos de suporte (PERINI, 2009). As etapas mencionadas são as principais, porém podem existir sub-etapas dentro de cada etapa, as quais diferem muito de um projeto para outro.

As principais vantagens desse modelo são suas etapas bem definidas e estruturadas. Quanto a desvantagem, pode-se destacar que o modelo é muito rígido, o que não facilita a aplicação de mudanças nas etapas, podendo tornar a construção do software engessada, o que representa algo inviável caso seja seguido à risca (SOUZA, 2015).

### 2.1.2 Ciclo de Vida Iterativo

O ciclo de vida iterativo é formado pelas etapas concepção, elaboração, construção e transição (ROYCE, 1998). Concepção, de acordo com Pádua (2011), é a "fase na qual se justifica a execução de um projeto de desenvolvimento de software, do ponto de vista do negócio do cliente".

A elaboração é a fase na qual o produto é detalhado o suficiente para permitir um planejamento, eliminando principais riscos. Nessa fase também é definido uma arquitetura estável para o sistema (PÁDUA, 2011). A construção é o momento de desenvolvimento do produto até que ele esteja pronto para testes. Esta etapa de implementação, segundo Mazzola (2010, p. 16) é:

> Onde as representações realizadas na etapa de projeto serão mapeadas numa ou em várias linguagens de programação, a qual será caracterizada por um conjunto de instruções executáveis no computador; nesta etapa, considera-se também a geração de código de implementação, aquele obtido a partir do uso de ferramentas (compiladores, linkers, etc.) e que será executado pelo hardware do sistema (MAZZOLA, 2010, p.16).

A última etapa ou fase é a transição, na qual o produto é colocado à disposição do usuário. Nessa etapa serão realizados os testes de aceitação no ambiente dos usuários e operações do produto do cliente, com a resolução de problemas através de processo de manutenção (PÁDUA, 2011).

O desenvolvimento de um sistema termina quando o produto é liberado para o cliente e o software é instalado para uso operacional. Daí em diante, deve-se garantir que esse sistema continue sendo útil e atendendo às necessidades do usuário (ROCHA, 2001).

## 2.2 PROCESSOS NA MANUTENÇÃO DE SOFTWARE

O modelo de processo de manutenção proposto pela norma ISO/IEC 12207 delimita o escopo da fase de manutenção como o período após a primeira liberação funcional do software aprovada pelos usuários. Entretanto, a transição entre a fase de desenvolvimento inicial e a fase de manutenção impõe mudanças nos processos de software, principalmente no que tange a perspectiva da execução desses processos (ISO, 2008).

> O processo de manutenção contém as atividades e tarefas do mantenedor. Este processo é ativado quando o produto de software é submetido a modificações no código e na documentação associada devido a um problema, ou à necessidade de melhoria ou adaptação. O objetivo é modificar um produto de software existente, preservando a sua integridade. Este processo inclui a migração e a descontinuação do produto de software. O processo termina com a descontinuação do produto de software (BERNARDI, 2003, p. 1).

Segundo Pressman (2011, p. 40):

> No contexto da engenharia de software, um processo não é uma prescrição rígida de como desenvolver um software. Ao contrário, é uma abordagem adaptável que possibilita às pessoas (a equipe de software) realizar o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas. A intenção é a de sempre entregar software dentro do prazo e com qualidade suficiente para satisfazer àqueles que patrocinaram sua criação e àqueles que irão utilizá-lo (PRESSMAN, 2011, p. 40).

Dentro do processo de manutenção são estabelecidos papéis para desempenhar atividades. Os papéis não são indivíduos, nem são necessariamente equivalentes aos cargos, e sim descrevem como os indivíduos atribuídos aos papéis se comportarão no contexto de um processo. Os papéis a serem desempenhados no processo são o de usuário (solicitante), suporte, desenvolvimento, testes e a documentação.

Sendo assim, de acordo com Yongchang et al. (2011, p. 3, tradução nossa), “manutenção de software começa com um pedido para mudar o sistema, geralmente um pedido do usuário”. A partir desse pedido, surge o papel fundamental para o processo de manutenção começar, sendo o usuário, que é quem interage com o produto (GUEDES, 2015). Sempre que surgir alguma dúvida ou algum problema for identificado, o usuário deverá descrever os procedimentos que está realizando, conforme seu conhecimento de uso do produto.

Outro papel visto no processo de manutenção de software é o papel de suporte ao usuário. Esse papel tem o dever de prestar assistência, a fim de solucionar dúvidas ou problemas técnicos relacionados ao produto. O atendimento de suporte é dado quando o usuário identifica erros ou dúvidas durante o uso dos produtos, e que levam determinada função não ser executada como o cliente desejaria (YONGCHANG et al., 2011). Os pedidos

abertos ao suporte devem ser analisados, ou seja, ao se constatar o erro no sistema, deve-se complementar o chamado com as novas informações coletadas, em especial a forma de simular o erro e classificá-la.

Após o suporte finalizar seu papel, a próxima etapa é voltada para a manutenção do sistema, ou seja, o desenvolvimento. Segundo Bellin (1993, p. 1), a manutenção "inclui procedimentos para assegurar que os programas funcionem adequadamente, corrigi-los quando necessário e incrementá-los com novas funções". O foco da manutenção é na solução técnica, ou seja, ser responsável por decisões técnicas e assim realizar as devidas alterações no código fonte do sistema. O implementador é quem procede a correção dos defeitos identificados e também realiza análise de impacto da correção. Segundo Yongchang et al. (2011, p. 3, tradução nossa), a implementação deverá "formular planos para mudar o sistema, incluindo o seguinte processo: codificação, testes, análise de risco, avaliação e documentação de atualização".

Desta forma, uma pessoa de manutenção é um solucionador de problemas. Porém na manutenção pode existir uma distância de comunicação entre os usuários e aqueles que dominam a tecnologia (BELLIN, 1993). Ou seja, a má interpretação do implementador sobre a solicitação, poderá resultar em uma correção indevida. Assim é de responsabilidade do suporte em ajudar nesse processo de empatia com o usuário final. Ainda sobre a manutenção, Zhang (2003, p. 2, tradução nossa) enfatiza que:

> Na manutenção de software a carga de trabalho é muito grande, embora em diferentes áreas os custos de manutenção em aplicação podem variar amplamente, mas, em média, a manutenção de software em grande escala custa cerca de 4 vezes os custos de desenvolvimento. Muitas equipes de desenvolvimento de software em países estrangeiros, usam 60 por cento da mão de obra para a manutenção do software existente (Zhang, 2003, p. 2, tradução nossa).

Deve ser verificado a qualidade da demanda implementada, assim como certificar-se da documentação relacionada, para ser evitado retrabalho (SANTOS, 2015). Portanto, ao concluir as devidas alterações pela manutenção, o próximo papel do processo é voltado aos testes do sistema. Trodo (2009, p. 12) afirma que:

> O teste de software pode ser considerado como uma fase do processo de desenvolvimento de software, cujo objetivo é atingir um nível de qualidade de produto superior. Os testes antecipam a descoberta de falhas e incompatibilidades, reduzindo o custo (TRODO, 2009, p. 12).

O teste tem o foco na qualidade da solução sob o ponto de vista dos usuários ou clientes. O testador deve verificar a aderência da solução implementada para as solicitações registradas e especificadas através do uso e reuso das funções e funcionalidades afetadas

observando-se os resultados da implementação frente aos requisitos de negócio (YONGCHANG et al., 2011).

Após finalizado os testes das demandas durante o período definido para liberação do projeto, é gerado um kit contendo todas as demandas corrigidas para realização dos testes de regressão (testes gerais efetuados no pacote antes da liberação). Esse kit tem por objetivo validar as integrações entre os produtos executando checklists das rotinas mais críticas.

Por fim, segundo Santos (2015, p. 96) "a última atividade, em relação à demanda, é verificar se o catálogo foi devidamente atualizado. Este é parte essencial da documentação relativa à demanda". O documentador deve realizar correções/complementações nos documentos de acordo com a solicitação recebida e assim disponibilizar versões alteradas para entrega com novas *releases* ou download com os documentos alterados para serem liberados.

Na seção a seguir é apresentado o Scrum, com seu conceito e como podemos utiliza-lo na manutenção de software. Assim como, ele é aplicado dentro do processo de manutenção.

### 2.2.1 Scrum

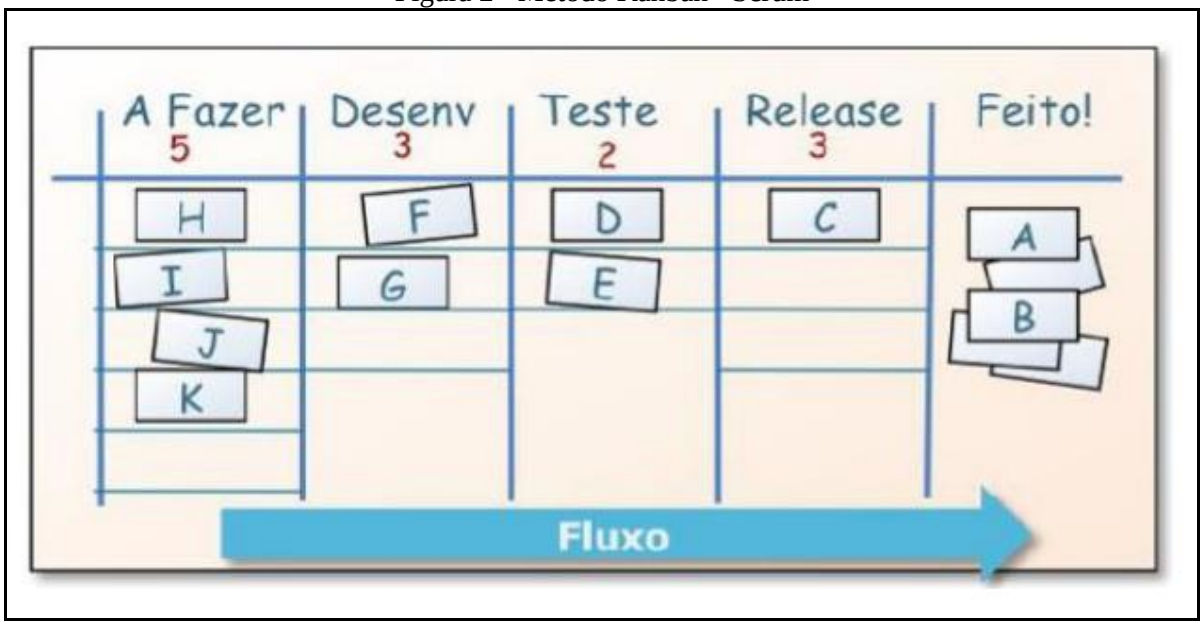
Dentro do processo de manutenção pode ser aplicado o Scrum. O Scrum é uma abordagem iterativa e incremental para gestão de projetos que visa tornar os processos ágeis e flexíveis, promovendo a transparência, inspeção e adaptação do projeto (NONAKA e TAKEUCHI, 1986 apud OLIVEIRA; MUNIZ, 2014, p. 2). O Scrum também visa promover um modelo de auto-organização da equipe, aonde todos são responsáveis pela qualidade, além de fornecer maior visibilidade e melhores níveis de detalhamento (GRIMHEDEN, 2013), transparência em decisões e acompanhamento dos processos (VLAANDEREN et al., 2011).

No caso da manutenção é aplicado o Scrum em Bug *Fixing*, um tipo de projeto destinado somente para correções do produto. Nestes projetos envolvem corrigir defeitos somente em projetos que estão em produção ou que de alguma forma, já não estão mais em desenvolvimento. Dessa forma, o projeto é realizado com Sprint. A Sprint, segundo Silva, Pires e Neto Carvalho (2015, p. 1) "é um período, geralmente de um mês ou menos, de execução das atividades, no qual um atributo ou funcionalidade é criado atendendo determinados itens do produto".

Cada item da Sprint é dividido em conjuntos de tarefas que representam pequenos passos que são necessários para que o item esteja pronto (SILVA, PIRES e NETO, 2015). Sendo assim, o sistema de Kanban é utilizado para detalhar quais são os incidentes que estão

sendo trabalhados e quais já foram corrigidos. A Figura 2 demonstra um exemplo do Kanban que deve estar em lugar visível para todo o time.

Figura 2 - Método Kanban - Scrum




Fonte: Kniberg e Skarin (2009).

## 2.3 TRABALHOS CORRELATOS

Esta seção apresenta uma ferramenta, o Qualitor e como ele é aplicado no atendimento e suporte ao cliente. Além disso, apresentação de dois trabalhos acadêmicos para o melhoramento no processo e controle na gestão de solicitações encaminhadas ao processo de manutenção.

### 2.3.1 Qualitor

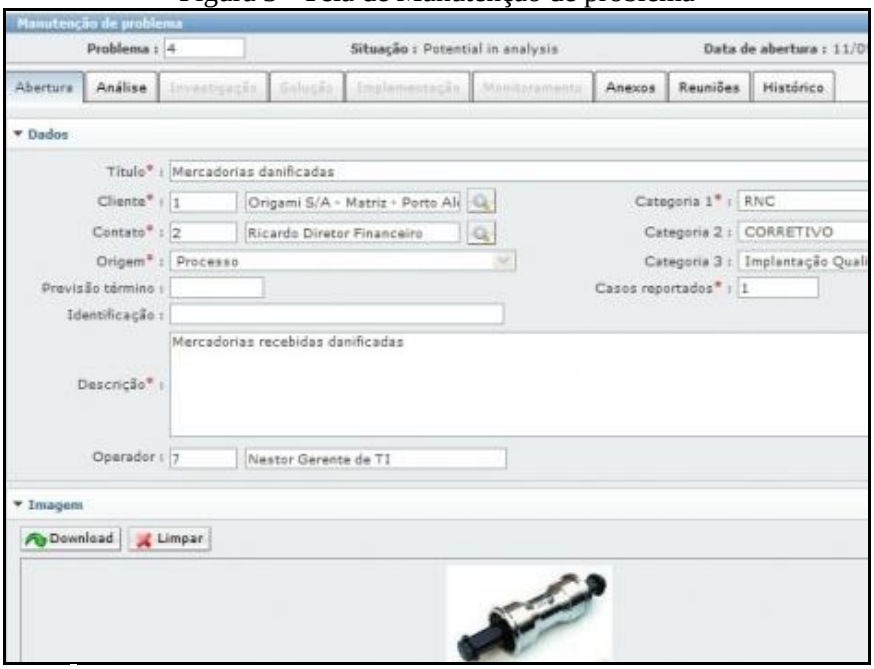
Um sistema correlato ao portal (trabalho desenvolvido), entretanto só atendendo a gestão do suporte, é o Qualitor (2016). O Qualitor é um software para gerenciamento de atendimento, podendo ser utilizado em *service-desk* ou help-desk (QUALITOR, 2016).

Um dos maiores diferenciais que o Qualitor possui em relação à gestão de suporte é o fato de concentrar estes recursos (controle de atendimento, planos de ação e entre outros) a outras funcionalidades conhecidas do produto, como catálogo de serviços, *scripts* de atendimento e recursos de gestão de conhecimento. Além disso, possui uma área que pode ser utilizada junto às instâncias como apoio, seja para visualização de artigos de base de conhecimento ou anexos (QUALITOR, 2016).

Na Figura 3 é apresentada a tela de abertura de uma demanda, uma das funcionalidades do Qualitor, onde o usuário passa as informações do problema ao suporte. Sendo assim, o

Qualitor tem o objetivo de trabalhar na abertura e gestão de demanda para o suporte e atendimento para determinado produto.

Figura 3 - Tela de Manutenção de problema



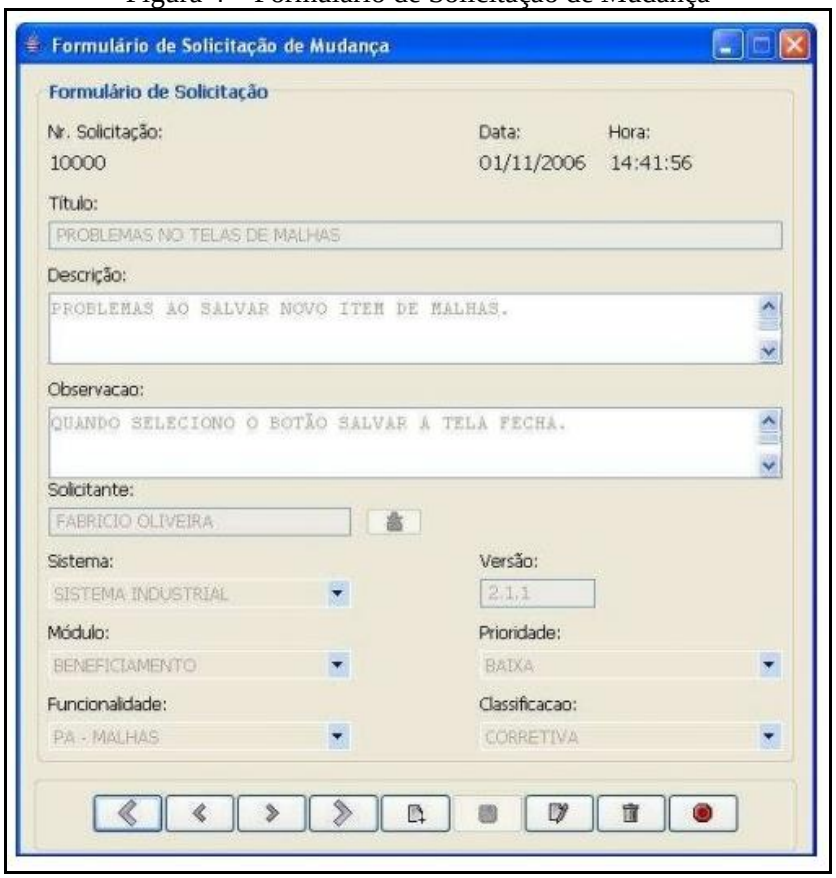

Fonte: Qualitor (2016).

### 2.3.2 Software de apoio à gerência de solicitação de mudanças

Oliveira, F. (2006), apresenta um software desenvolvido na linguagem Java que aborda um processo de gerência de solicitações de mudanças. Esse software atende as normas do ISO/IEC 15504, auxiliando as empresas na criação, definição e gerenciamento das solicitações de alteração de sistemas. O software também realiza a troca de informações entre os envolvidos no processo de mudança, permitindo que as solicitações sejam gerenciadas, acompanhadas e controladas até a sua conclusão (OLIVEIRA, F., 2006).

Na Figura 4 é apresentado o formulário de solicitação de mudança. Após o preenchimento de todas as informações pelo usuário, o sistema registra a data e hora que a solicitação foi feita (OLIVEIRA, F., 2006).

Figura 4 – Formulário de Solicitação de Mudança




Fonte: Oliveira, F., (2006).

Sendo assim um analista recebe a solicitação, dá o seu parecer a respeito e assim encaminha ao testador. O testador por sua vez aprova o teste e registra todas as informações necessárias para que a solicitação seja concluída, ou seja, um parecer do testador. A partir desse ponto, o solicitante é notificado e o caso é encerrado.

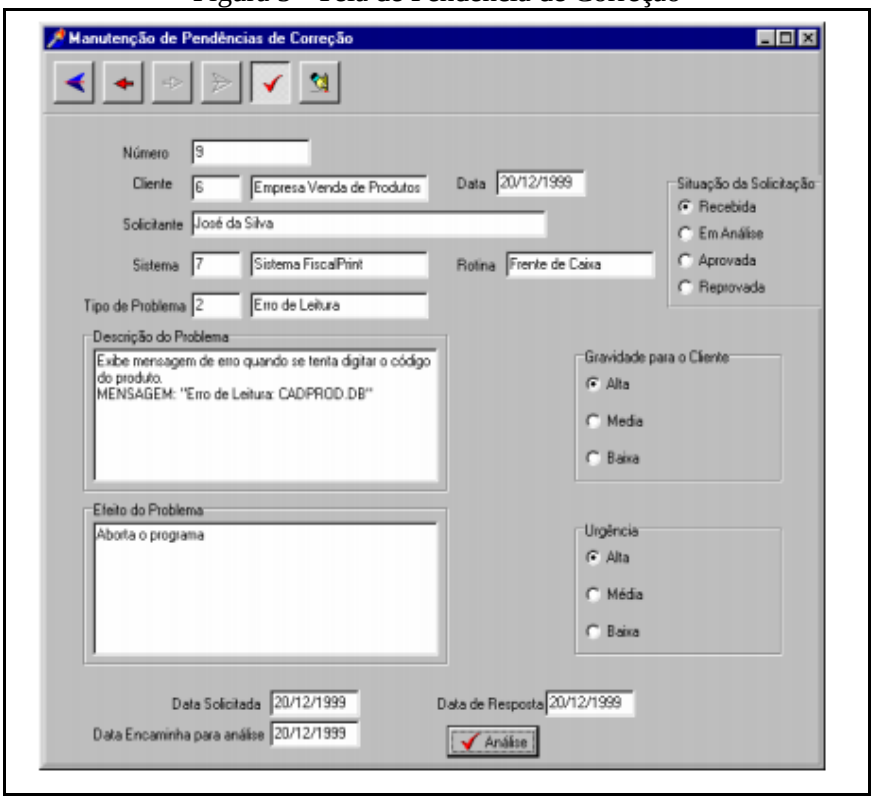
### 2.3.3 Software de apoio à manutenção de sistemas baseado em normas de qualidade

O "software de apoio à manutenção de sistemas baseado em normas de qualidade", refere-se a um sistema para uso em rede local da empresa, ou seja, uma intranet (HOPPE, 1999). Os processos são definidos de acordo com as normas de qualidade ISO/IEC 12207, ISO/IEC 9000-3 e ISO/IEC 9000-2. Na implementação e codificação do sistema foi utilizado o Delphi 3.0 como ambiente de desenvolvimento e o Paradox para armazenamento dos dados (HOPPE, 1999, p. 105).

> O software especificado e implementado mostrou-se eficaz no auxílio ao processo de manutenção de sistemas, nas etapas de solicitação e análise das pendências, bem como no registro das manutenções do sistema. Este tipo de controle ajuda a equipe de manutenção de sistemas e até mesmo os desenvolvedores de novas aplicações com feedback de problemas passados para auxiliá-los em dificuldades futuras (HOPPE, 1999, p. 105).

Na Figura 5 é apresentada uma das telas do sistema, onde contém todas as pendências de correção para a equipe de manutenção de software. Segundo Hoppe (1999, p. 74) “essa etapa é importante para que haja uma sequência de pendências a resolver e também para as solicitações mais urgentes não fiquem paradas”.

Figura 5 - Tela de Pendência de Correção




Fonte: Hoppe (1999).

O sistema também consiste no registro de todas as causas dos problemas. Sendo assim, esses registros são fundamentais para a equipe ter um controle dos problemas solucionados, além de dar a equipe um grande recurso de no futuro encontrar e solucionar os problemas mais rapidamente (HOPPE, 1999).

### 2.3.4 Correlação dos trabalhos correlatos

Esta seção apresenta um comparativo entre os trabalhos correlatos mencionados que foram apresentados. Portanto, para cada trabalho/ferramenta é demonstrado as suas respectivas características, assim como outras informações para o comparativo de ambos.

Na seção 2.3.1 foi apresentado o Qualitor. Uma ferramenta que atende a gestão do suporte com o objetivo de trabalhar na abertura e gestão de demandas para o suporte e atendimento. Um de seus pontos principais é que ele é 100% baseado na web, onde os dados, informações e o processo pode ser consultado de qualquer lugar que possua acesso à internet. Porém uma de suas desvantagens é que só atende a uma parte do suporte ao cliente.

Na seção 2.3.2 foi demonstrado um software de apoio à gerência de solicitação que auxilia empresas no seu gerenciamento de solicitações de alteração de sistema. Esse software tem por objetivo a troca de informações entre as etapas do ciclo de vida da manutenção, até a solicitação ser totalmente encerrada.

Já na seção 2.3.3 é apresentado outro software de apoio à manutenção de software mas baseado em normas de qualidade. Esse software tem potencial em controlar a equipe de manutenção de sistemas e os desenvolvedores de novas aplicações com feedback de problemas conhecidos. O Quadro 1 e Quadro 2 contemplam algumas informações sobre a correlação entre os trabalhos correlatos mencionados. O primeiro quadro define as áreas no qual o software se aplica ao processo de manutenção de software. Por sua vez, o segundo quadro define algumas características e a plataforma em que o mesmo foi desenvolvido.

Quadro 1 - Correlação: Áreas Atendidas Trabalhos Correlatos

| **Áreas Atendidas** | **Qualitor** | **Soft. Solicitação Mudança** | **Soft. Apoio à manutenção** |
|---|---|---|---|
| Suporte | X | | |
| Desenvolvimento | | X | X |
| Testes | | X | |
| Documentação | | | |

Fonte: Elaborado pelo autor.

Quadro 2 - Correlação: Características e Plataforma desenvolvida

| **Ferramenta** | **Característica** | **Plataforma** |
|---|---|---|
| Qualitor | Gestão no atendimento ao usuário. Quanto ao *service-desk* ou *help-desk.* | Linguagem PHP e banco de dados Postgress |
| Soft. Solicitação Mudança | Gestão das solicitações de mudança. | Linguagem Java |
| Soft. Apoio à manutenção | Gestão das solicitações de mudança. Auxilio ao processo de manutenção sistema e uma análise das pendencias. | Linguagem Delphi 3.0 |

Fonte: Elaborado pelo autor.

# 3 DESENVOLVIMENTO

Neste capítulo são descritos as especificações e o detalhamento do software desenvolvido, apresentando as suas características, os Requisitos Funcionais (RF), os Requisitos Não Funcionais (RNF), os Diagramas de Caso de Uso (DUC) e a sua descrição, diagrama de classes, diagrama de atividade, modelo conceitual de dados, sua implementação, assim como técnicas e ferramentas utilizadas e a operacionalidade da implementação.

## 3.1 LEVANTAMENTO DE INFORMAÇÕES

Este trabalho apresenta o desenvolvimento de um portal web para auxiliar e organizar as solicitações abertas e encaminhadas ao processo de manutenção, bem como disponibiliza ao solicitante informações de onde e como anda a execução da sua solicitação.

O sistema possui um controle de usuário para que somente pessoas autorizadas possam visualizar e alterar os dados de uma solicitação. Entretanto somente a gerência tem o poder de criar grupos de usuários (chamados de perfis) e determinar quais recursos cada usuário responsável pelo andamento da demanda poderá acessar.

O processo de gestão de solicitações é um processo que serve como um guia para ter visibilidade e controle das demandas geradas pelos usuários que utilizam o produto. O estabelecimento deste processo tem como principal objetivo possibilitar a solução e experiência em gestão de demanda adquirida na relação de diferentes projetos, visando a melhoria no produto e qualidade do atendimento.

## 3.2 ESPECIFICAÇÃO

A seguir é apresentada a especificação do sistema, contemplando os requisitos funcionais, requisitos não funcionais, além dos diagramas de casos de uso, diagrama de classes, diagrama de atividade e o Modelo Conceitual de Dados (MER). Para criar o diagrama de casos de uso, diagrama de classes e o diagrama de atividade foi utilizada a ferramenta Enterprise Architect (EA) da Sparx System, já para a criação do MER foi utilizado a ferramenta DBDesigner.

O Quadro 3 apresenta os Requisitos Funcionais (RF) desenvolvidos e sua rastreabilidade, ou seja, vinculação com o Caso de Uso (UC) associado.

Quadro 3 - Requisitos funcionais

| Requisitos Funcionais | Caso de Uso |
|---|---|
| RF01: O sistema deverá permitir o usuário efetuar o *login* no sistema. | UC01 |
| RF02: O sistema deverá permitir cadastrar uma solicitação. | UC02 |
| RF03: O sistema deverá permitir consultar uma solicitação. | UC03 |
| RF04: O sistema deverá permitir escalar a solicitação para o nível superior. | UC04 |
| RF05: O sistema deverá permitir assumir a responsabilidade de uma solicitação. | UC05 |
| RF06: O sistema deverá permitir concluir a solicitação. | UC06 |
| RF07: O sistema deverá permitir revisar a solicitação. | UC07 |
| RF08: O sistema deverá permitir finalizar testes da solicitação. | UC08 |
| RF09: O sistema deverá permitir abrir bug da solicitação. | UC09 |
| RF10: O sistema deverá permitir classificar a causa da falha da solicitação. | UC10 |
| RF11: O sistema deverá permitir cadastrar projetos. | UC11 |
| RF12: O sistema deverá permitir cadastrar usuário. | UC12 |
| RF13: O sistema deverá permitir alterar dados do usuário. | UC14 |
| RF14: O sistema deverá permitir encaminhar a solicitação para outro usuário. | UC13 |
| RF15: O sistema deverá permitir configurar o perfil do usuário. | UC14 |
| RF16: O sistema deverá permitir consultar projetos com suas respectivas solicitações. | UC15 |
| RF17: O sistema deverá permitir o gerente a consultar indicadores sobre o processo no ciclo da manutenção de software. | UC16 |
| RF18: O sistema deverá permitir comentar na solicitação. | UC17 |
| RF19: O sistema deverá demonstrar todas as solicitações pendentes em seu nome, aguardando sua ação. | UC18 |

Fonte: Elaborado pelo autor.

O Quadro 4 lista os Requisitos Não Funcionais (RNF) do sistema desenvolvido.

Quadro 4 - Requisitos não funcionais

| Requisitos Não Funcionais |
|---|
| RNF1: O sistema deve utilizar o banco de dados SQL Server 2012 ou superior. |
| RNF2: O sistema deve ser acessado através de um web *browser* (*Internet Explorer* 8.0 ou superior e Google Chrome). |
| RNF03: O ambiente de desenvolvimento será o Visual Studio 2012 ou superior. |
| RNF04: O sistema deve utilizar a linguagem C# e NET *framework* que serão utilizados no sistema de gerenciamento na *web*. |
| RNF05: O sistema deve apresentar mensagens de erro de modo evidente e intuitivo, fazendo com que o usuário identifique sua origem e como proceder após sua ocorrência. |

Fonte: Elaborado pelo autor.

### 3.2.1 Diagrama de casos de uso

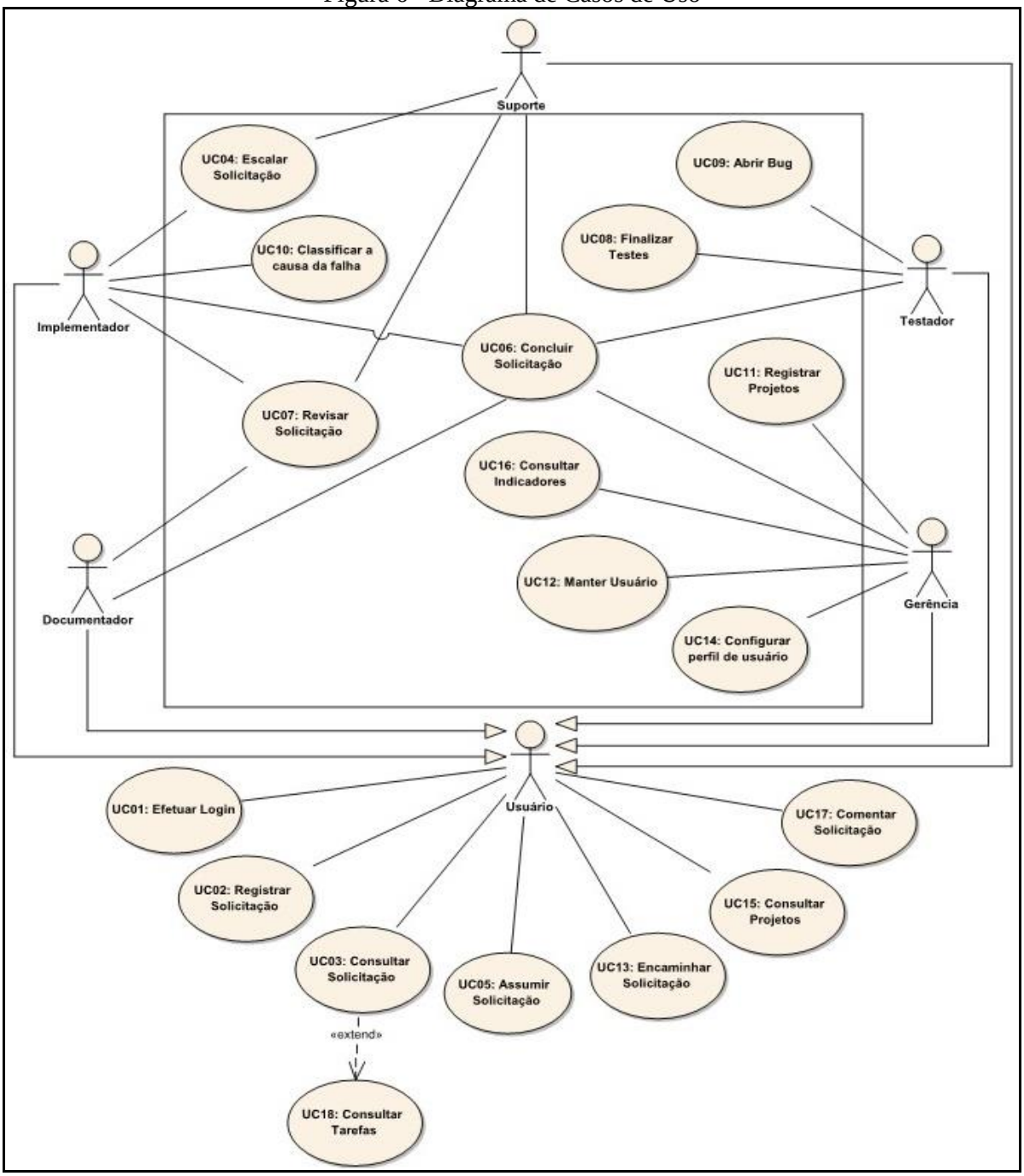
A Figura 6 apresenta o diagrama de casos de uso do sistema desenvolvido. No diagrama de casos de uso são apresentados os atores que persistem no sistema desenvolvido, sendo eles: `usuário`, `suporte`, `implementador`, `testador`, `documentador` e `gerência`.

O `usuário` é o solicitante ou proprietário de uma solicitação. Ele tem apenas privilégios de consultar projetos, registrar/consultar solicitações, efetuar *login* e assumir uma solicitação, caso necessário. Os demais atores também podem acessar a essas funcionalidades.

O ator `suporte` tem com o objetivo levar a solução, assim como diagnosticar ao desenvolvimento sobre o possível erro no sistema. Dessa forma, ele tem acesso à funcionalidade de concluir solicitação e escalar solicitação caso encontre problemas no sistema. Por sua vez, o `implementador` também tem acesso à funcionalidade de escalar solicitação, mas poderá escalar somente para o testador. Entretanto, caso o `implementador` constatar que não se refere a um erro/problema no sistema, ele poderá acessar a opção de revisar a solicitação e esclarecer a solução para o suporte.

O `testador` com a função de verificar a aderência da solução implementada tem acesso à funcionalidade de abrir bug, quando encontrado um possível erro na implementação e finalizar testes, caso contrário. Já o `documentador` tem acesso a duas funcionalidades, sendo elas: revisar e concluir solicitação. Por fim, a gerência responsável pela gestão do processo tem privilégios de registrar novos projetos, cadastrar ou editar usuários (demais atores), configurar o perfil do usuário e consultar indicadores.

Figura 6 - Diagrama de Casos de Uso




Fonte: Elaborado pelo autor.

A seguir serão apresentados alguns dos casos de uso principais, com a sua descrição geral, assim como os atores envolvidos, pré-condições, fluxo principal/alternativo e a pós-condição.

#### 3.2.1.1 UC02 - `Registrar solicitação`

Este caso de uso permite a inclusão de uma solicitação por parte do solicitante, é a etapa inicial do ciclo de vida. O Quadro 5 apresenta maiores detalhes sobre o caso de uso.

Quadro 5 - Descrição do caso de uso Registrar Solicitação (UC02)

| **Nome do Caso de Uso** | **Registrar Solicitação** |
|---|---|
| Descrição | O sistema deverá permitir cadastrar solicitação para manutenção. De posse dos dados para acesso à ferramenta de registro de solicitações, deverá o usuário proceder a abertura do chamado. |
| Ator | Usuário, Suporte, Implementador, Testador, Documentar e Gerência |
| Pré-condição | Usuário com acesso à internet, acessa o portal web.<br>Usuário solicitante deve estar cadastrado no sistema.<br>Usuário deverá estar logado no sistema. |
| Fluxo principal | Usuário efetua *login*.<br>Usuário seleciona opção – Abrir Solicitação.<br>Usuário informa todos os dados obrigatórios.<br>Sistema valida informações cadastrais.<br>Sistema armazena os dados. |
| Fluxo alternativo (a) | Dados do cadastro nulo ou inválido;<br>Mensagem - Verifique os campos obrigatórios é apresentada;<br>Campos inválidos ou em branco destacados. |
| Pós-condição | Cadastrado solicitação |

Fonte: Elaborado pelo autor.

Como é demonstrado no Quadro 5 o fluxo principal se inicializa com o *login*, no qual, permite ao usuário através da identificação por usuário e senha conectar-se ao sistema. Após efetuar o *login* o usuário entra na tela de solicitação, clica na opção abrir solicitação e assim de posse dos dados do problema grava seus respectivos dados, sendo eles nome, responsável (dado já informado pelo sistema), projeto, anexo caso necessário e a descrição do problema. O sistema valida todas as informações cadastrais e o mesmo grava na sua base de conhecimento.

O fluxo alternativo se refere a um cenário em que o usuário deixou de preencher/informar algum campo obrigatório, como por exemplo a descrição da solicitação em branco. O sistema valida os dados cadastrais e retorna uma mensagem advertindo ao usuário que o campo é obrigatório e a solicitação não foi registrada. Desta forma, a solicitação só será gravada na sua base após o campo ser informado.

#### 3.2.1.2 UC03 - `Consultar solicitação`

Este caso de uso permite a consulta das solicitações. Permite que cada membro da manutenção verifique com maiores detalhes cada uma das solicitações reportada. O Quadro 6 apresenta maiores detalhes sobre o caso de uso.

Quadro 6 - Descrição do caso de uso Consultar Solicitação (UC03)

| **Nome do Caso de Uso** | **Consultar Solicitação** |
|---|---|
| Descrição | O sistema deverá permitir consultar as solicitações abertas. Acessar a solicitação disponível sobre o produto ou projeto a fim de buscar informações do andamento do chamado em determinada área de serviços. |
| Ator | Usuário, Suporte, Implementador, Testador, Documentar e Gerência |
| Pré-condição | Usuário com acesso à internet, acessa o portal web.<br>Usuário solicitante deve estar cadastrado no sistema.<br>Usuário deverá estar logado no sistema. |
| Fluxo principal | Usuário efetua login.<br>Usuário seleciona opção – Solicitação.<br>Usuário informa os dados.<br>Sistema valida informações inseridas.<br>Sistema busca as solicitações desejadas. |
| Pós-condição | Retorna as solicitações de acordo com os dados de buscar informado. |

Fonte: Elaborado pelo autor.

No Quadro 6 o fluxo principal se inicializa com o *login*, caso não haja efetuado esse processo. Após efetuar o *login* o usuário entra na tela de solicitação e assim o sistema apresenta os campos para filtrar e localizar as solicitações de acordo com a sua necessidade, sendo eles pelo código, nome, situação, responsável e/ou por projeto. Após informar os campos o usuário deve clicar em buscar e desta forma o sistema valida os dados e busca as solicitações desejadas. Caso o usuário não informe nenhum campo, o sistema retorna todas as solicitações existentes ordenadas pelo seu código.

#### 3.2.1.3 UC06 - `Concluir solicitação`

Este caso de uso contempla a conclusão de uma solicitação por parte de um dos membros da manutenção. Sendo assim, a solicitação volta ao solicitante com as especificações e orientações informada na conclusão da solicitação. O Quadro 7 apresenta maiores detalhes sobre o caso de uso.

Quadro 7 - Descrição do caso de uso Concluir Solicitação (UC06)

| **Nome do Caso de Uso** | **Concluir Solicitação** |
|---|---|
| Descrição | O sistema deverá permitir concluir as solicitações exceto o usuário que abriu a solicitação. Identificada a solução caberá o registro das informações para realizar o encerramento da solicitação. |
| Ator | Suporte, Implementador, Testador, Documentar e Gerência |
| Pré-condição | Usuário com acesso à internet, acessa o portal web.<br>Usuário solicitante deve estar cadastrado no sistema.<br>Usuário deverá estar logado no sistema.<br>Usuário deverá estar no corpo da solicitação, ou seja, com a solicitação selecionada. |
| Fluxo principal | Usuário efetua *login*.<br>Usuário seleciona opção – Tarefa ou Solicitação.<br>Usuário seleciona a solicitação que deseja Concluir.<br>Usuário seleciona a opção de Concluir.<br>Usuário insere seu comentário na respectiva solicitação.<br>Sistema valida informações cadastrais.<br>Sistema armazena os dados. |
| Fluxo alternativo (a) | Dados nulo ou inválido.<br>Mensagem - Verifique os campos obrigatórios é apresentada.<br>Campos inválidos ou em branco destacados. |
| Pós-condição | Concluída a solicitação. |

Fonte: Elaborado pelo autor.

Como é apresentado no Quadro **7**, o fluxo principal valida se o usuário está logado no sistema, caso contrário o mesmo deverá ser realizado. Após esta etapa o usuário entra na tela de solicitação ou de tarefa e assim seleciona qual solicitação queira concluir. Desta forma, o usuário deverá informar a descrição da conclusão da respectiva da solicitação e o sistema valida essas informações, armazena os dados sobre a solicitação aberta, muda a situação para concluída e envia para o solicitante o caso reportado.

O fluxo alterativo se refere a um cenário em que o usuário deixou de preencher/informar algum campo obrigatório, como por exemplo a descrição da conclusão em branco. O sistema valida os dados cadastrais e retorna uma mensagem advertindo ao usuário que o campo é obrigatório e a solicitação não foi registrada. Desta forma, a solicitação só será concluída após o campo ser informado.

#### 3.2.1.4 UC07 - `Revisar solicitação`

Este caso de uso possibilita ao Suporte, Implementador ou Documentador revisar a solicitação caso identifique alguma irregularidade ou falta de informação sobre o problema reportado. O Quadro 8 apresenta maiores detalhes sobre o caso de uso.

Quadro 8 - Descrição do caso de uso Revisar Solicitação (UC07)

| **Nome do Caso de Uso** | **Revisar Solicitação** |
|---|---|
| Descrição | O sistema deverá permitir a revisar as solicitações. Caso um dos atores encontrar alguma inconsistência de informação, a solicitação deve ser revisada, assim voltando ao nível anterior do processo e ser esclarecido os assuntos pontados. |
| Ator | Suporte, Implementador, Documentador |
| Pré-condição | Usuário com acesso à internet, acessa o portal web.<br>Usuário solicitante deve estar cadastrado no sistema.<br>Usuário deverá estar logado no sistema.<br>Usuário deverá estar no corpo da solicitação, ou seja, com a solicitação selecionada. |
| Fluxo principal | Usuário efetua *login*.<br>Usuário seleciona opção – Tarefa ou Solicitação.<br>Usuário seleciona a solicitação que deseja Revisar.<br>Usuário seleciona a opção de Revisar.<br>Usuário insere seu comentário na respectiva solicitação.<br>Sistema valida informações cadastrais.<br>Sistema armazena os dados. |
| Fluxo alternativo (a) | Dados nulo ou inválido.<br>Mensagem - Verifique os campos obrigatórios é apresentada.<br>Campos inválidos ou em branco destacados. |
| Pós-condição | Revisada a solicitação. |

Fonte: Elaborado pelo autor.

No Quadro 8 o fluxo principal valida se o usuário está logado no sistema, caso contrário o mesmo deverá ser realizado. Após esta etapa o usuário entra na tela de solicitação ou de tarefa e seleciona qual solicitação queira revisar. Desta forma, o usuário deverá informar a descrição da revisão da respectiva da solicitação e o sistema valida essas informações, armazena os dados sobre a solicitação aberta, muda a situação e envia para o nível anterior do ciclo da manutenção.

O fluxo alternativo se refere a um cenário em que o usuário deixou de preencher/informar algum campo obrigatório, como por exemplo a descrição da revisão em branco. O sistema valida os dados cadastrais e retorna uma mensagem advertindo ao usuário que o campo é obrigatório e a solicitação não foi registrada. Desta forma, a solicitação só será revisada após o campo ser informado.

#### 3.2.1.5 UC16 - `Consultar indicadores`

Este caso de uso é restrito e disponível somente ao Gerente da manutenção, onde contempla indicadores da situação da manutenção e performance de seus processos. O Quadro 9 apresenta maiores detalhes sobre o caso de uso.

Quadro 9 - Descrição do caso de uso Consultar Indicadores (UC16)

| **Nome do Caso de Uso** | **Consultar Indicadores** |
|---|---|
| Descrição | O sistema deverá permitir o gerente visualizar indicadores que acompanharão a performance da manutenção nos seus processos. |
| Ator | Gerência |
| Pré-condição | Gerente com acesso à internet, acessa o portal web.<br>Gerente deve estar cadastrado no sistema.<br>Gerente deverá estar logado no sistema. |
| Fluxo principal | Gerente efetua *login*.<br>Gerente seleciona opção – Gerência.<br>Gerente seleciona opção – Indicadores.<br>Sistema apresenta indicadores sobre a performance e *status* da manutenção em ambos os processos. |
| Pós-condição | Retorna os indicadores sobre a manutenção de software |

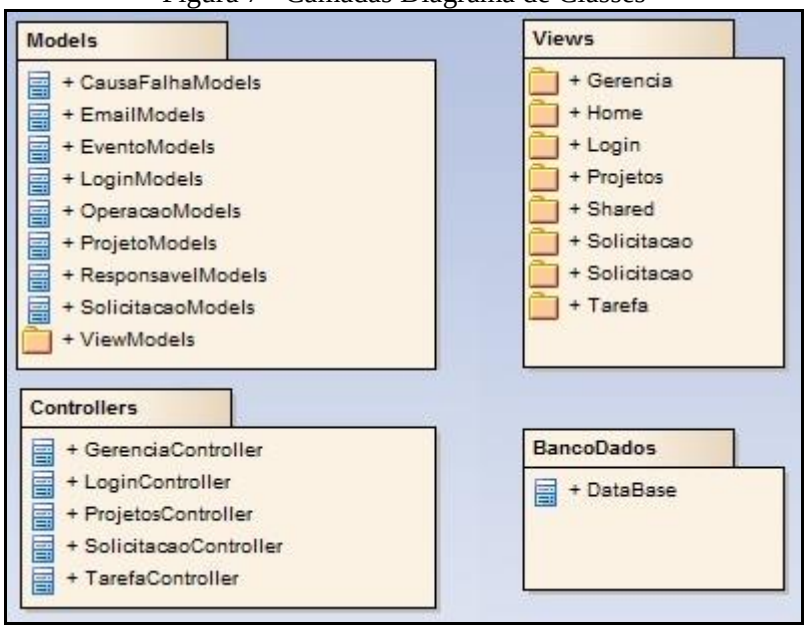
Fonte: Elaborado pelo autor.

No Quadro 9 o fluxo principal se inicializa com o *login*, caso não haja efetuado esse processo. Após efetuar o *login* o gerente, clica na opção gerência e, em seguida, na opção indicadores e o sistema ilustra todos os indicadores disponível pelo sistema, sendo eles o indicador de causa da falha das solicitações reportadas, indicador de tempo médio de cada área e por membro da manutenção de software.

### 3.2.2 Diagrama de classes

Esta seção apresenta os diagramas de classes que contemplam o sistema desenvolvido. O diagrama de classes está dívido em 4 pacotes, sendo eles `Models`, `Views`, `Controllers` e `BancoDados`, e pode ser visto na Figura 7.

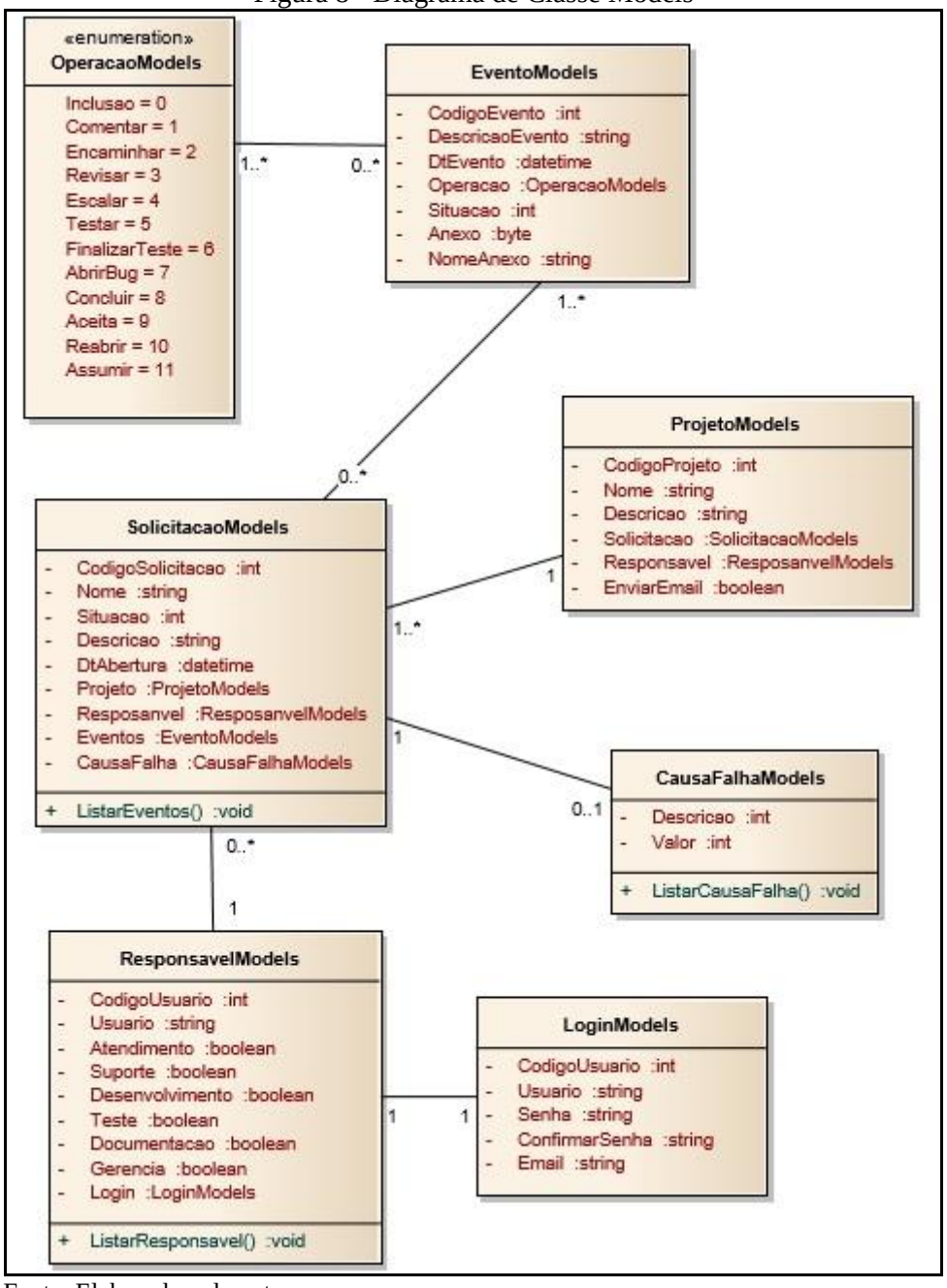
Figura 7 - Camadas Diagrama de Classes




Fonte: Elaborado pelo autor.

O pacote de `models` apresentado na Figura 7 concentra todas as classes destinadas ao negócio que é esclarecida e detalhada na seção 3.3.1. Sendo assim, as classes que se encontram neste pacote, podem ser visualizadas no diagrama de classes que está representado na Figura 8.

Figura 8 - Diagrama de Classe Models




Fonte: Elaborado pelo autor.

A classe `SolicitacaoModels` ilustrada na Figura 8 é responsável por representar a leitura e escrita de dados e também de suas validações. Ela é utilizada para montar a estrutura de objetos de uma solicitação, instanciando e interligando as camadas de objetos persistentes nas demais classes. Por sua vez, a classe `ProjetoModels`, representa os dados referentes aos

projetos da manutenção de software e suas solicitações interligadas a ele. Um projeto poderá possuir várias solicitações, porem uma solicitação deverá possuir somente um projeto na sua estrutura. Enquanto a classe `CausaFalhaModels` tem como função de tratar das causas das falhas reportadas sobre a respectiva solicitação. Além disso, toda a causa da falha deve conter uma solicitação relacionada ao mesmo.

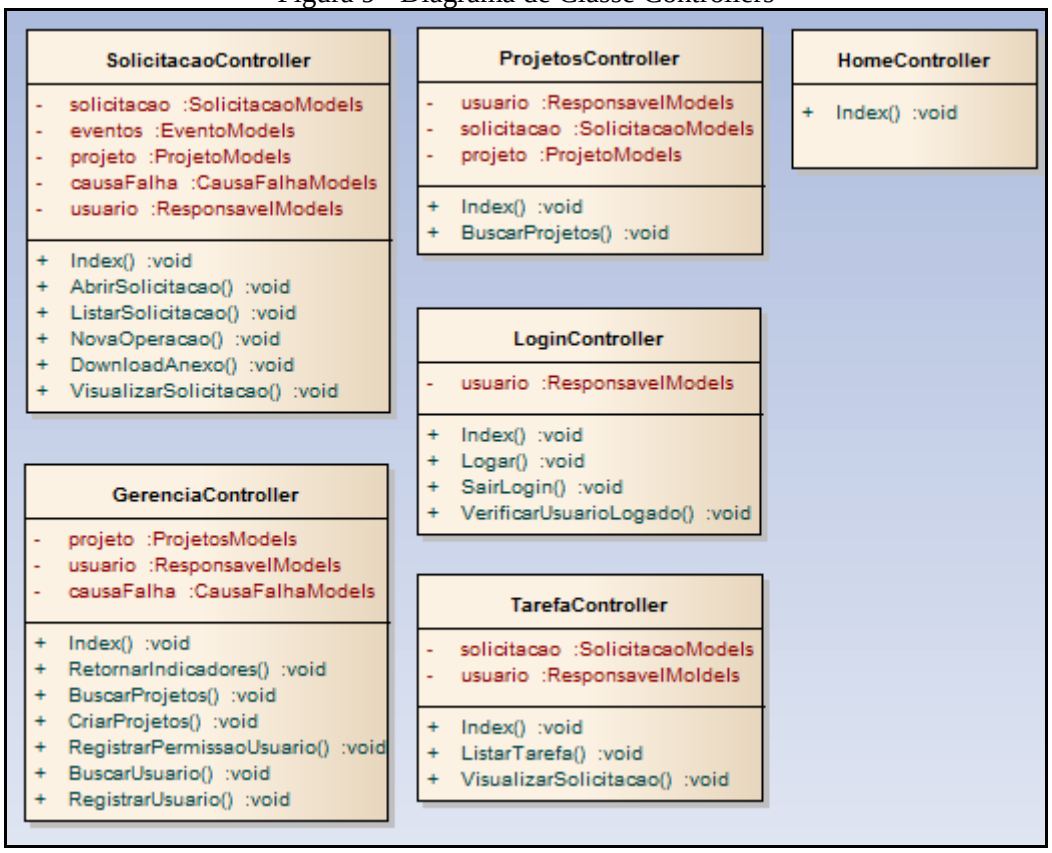
Em relação aos dados dos usuários e o seu perfil foram implementadas as classes `ResponsavelModels` e `LoginModels` que têm como responsabilidade obter os dados dos membros contemplados na manutenção. Assim como, processar, validar e associar a qualquer outra tarefa relativa ao tratamento dos dados sobre o usuário em seus processos no ciclo de vida da manutenção.

A classe de `EventoModels` juntamente com a classe `OperacaoModels` são responsáveis por implementar a função de tratar dados das operações realizadas no ciclo de vida de uma solicitação. Cada processo que é executado sobre a solicitação é processado/tratado, assim é possível ver o andamento da respectiva solicitação.

O diagrama de classes do pacote *controller* apresentado na Figura 9 contempla todas as classes que tem como função receber as requisições do usuário. Seus métodos chamados *actions* são responsáveis por uma página, controlando qual *model* usar e qual *view* será mostrado ao usuário. Sendo assim, a classe `SolicitacaoController` é destinada a manipular e controlar a *model* `SolicitacaoModels`. O carregamento da *view* correspondente é realizada pela classe `Solicitacao`, que contém as páginas em HTML na qual é exibida ao usuário em seu navegador.

Desse modo, é para as demais *controllers*, em que, `ProjetosController` controla as *models* `ProjetoModels`, `ResponsavelModels` e `SolicitacaoModels` e exibe as *views* do `Projeto`. A classe `GerenciaController` manipula as *models* `ProjetoModels`, `ResponsavelModels` e `CausaFalhaModels` e exibe as *views* da `Gerencia`. Por sua vez, a classe `HomeController` é utilizada para exibir a tela principal e a classe `TarefaController` é responsável em controlar as *models* `SolicitacaoModels` e `ResponsavelModels`. Essa classe ilustra a tela de tarefas ao usuário, no qual é demonstrado as solicitações que estão em seu nome e aguardando sua ação para andamento do processo na manutenção de software.

Figura 9 - Diagrama de Classe Controllers



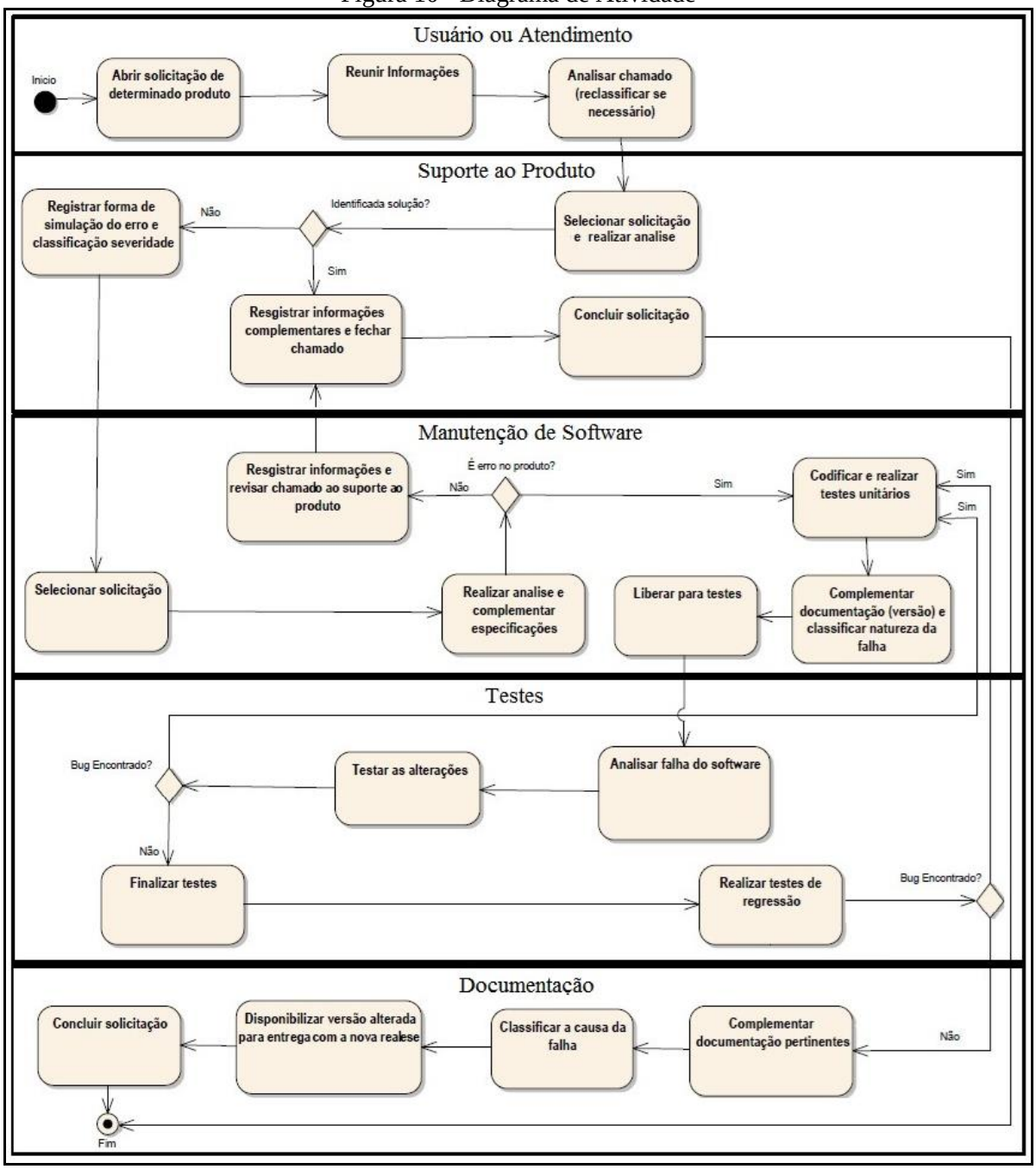

Fonte: Elaborado pelo autor.

Por fim, a classe `DataBase`, demonstrada na Figura 7 (Pacote `BancoDados`) é destinada a realizar as *queries* ao banco de dados. As *queries* são separadas em um projeto intermediário, formando uma camada no projeto correspondente à conexão com a base de dados e gerenciamento de todas as informações. Nessa classe são realizados os tratamentos e validações dos dados de forma centralizada.

### 3.2.3 Diagrama de atividade

O diagrama de atividade mostra o processo de negócio que o trabalho desenvolvido proporciona ao ciclo de vida do software dentro da fase de manutenção. O diagrama inicia da abertura da demanda de determinado projeto, passando pelo suporte, desenvolvimento, testes e assim se encerrando na conclusão da solicitação na área de documentação com a disponibilização da nova realese com a documentação pertinente. A Figura 10 apresenta o diagrama de atividade com o fluxo das solicitações propostas na manutenção de software.

Figura 10 - Diagrama de Atividade




Fonte: Elaborado pelo autor.

Na primeira área a de `Usuário ou Atendimento` como se observa na Figura 10, é contemplada a etapa inicial do processo, ou seja, é nela que são reunidas todas as informações possíveis e realizada a abertura da solicitação. Após essa etapa, o próximo processo passa a ser de responsabilidade da área de `Suporte`, o qual é responsável por buscar a solução e caso encontre-a concluir a solicitação. Caso contrário, o suporte deve registrar como simular o problema e então reportar maiores informações do mesmo.

Caso for constado pela equipe de suporte um erro no projeto ou produto, a área de `Manutenção de software` é quem tem a função de selecionar a solicitação que será tratada.

Sendo assim, ela será analisada, corrigida e serão realizados os testes unitários pelo implementador. Por fim, a solicitação será direcionada para a área de `Testes`. Entretanto, caso o implementador identifique que não se trata de um erro e sim de um esclarecimento sobre o produto, o mesmo revisará a solicitação para a área de `Suporte` que dará continuidade ao caso reportado.

A área de `Testes` tem como foco analisar a falha do software e testar a funcionalidade que sofreu alteração. Caso encontre alguma inconsistência ou bug, o mesmo deverá revisar a solicitação e encaminhar ao processo anterior (`Manutenção de software`). Portanto, se não há nenhuma inconsistência o mesmo deverá finalizar os testes e caso for necessário realizar um teste de regressão sobre o produto ou projeto.

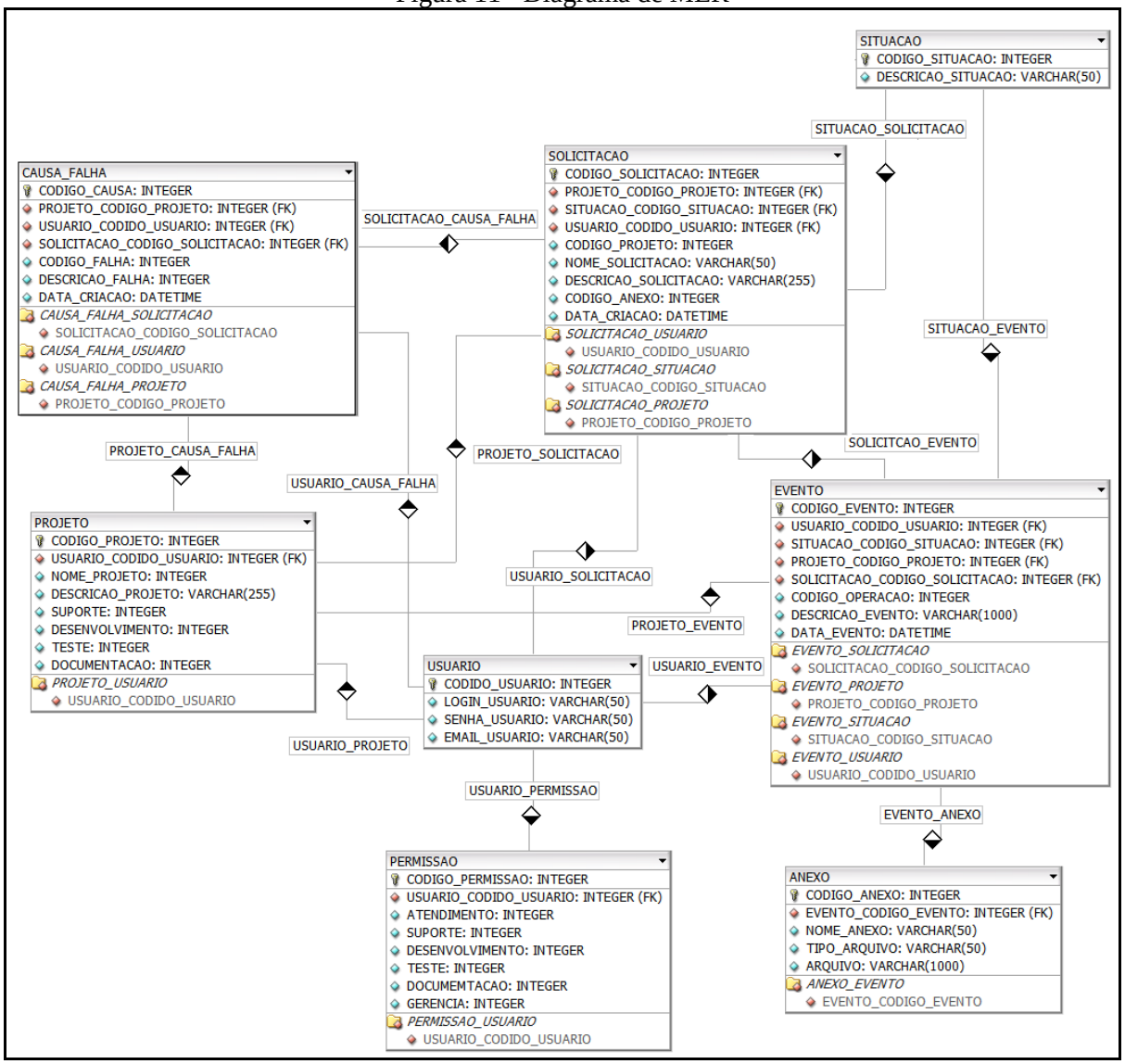
Por fim, a última área desse ciclo é a `Documentação`, que é responsável em persistir a documentação sobre o produto e deixar claro a real falha do sistema. Após a documentação, a solicitação é concluída e o processo é finalizado.

Apesar dos atores definidos no ciclo de atividades da manutenção de software, o sistema permite que um gerente defina e estabeleça quais áreas terão acesso ao ciclo da manutenção. O gerente pode escolher entre as áreas de suporte, desenvolvimento, teste e documentação. Sendo assim, é possível que as áreas sejam parametrizadas de acordo com a necessidade de cada projeto ou produto que contemplam a manutenção de software.

### 3.2.4 Modelo conceitual de dados

Esta seção apresenta o modelo conceitual de dados que representa as entidades que serão persistidas no banco de dados deste sistema. O modelo pode ser visualizado na Figura 11. Cada entidade é representada no banco de dados como uma tabela.

Figura 11 - Diagrama de MER




Fonte: Elaborado pelo autor.

A seguir é apresentada uma breve descrição das entidades criadas para o desenvolvimento do sistema:

a) `Solicitacao`: entidade que armazena as informações das solicitações abertas pelo solicitante;

b) `Situacao`: entidade que armazena as situações referente a solicitação;

c) `Evento`: entidade que armazena dados das operações realizadas no ciclo de vida de uma solicitação. Cada processo que é executado sobre a solicitação é armazenado, assim é possível ver o andamento da respectiva solicitação;

d) `Anexo`: entidade responsável pelo armazenamento dos anexos sobre os eventos das solicitações;

e) `Causa da falha`: entidade que armazena todas as causas da falha das solicitações;

f) `Projeto`: entidade que armazena todos os projetos que contemplam a manutenção;

g) `Usuario`: entidade que armazena dados dos usuários que persistem na manutenção;
h) `Permissao`: entidade responsável por armazena as permissões dos usuários sobre o ciclo da manutenção de software.

## 3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas e detalhadas as técnicas e ferramentas utilizadas para o desenvolvimento da aplicação, assim como a implementação e sua operacionalidade.

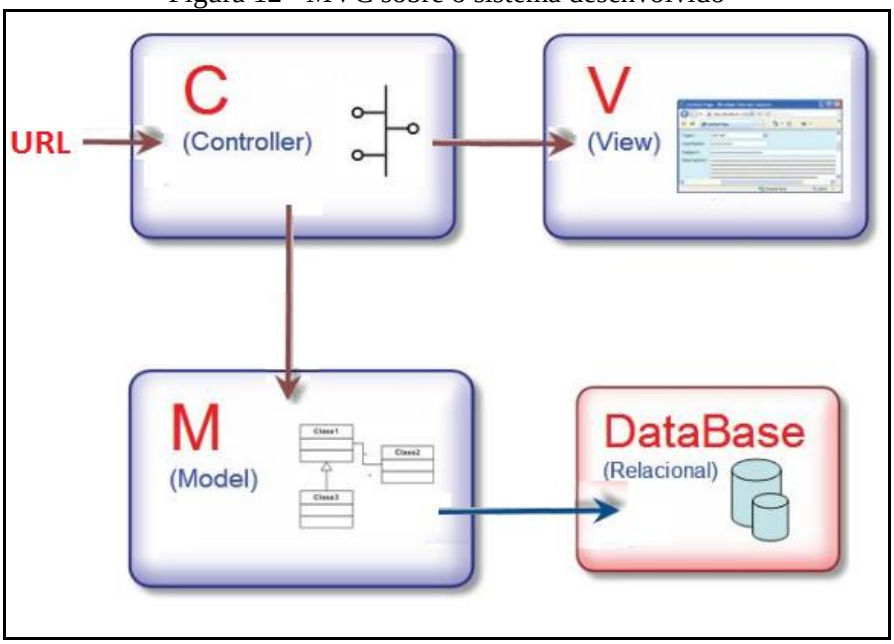
### 3.3.1 Técnicas e ferramentas utilizadas para o sistema desenvolvido

O sistema foi desenvolvido utilizando a linguagem C# com *Framework* .Net. Para o desenvolvimento aplicou-se programação Orientada a Objeto (OO) para manipulação de objetos, declaração de atributos e métodos. Para manipulação de dados vindos da base SQL Server criou-se um pacote intermediário com classes, formando uma camada no projeto que se conectam à base e gerenciam todas as informações, realizando tratamentos e validações dos dados de forma centralizada e organizada.

Por sua vez, como padrão de arquitetura de software, foi aplicado o modelo Design *Pattern Model-view-controller* (MVC), em português, modelo-visão-controlador. Na Figura 12 apresenta-se como o MVC é aplicado sobre o sistema e a Figura 13 demonstra como o projeto do sistema ficou estruturado e organizado com esse padrão adotado.

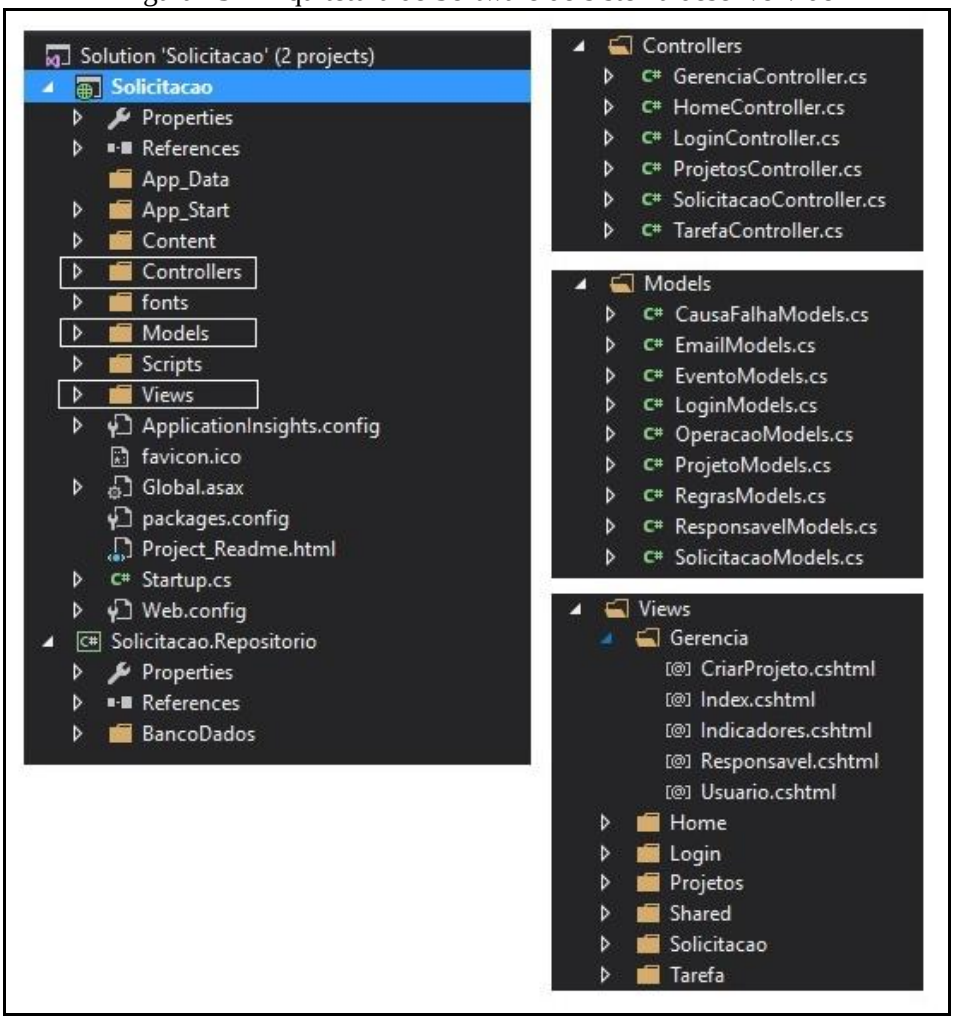
O modelo (*model*) consiste nos dados da aplicação, regras de negócios, lógica e é responsável pela manipulação de dados que se apresenta no projeto intermediário. No projeto intermediário, concentram-se os dados da base de dados. Na visão (*view*) encontram-se as saídas de representação dos dados, como as páginas em HTML criadas.

Figura 12 - MVC sobre o sistema desenvolvido




Fonte: Elaborado pelo autor.

Figura 13 - Arquitetura de Software do sistema desenvolvido




Fonte: Elaborado pelo autor.

O controlador (*controller*) que é ilustrado na Figura 13 é formado pelas classes responsáveis em realizar a mediação, convertendo-a em comandos para o modelo ou visão. Por sua vez, é o controlador que determina que resposta será enviada de volta ao usuário quando ele faz uma requisição via navegador. De forma geral, as ideias centrais por trás do MVC são a reusabilidade de código e separação de conceitos (MANTOVANI, 2015. p. 1).

### 3.3.2 Desenvolvimento do sistema

O desenvolvimento do sistema foi dividido em três camadas: na primeira foram implementadas a *model* e a *view*, na segunda à *controller* que realiza a medição e na terceira, a implementação de uma camada intermediária com classes responsáveis por se conectarem a base e que gerenciam todas as informações. A seguir são apresentados os trechos de códigos mais significativos dessas etapas.

#### 3.3.2.1 Implementação da *model-view-controller*

Como mencionado, as classes modelo (*model*) consistem nos dados da aplicação, regras de negócios, lógica e é responsável pela manipulação de dados. Por sua vez, todas as classes desse tipo estão no pacote de *models* como descrito na seção 3.2.2. Conforme o Quadro 10 a classe *model* `ProjetoModels` constitui quais propriedades são persistentes em um projeto da manutenção no sistema, sendo elas código do projeto, nome, descrição, responsável e situação. Além disso, pode-se adicionar atributos necessários ao elemento, como o `required` para indicar que o campo se torna obrigatório e `display` como o campo será apresentado na *view*.

Quadro 10 - Classe ProjetoModels

```
namespace Solicitacao.Models
{
    8 references
    public class ProjetoModels
    {
        public int CodigoProjeto { get; set; }

        [Required]
        [Display(Name = "Nome do Projeto")]

        public string Nome { get; set; }

        [Required]
        [Display(Name = "Descrição")]

        public string Descricao { get; set; }

        [Required]
        [Display(Name = "Responsável")]

        public string Responsavel { get; set; }
```

Fonte: Elaborado pelo autor.

Em seguida a *view* é representada em páginas em HTML, nela é inserida qual a *model* será composta, ou seja, quais campos serão manipulados na página. Por exemplo, a tela de criar projetos no modulo da gerência na manutenção de software, no qual a *view* `CriarProjeto` contém a *model* `ProjetoModels`, os respectivos campos que serão utilizados e assim apresentado na Quadro 11 com o preenchimento em vermelho.

Quadro 11 - View CriarProjeto

```
@model Solicitacao.Models.ProjetoModels
@{
    ViewBag.Title = "Criar Novo Projeto";
    List<string> ListResposanvel = ViewBag.ListResposanvel;
}

<h2>Novos Projeto</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
<div>
    <h4>Informações sobre o projeto</h4>
    <hr />
    <dl class="dl-horizontal">
        @using (Html.BeginForm("CriarProjeto", "Gerencia", FormMethod.Post))
        {
            @Html.ValidationSummary("", new { @class = "text-danger" })
            <div>
                <fieldset>
                    <div class="editor-label">
                        @Html.LabelFor(m => m.Nome, new { @class = "col-md-2 control-label" })
                    </div>
                    <div class="editor-field">
                        @Html.TextBoxFor(m => m.Nome, new { @class = "form-control" })
                    </div>
```

Fonte: Elaborado pelo autor.

A *controller* é responsável por receber todas as requisições do usuário. Seus métodos chamados *actions* são responsáveis por uma página. No Quadro 11, no preenchimento em laranja tem-se um exemplo de qual *actions* será utilizado na *view* e qual *model* será manipulada. Desta forma, o método `CriarProjeto` é pertinente a *controller* de `GerenciaController` como é ilustrada no Quadro 12.

Quadro 12 - Classe GerenciaController

```
[HttpPost]
0 references
public ActionResult CriarProjeto(ProjetoViewModels projeto)
{
    if (regras.VerificarUsuarioLogado())
    {
        if(ModelState.IsValid)
        {
            banco.RegistrarProjeto(projeto.Nome,
                                   regras.CodigoUsuario(projeto.Responsavel),
                                   projeto.Descricao);
            ModelState.AddModelError("Mensagem", "Projeto criado com sucesso.");
        }
        else
        {
            ModelState.AddModelError("Mensagem", "Não foi possível criar novo projeto.");
        }

        return View("CriarProjeto");
    }
    else
        return RedirectToAction("Login", "Login");
}
```

Fonte: Elaborado pelo autor.

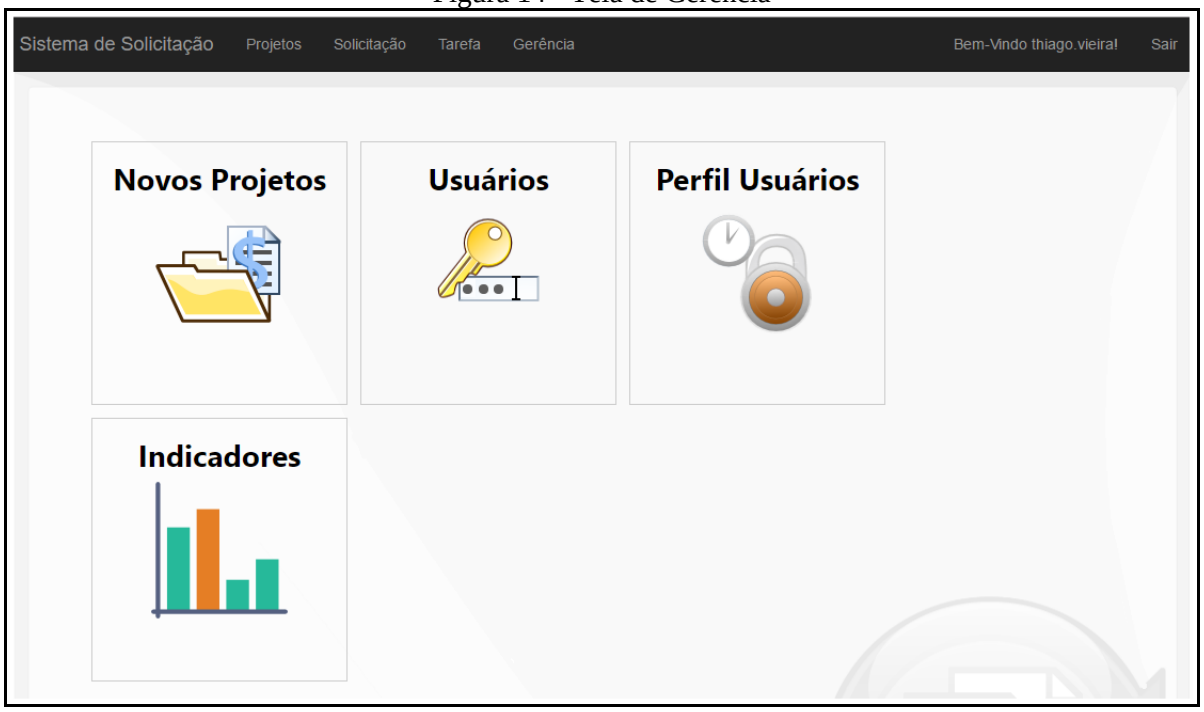
No Quadro 12 é apresentada a classe *controller* `GerenciaController` que pertence ao pacote das `Controllers`. A mesma é responsável por receber a *model* `ProjetoModels`, verificar se o usuário está logado e verificar as informações preenchidas pelo usuário. Após a validação, o projeto é registrado, passando todos os dados para o pacote de `BancoDados` e assim gravando na base de conhecimento.

### 3.3.3 Operacionalidade da implementação

Para ilustrar a operacionalidade da implementação desenvolvida, é demonstrado a seguir a etapa para efetuar o cadastramento e o acompanhamento de uma solicitação, desde os cadastros básicos do sistema até o seu encerramento. Cada membro do sistema tem funções definidas e para fazer uso deste deverá estar cadastrado no sistema. O gerente dos projetos tem acesso a todas as funcionalidades do sistema, além do seu ambiente de gerência onde é responsável por manter todos os cadastros necessários para que uma solicitação possa ser criada.

Como primeiro passo a ser seguido para a configuração do software, o gerente necessita efetuar o cadastramento de alguns dados no sistema, como projetos, usuários e estabelecer o perfil de cada membro. Na Figura 14 é apresentada a tela principal visto pelo gerente, na qual tem-se as opções para acesso aos módulos de novos projetos, usuários, perfil de usuário e indicadores.

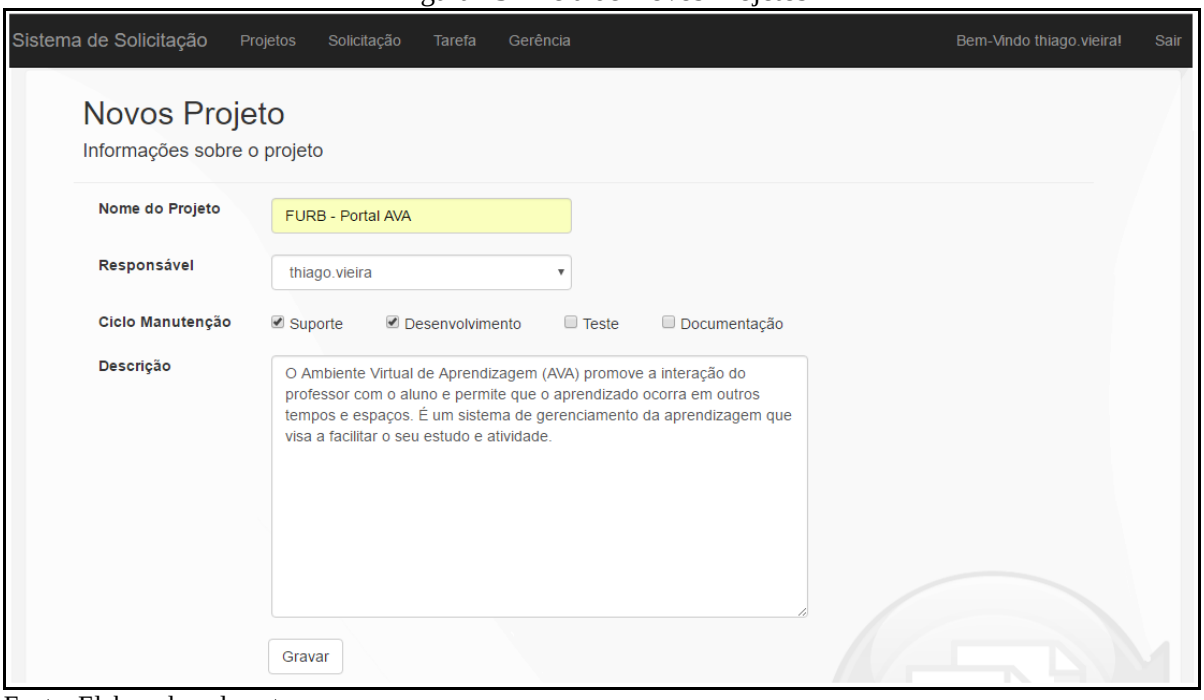
Figura 14 - Tela de Gerência




Fonte: Elaborado pelo autor.

Na Figura 15 é apresentada a tela de novos projetos, onde é possível cadastrar novos projetos que serão trabalhados no ciclo da manutenção. Assim o gerente deve informar o nome, responsável e a descrição do projeto. Além disso, o gerente deve definir e estabelecer quais áreas irão compor o ciclo da manutenção, sendo elas suporte, desenvolvimento, teste e documentação.

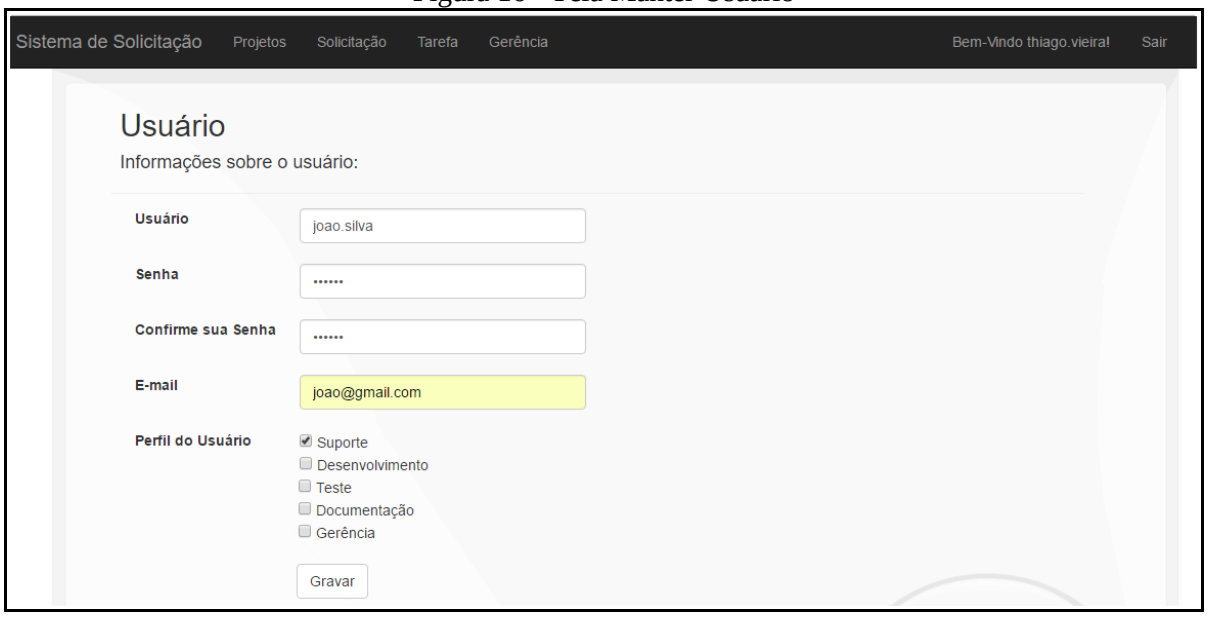
Figura 15 - Tela de Novos Projetos




Fonte: Elaborado pelo autor.

A tela de usuário é apresentada na Figura 16 e é destinada a inserir e alterar os membros contemplados na manutenção. Nessa tela, o gerente deve informar o nome do usuário, senha, e-mail e parametrizar o perfil do usuário, ou seja, qual área que esse membro irá atuar no clico da manutenção.

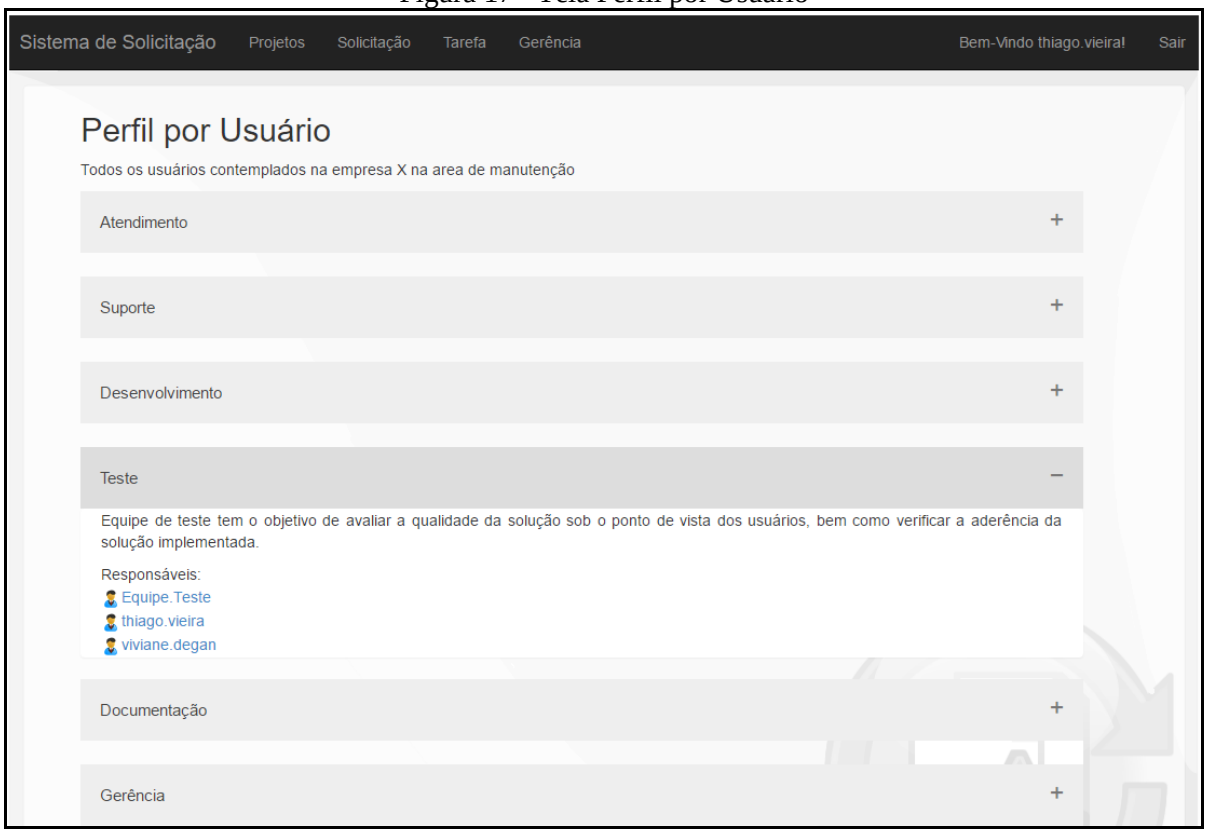
Figura 16 - Tela Manter Usuário




Fonte: Elaborado pelo autor.

A tela perfil por usuário, ilustrada na Figura 17, apresenta todos os membros e disponibiliza ao gerente um controle das áreas atuantes na manutenção. Por sua vez, ao clicar no usuário o gerente poderá inserir ou alterar qual área o respectivo membro atuará, determinando assim quais recursos o usuário poderá acessar.

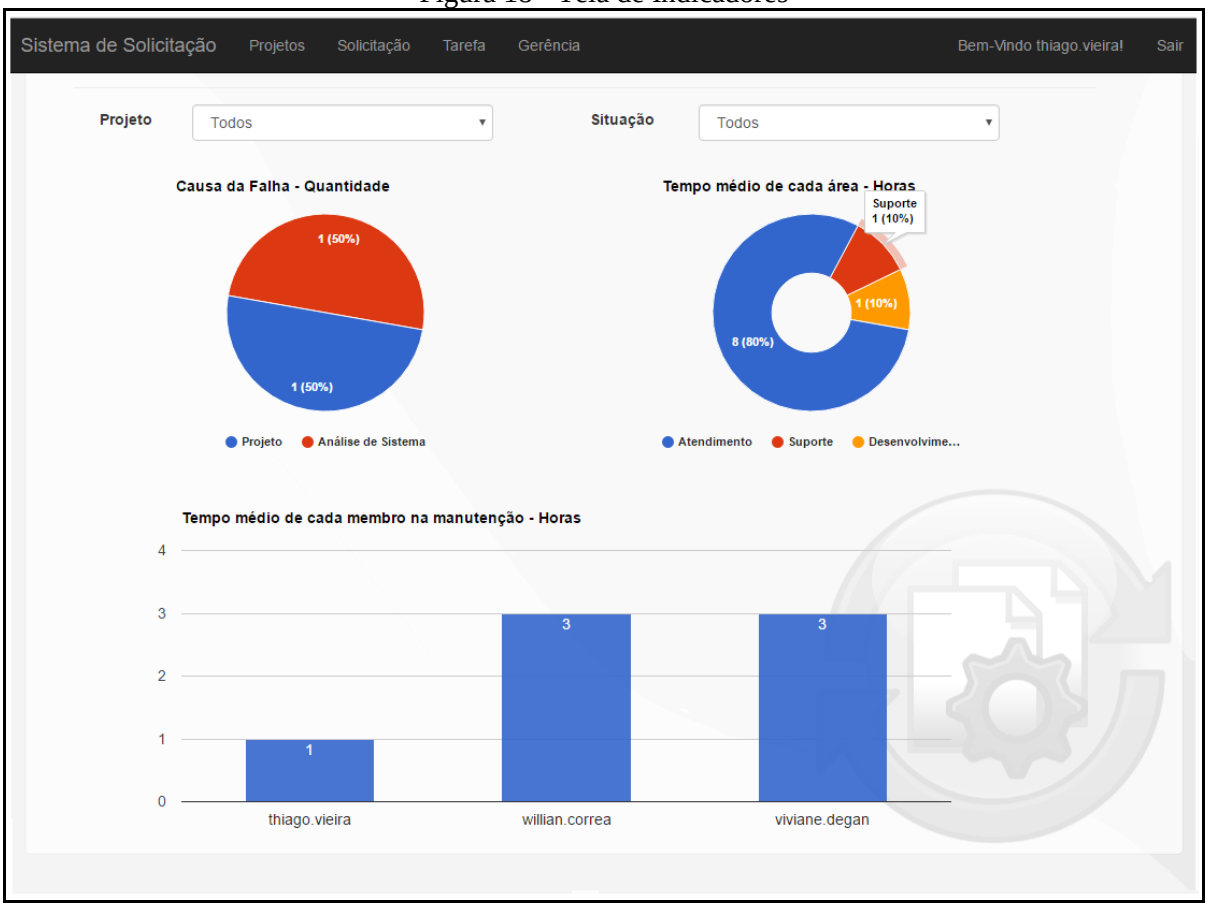
Figura 17 - Tela Perfil por Usuário




Fonte: Elaborado pelo autor.

A tela de indicadores, como é apresentada na Figura 18, é responsável por dar uma dimensão de como está a performance e a situação dos processos na manutenção de software. Nela são disponibilizados indicadores em que é possível visualizar as causas das falhas reportadas, o tempo médio de cada área, assim como o tempo médio de cada membro que exerce numa função no ciclo de manutenção. Os cálculos do tempo médio são baseados na quantidade de solicitações que o membro atuou pelas horas acumuladas exercidas na mesma. Na tela é possível filtrar por projetos e situação, permitindo que o gerente refine o resultado dos dados.

Figura 18 - Tela de Indicadores




Fonte: Elaborado pelo autor.

Na Figura 19 é apresentada a tela geral disponível a qualquer usuário que tenha acesso ao sistema e em sua página principal conta com os principais módulos. Ainda sobre essa tela, tem-se a opção de acessar os projetos que serão trabalhados no processo de manutenção. Na opção solicitação são disponibilizadas informação do andamento do chamado em determinada área de serviço e tarefa que apresenta ao usuário as solicitações que estão em seu nome, aguardando sua ação para andamento do processo.

Figura 19 - Tela Principal


Fonte: Elaborado pelo autor.

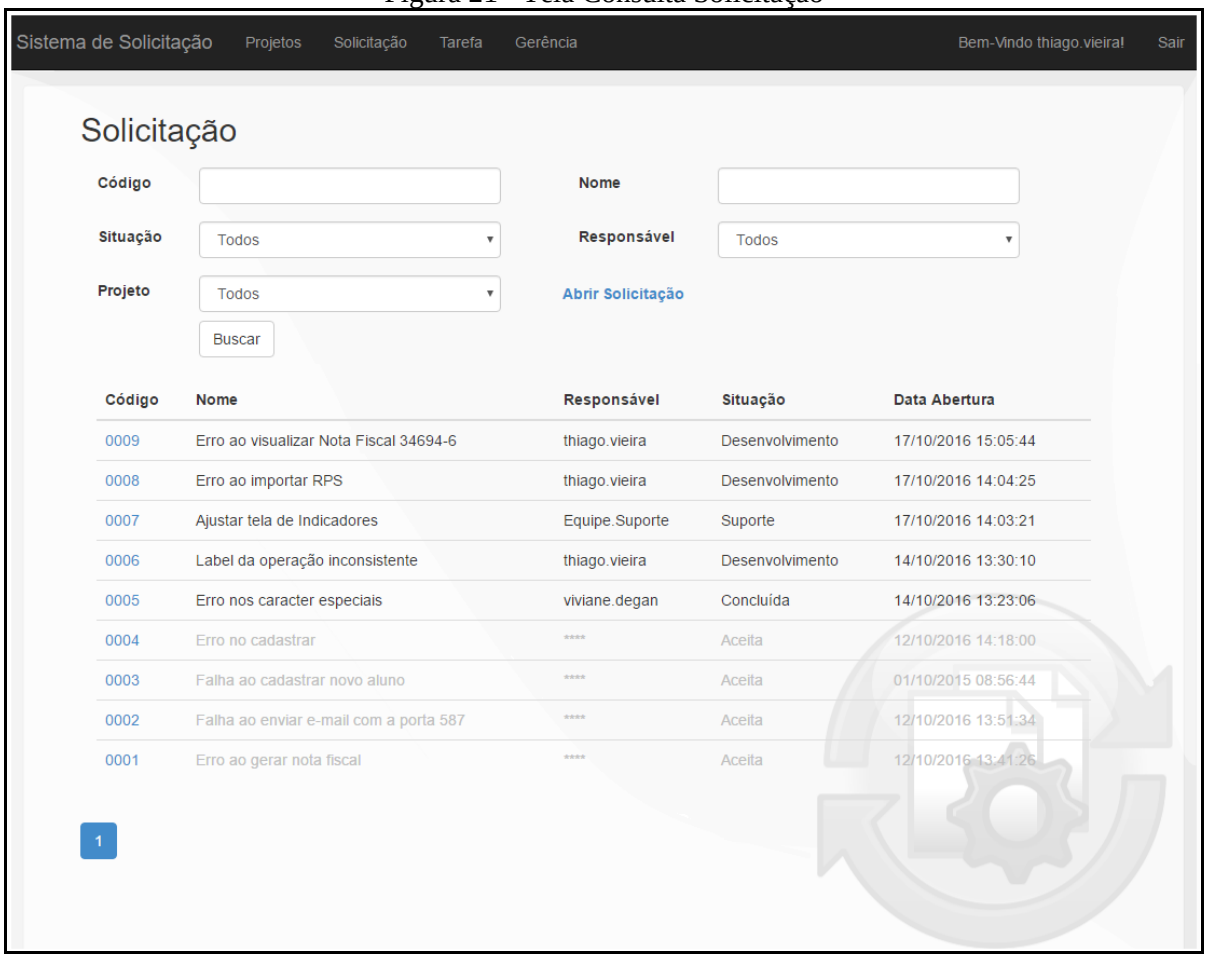
A tela de projetos ilustrada na Figura 20, está disponível para todos os usuários sem exceção, idem os projetos que a manutenção atua. Nesta tela, ainda consta alguns indicadores em relação aos projetos, sendo eles: solicitações por projetos, por área e por período. Além desses, é apresentada a descrição do projeto, disponibilizando mais informações sobre o que se refere o projeto ou produto.

Figura 20 - Tela Consulta de Projetos


Fonte: Elaborado pelo autor.

Na Figura 21 pode-se observar a tela de solicitações. Esta tela contempla todas as solicitações abertas com informativos em relação ao seu estado atual, sendo elas: situação, responsável e data de abertura. A tela disponibiliza filtros para o usuário buscar e filtrar as solicitações que deseja atuar ou visualizar.

Figura 21 - Tela Consulta Solicitação

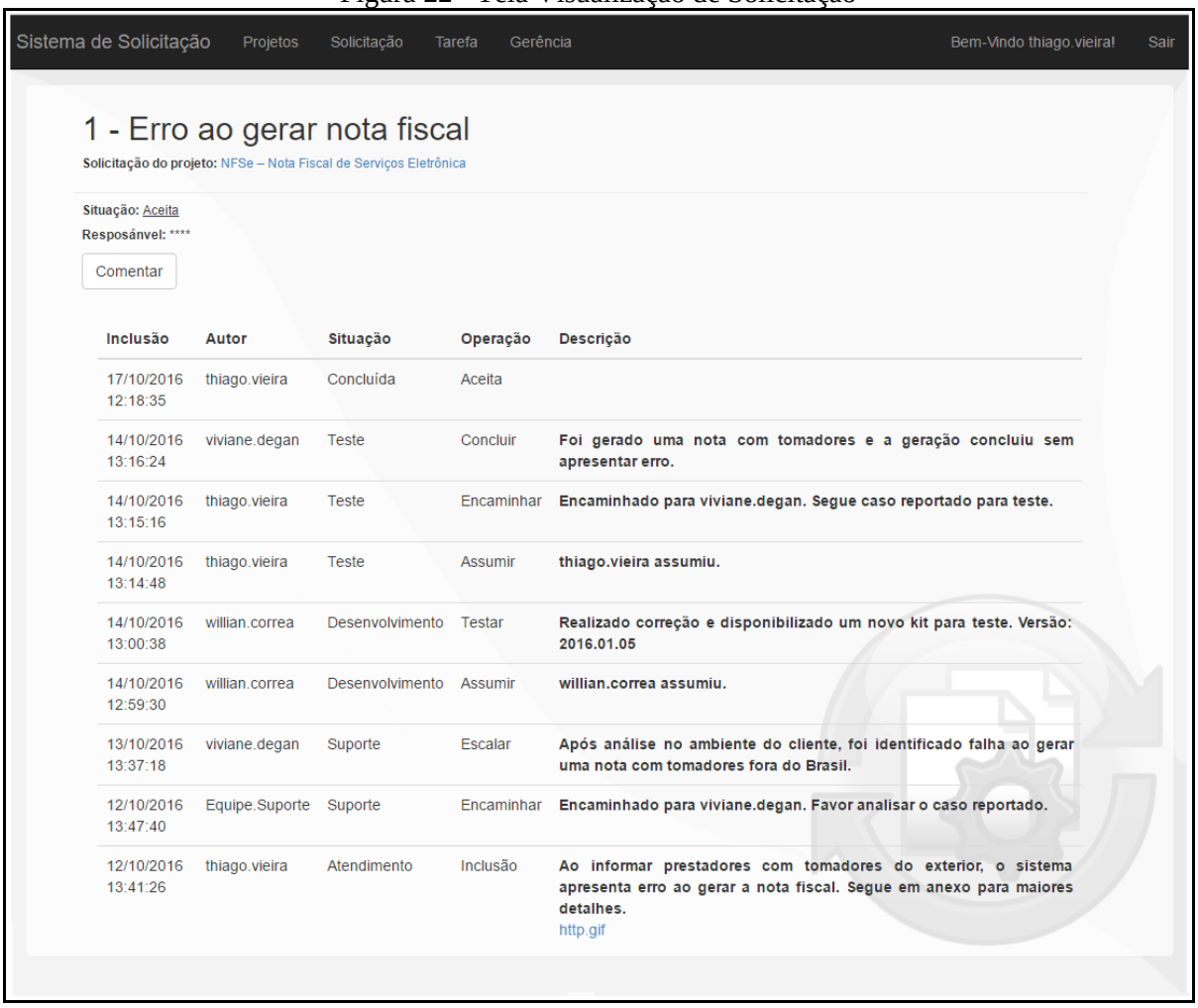



Fonte: Elaborado pelo autor.

Quando o usuário clicar no código da solicitação, o sistema redirecionará para a página de visualização da solicitação ilustrada na Figura 22. Nessa tela, serão apresentadas todas as informações referentes à solicitação, assim como sua atual situação e informações do andamento do respectivo caso reportado.

Na Figura 23 é apresentada a tela de visualização da solicitação, mas nesse exemplo específico, a solicitação encontra-se na situação em desenvolvimento. Assim, todos os implementadores poderão atuar sobre ela. Entretanto, com essas condições, algumas operações estão disponíveis para a solicitação prosseguir no ciclo da manutenção. Sendo elas a operação `Revisar`, assim a solicitação voltará ao processo anterior e `Testar` ou `Escalar` onde a solicitação avançará para o processo sucessor. Além dessas operações citadas, outras também são fundamentais para o andamento e progresso das solicitações reportadas, como: `Comentar`, `Encaminhar`, `AbrirBug`, `FinalizarTestes`, `Concluir`, `Reabrir` e `Aceitar`.

Figura 22 - Tela Visualização de Solicitação




Fonte: Elaborado pelo autor.

Figura 23 - Tela Visualização de Solicitação

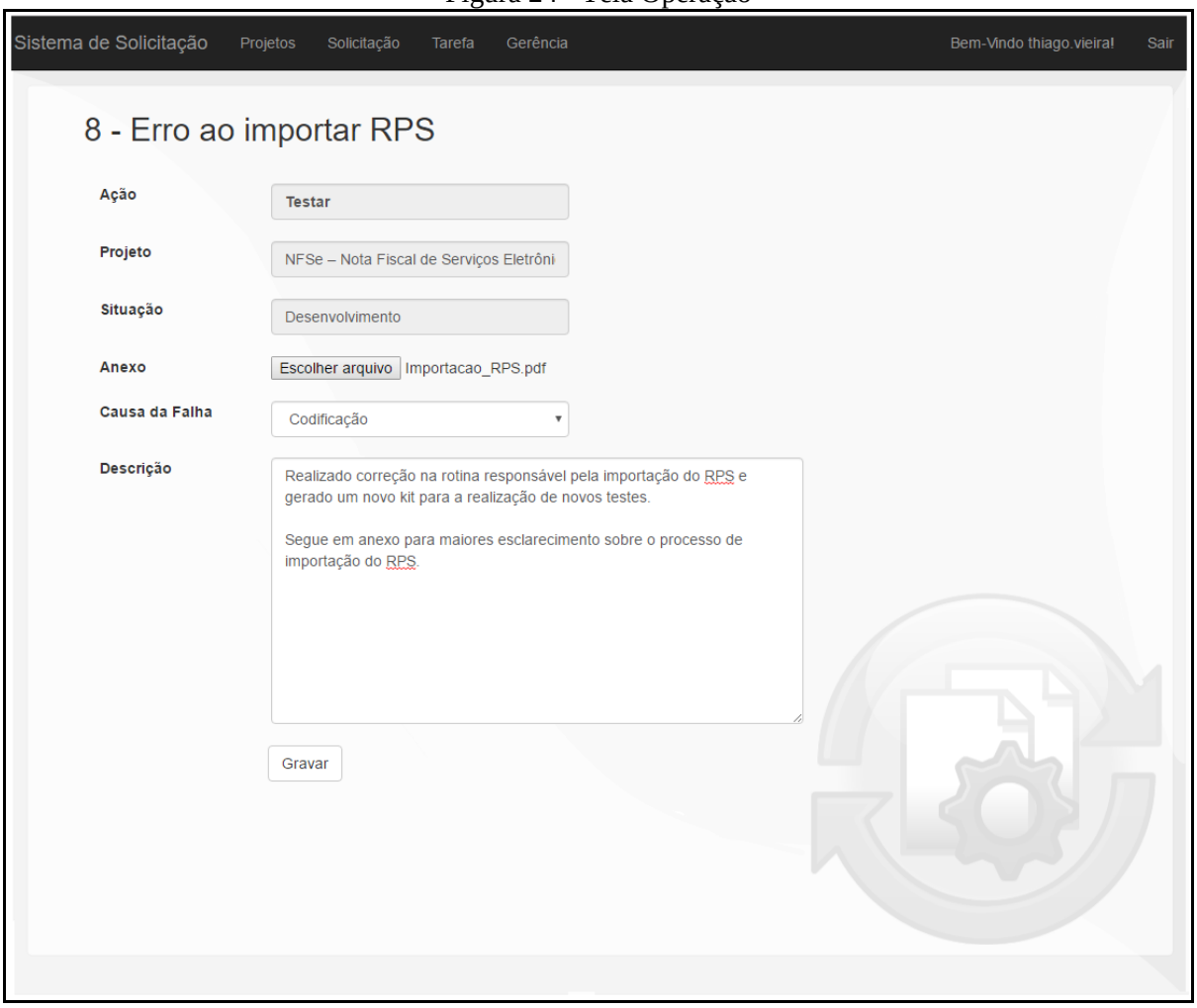

Fonte: Elaborado pelo autor.

Quando uma solicitação é escalada ou revisada de acordo com a necessidade do usuário, o mesmo passará pela tela de operações. Um exemplo que pode ser destacado é, onde

uma solicitação foi corrigida pelo implementador e passará para o próximo processo (Teste). O implementador clica na opção Testar e a tela de operação será apresentada, como ilustra a Figura 24. Sendo assim, o implementador deverá informar os dados fundamentais para que a equipe de testes possa dar continuidade à solicitação.

Esse mecanismo funciona para os demais processos, ou seja, desde a sua abertura até a sua aceitação. Todas as informações referentes aos processos serão armazenados e assim disponibilizarão informações aos gerentes sobre a performance da manutenção nos seus processos, como citado no caso da Figura 18 (Indicadores).

Figura 24 - Tela Operação




Fonte: Elaborado pelo autor.

## 3.4 RESULTADOS E DISCUSSÕES

Em relação aos trabalhos correlatos, a ferramenta descrita sobre o Qualitor serviu como base na abertura e gestão de demandas para o suporte e atendimento para determinado produto. O software de apoio à gerência de solicitação de mudança e o software de apoio à manutenção de sistemas baseado em normas de qualidade (ambas aplicações desktop), foram

utilizados como base para a troca de informações entre os envolvidos no processo e para a análise das pendências, bem como no registro das manutenções do sistema.

O diferencial do sistema desenvolvido é que se trata de uma ferramenta web, ou seja, os dados, informações e o processo pode ser consultado de qualquer lugar que possua acesso à internet. Outra característica que pode ser mencionado é a área que a ferramenta/aplicação atua sobre o processo de manutenção de software, no qual, ambos os trabalhos correlatos atendem uma parte do processo em geral, enquanto o sistema desenvolvido abrange todo o clico de vida dentro da manutenção como se pode observar no Quadro 13.

Quadro 13 - Áreas Atendidas: Trabalhos Correlatos x Sistema Desenvolvido

| **Áreas Atendidas** | **Qualitor** | **Soft. Solicitação Mudança** | **Soft. Apoio à manutenção** | **Trabalho Desenvolvido (Portal Web)** |
|---|---|---|---|---|
| Suporte | X | | | X |
| Desenvolvimento | | X | X | X |
| Testes | | X | | X |
| Documentação | | | | X |

Fonte: Elaborado pelo autor.

O sistema foi publicado em uma rede local em uma empresa do setor de software da região. No qual três membros desta empresa realizaram uma avaliação heurística proposto e definida por Nielsen (Um dos pioneiros da avaliação heurística). A avaliação heurística é um método de inspeção de usabilidade em que a avaliação é realizada com base em um conjunto de heurísticas, em busca de problemas que prejudiquem a usabilidade (NIELSEN; MACK, 1994 apud MANTAU et al; 2013, p. 1).

O perfil dos avaliadores é de diferentes áreas que atuam, no qual um dos avaliadores atua na área de suporte ao cliente e os demais na área de desenvolvimento e qualidade de software. O processo de avaliação e coleta dessas informações se deu via um formulário elaborado pelo autor, sendo que a criação deste formulário foi utilizada as ferramentas do Google que disponibiliza sem custo. O Quadro 14 contempla os questionários que foram aplicados neste trabalho.

Quadro 14 - Questionário da heurística

| | |
|---|---|
| **H1: Visibilidade do sistema** | O Sistema fornece um feedback adequado aos usuários. Mantém-se o usuário informado sobre o que está ocorrendo? |
| **H2: Compatibilidade do sistema com o mundo real** | Sistema utiliza termos familiares ao usuário ao invés de termos orientados ao software? |
| **H3: Consistência e mapeamento** | A interação do usuário com o sistema está de acordo com o contexto? |
| **H4: Reconhecimento ao invés de memorização** | O sistema propõe de instruções visíveis ou ser de fácil recuperação quando necessárias? |
| **H5: Flexibilidade e eficiência de uso** | Permitir que os usuários configurem o sistema de acordo com a necessidade do contexto? |
| **H6: Design estético e minimalista** | Exibe apenas as informações que sejam importantes e necessárias? |
| **H7: Gerenciamento de erros** | O sistema apresenta as mensagens de erro de forma clara, indicando o problema e sugerindo uma solução? |
| **H8: Facilidade de entrada, visualização e tela** | O sistema exige e apresenta apenas informações cruciais sobre o sistema? |
| **H9: Convenções estéticas, sociais e privativas** | O sistema propõe e deixa claro que as informações do usuário estão seguras? |
| **H10: Fornece comunicação de artefatos compartilhados** | O sistema disponibiliza informações sobre as ações dos outros usuários? |
| **H11: Fornecer proteção** | O sistema propõe de um mecanismo de segurança? |
| **H12: Gerenciamento de colaboração** | O sistema possibilita o trabalho em conjunto no processo de manutenção de software? |

Fonte: Elaborado pelo autor.

O objetivo dessa avaliação é identificar problemas de usabilidade do sistema e o seu grau de gravidade, como: Baixa (1), Média (2), Alta (3) e Altíssima (4). A avaliação heurística encontrou 2 problemas de usabilidade. Por sua vez, a distribuição dos problemas encontrados, conforme apresentado na Tabela 1, indica que os principais problemas identificados estão relacionados aos aspectos do gerenciamento de erros (H7) e compatibilidade do sistema com o mundo real (H2).

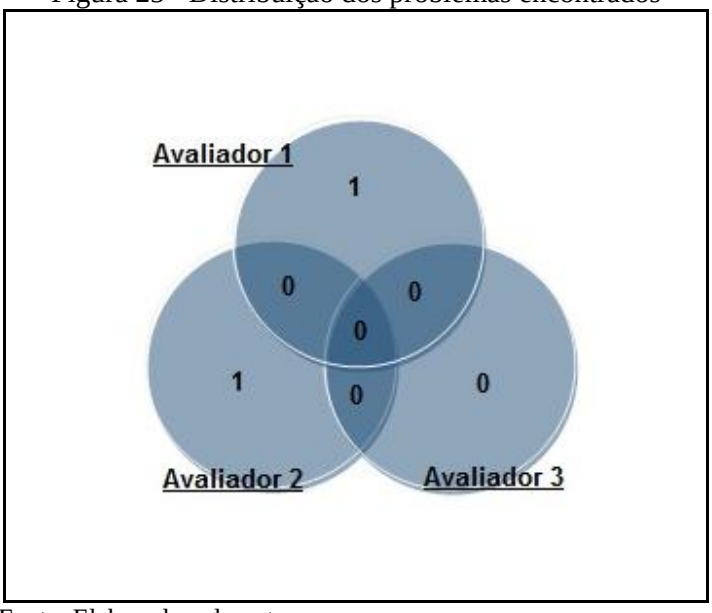
Tabela 1 - Resultado das heurísticas, problemas e gravidade

| Heurística | Problema | Gravidade |
|---|---|---|
| H1 | -- | -- |
| H2 | 1 | 2 |
| H3 | -- | -- |
| H4 | -- | -- |
| H5 | -- | -- |
| H6 | -- | -- |
| H7 | 2 | 2 |
| H8 | -- | -- |
| H9 | -- | -- |
| H10 | -- | -- |
| H11 | -- | -- |
| H12 | -- | -- |

Fonte: Elaborado pelo autor.

Não foi encontrado nenhum problema de gravidade 4 e 3, no qual considerado um desastre de usabilidade. Ambos os problemas encontrados nas heurísticas H2 e H7 são de média gravidade. Entretanto, para as demais heurísticas não foram encontrados problemas. Sendo assim, a Figura 25 apresenta a distribuição dos problemas encontrados pelos avaliadores.

Figura 25 - Distribuição dos problemas encontrados




Fonte: Elaborado pelo autor.

Como demonstrado na Figura 25, dois avaliadores encontraram problemas distintos, sendo que um dos avaliadores não encontrou nenhum problema, tendo todos os requisitos de

usabilidade atendidas. O problema reportado pelo avaliador 1 se refere a compatibilidade do sistema com o mundo real (H2), no qual o problema se trata da exibição das operações na visualização de uma solicitação. Segundo o avaliador o sistema está apresentando a descrição da operação de maneira inconsistente, praticamente numa linguagem do software. Como por exemplo, a operação "Finalizar Teste" que ficava como `FinalizarTeste`, como se fosse um atributo sendo exibido no grid de uma visualização de uma solicitação. Desta forma, o problema reportado pelo avaliador foi tratado pois foi implementado o ajuste no sistema.

O avaliador 2 identificou problema no H7 que se refere ao gerenciamento de erros. Por sua vez, o problema foi apresentado e exibido na abertura da solicitação, no qual, segundo o avaliador quando é incluído uma descrição que contém caracteres especiais ocorre erro, porém o erro não sugere o que deve ser feito. Sendo assim, o problema identificado pelo respectivo avaliador também foi tratado e implementado o ajuste no sistema.

Com a realização da avaliação heurística sobre este trabalho, foi possível avaliar a usabilidade e a eficiência do sistema. Contudo, algumas heurísticas não foram atendidas, mostrando que há detalhes a serem melhorados no sistema em questão.

# 4 CONCLUSÕES

Neste trabalho é apresentado um sistema para atender os segmentos que visam planejar e controlar a manutenção de software em uma organização com vários produtos ou projetos. Pode-se afirmar que os objetivos foram totalmente cumpridos, pois a aplicação promove a troca de informações entre os envolvidos no processo de manutenção, permitindo que as solicitações sejam gerenciadas, acompanhadas e controladas até a sua conclusão. Dessa forma, é possível garantir o andamento e a integridade das solicitações dentro do ciclo de vida na manutenção. Além disso, a aplicação disponibiliza indicadores sobre a performance e status da manutenção em todos os processos e projetos.

A utilização do Visual Studio com a linguagem C# e .NET *framework* viabilizou o desenvolvimento mais ágil e o conhecimento do autor sobre as ferramentas e a linguagem permitiu um maior grau de produtividade em relação as estruturas de dados utilizadas. Como *container* e base de dados foi escolhido o SQL Server, software cuja principal função é a de armazenar e recuperar dados solicitados por outras aplicações de software, seja aqueles no mesmo computador ou aqueles em execução em outro computador através de uma rede.

O objetivo proposto por este trabalho foi atingido uma vez que foi desenvolvido um sistema que contempla os requisitos identificados. O objetivo específico que tinha por necessidade o acompanhamento e gerenciamento dos processos no ciclo de vida da manutenção de software por cada um dos papéis, foi atingido, pois o sistema desenvolvido possibilita a troca de informações entre os envolvidos no processo de manutenção, permitindo que as solicitações sejam gerenciadas, acompanhadas e controladas até a sua conclusão.

O segundo objetivo específico que tinha como necessidade disponibilizar indicadores que acompanharão a performance da manutenção nos seus processos, foi atingido, pois o sistema disponibiliza ao gerente da manutenção de software um modulo dedicado a obter esse resultado, no qual apresenta indicadores sobre a performance e status da manutenção em ambos os processos e projetos. O último objetivo específico que tinha como necessidade de garantir o andamento e a integridade da gestão das solicitações em determinada área de serviço, como suporte, desenvolvimento, teste e documentação, foi atingindo, no qual o sistema proporciona o trabalho em equipe, através de um mecanismo de controle de solicitações, que notifica via e-mail e autoriza o registro das demais informações aplicáveis a cada etapa do processo. Sendo assim, este mecanismo busca assegurar a agilidade e o compromisso com o cumprimento dos prazos em todas as etapas do processo da manutenção de software.

A contribuição deste trabalho é permitir que qualquer organização possa melhorar as áreas de comunicação e troca de informações dentro de cada área contemplada na manutenção de software, sendo que um dos problemas que a equipe de manutenção enfrenta está relacionada ao gerenciamento das atividades da equipe (WEBER, 2014). Com a utilização do sistema, esperasse uma facilidade na visibilidade do andamento dos processos na manutenção de software em uma organização, além de um acompanhamento mais simples e fácil de ser gerenciado e controlado. Buscando prover uma agilidade no processo, e assim, conseguir um ganho significativo de tempo, que poderá ser utilizado para aumentar a produtividade.

Outro item importante é que os gerentes responsáveis pela gestão da manutenção podem monitorar e visualizar a performance de cada membro em suas respectivas áreas, além de um diagnóstico das causas das falhas em seus projetos ou produtos.

Contudo as solicitações abertas no ciclo de manutenção do software apresentam limitações, sendo:

a) não é possível agrupar ou vincular solicitações com o mesmo problema reportado;
b) as solicitações não possuem severidade, ou seja, não se sabe qual solicitação deverá ser priorizada.

## 4.1 EXTENSÕES

Como sugestões para possíveis extensões ao trabalho desenvolvido citam-se:

a) gerar novos relatórios para auxiliar nas estatísticas e no gerenciamento das solicitações;
b) agregar o modelo de acordo de nível de serviço (SLA) sobre as solicitações reportadas;
c) desenvolver um mecanismo capaz de identificar o membro mais adequado para o tratamento de determinada atividade;
d) permitir agrupar solicitações com o mesmo problema reportados;
e) apresentar sugestões de correções com base em problemas semelhantes encontrados em outras solicitações;
f) permitir que os profissionais troquem mensagens através do sistema, de modo que as conversas relacionadas a uma solicitação fique armazenada no sistema;
g) alertar os profissionais quando houver uma nova solicitação que seja correspondente ao seu perfil.

**REFERÊNCIAS**

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 12207: **Tecnologia da informação** – processo de ciclo de vida de software. Rio de Janeiro, 1998.

BELLIN, David. **Manutenção de software**: guia para administração de pequenos sistemas. Sao Paulo: Makron Books, 1993. Xviii, 229p.

BERNARDI, Altair. **Software de auxílio à implantação da norma ISO/IEC 12207 – processos do ciclo de vida do software**. 2003. 120 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BERTOLDI, Sérgio Cozzetti et al. **Documentação essencial para manutenção de software II**. In: IV Workshop de Manutenção de Software Moderna (WMSWM), Porto de Galinhas, PE, 2007.

CORDEIRO, Marco Aurélio. **Manutenibilidade de Software**. Curitiba, 2009. Disponível em: <http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=132>. Acesso em: 27 de abril 2016.

ESPINDOLA, Rodrigo Santos; MAJDENBAUM, Azriel; AUDY, Jorge Luiz Nicolas. **Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software**. 2012. 12 f. Monografia (Especialização) - Curso de Ciência da Computação, Faculdade de Informática Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2012.

GRIMHEDEN, Martin Edin. **Can agile methods enhance mechatronics design education?**. Livro Online, 2013. 973 p. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957415813000044>. Acesso em: 10 out. 2015.

GUEDES Gildásio. **Interface Humano Computador**: prática pedagógica para ambientes virtuais. Livro Online, 2015. 217 p. Disponível em: < http://cead.ufpi.br/conteudo/material_online/disciplinas/video/livro_gildasio.pdf >. Acesso em: 28 out. 2015.

HOPPE, Charles. **Software de apoio a manutenção de sistemas baseado em normas de qualidade**. 1999. 110 f. TCC (Graduação) - Curso de Bacharelado em Ciências da Computação, FURB - Universidade Regional de Blumenau, Blumenau, 1999.

IEEE, Institute. **IEEE Standard for Software Maintenance**. New York: Institute of Electrical and Eletronic Engineers. Inc., 1998, 52p.

ISO. **ISO/IEC 14764** – Software Maintenance. Genebra: International Organization for Standardization, 1999, 38p.

JALOTE, P. **An integrated approach to software engineering.** 3. ed. New York: Springer, 2005, 566p.

KNIBERG, H; SKARIN, M. **Kanban e scrum obtendo o melhor de ambos**. C4Media. Estados Unidos, 2009

LAUDON, K. **Management information systems**: managing the digital firm. 7 ed. Upper Saddle River: Prentice Hall, 2002.

MACÊDO, Ana Bárbara Lins de; SPÍNOLA, Rodrigo. **Ciclos de Vida do Software**: Engenharia de Software Magazine 36. Devmedia, Rio de Janeiro, p.1-10, 06 nov. 2016. Mensal. Disponível em: <http://www.devmedia.com.br/ciclos-de-vida-do-software-artigo-revista-engenharia-de-software-magazine-36/21099>. Acesso em: 06 nov. 2016.

MANTAU, Márcio J. et al. **Avaliação heurística para groupwares móveis**: um estudo de caso utilizando um audience response system. Joinville, 2013.

MANTOVANI, Ricardo. **ASP NET** – Introdução ao framework MVC – C#. 2015. Disponível em: <https://desenvolvimentoaberto.org/2015/05/14/asp-net-introducao-ao-framework-mvc-c/>. Acesso em: 25 out. 2016.

MAZZOLA, Vitório Bruno. **Engenharia de software**: conceitos básicos. 2010. Disponível em: <https://jalvesnicacio.files.wordpress.com/2010/03/engenharia-de-software.pdf >. Acesso em: 20 de maio 2016.

OLIVEIRA, Fabrício. **Software de apoio a manutenção de sistemas baseado em normas de qualidade**. 2006. 74 f. TCC (Graduação) - Curso de Bacharelado em Ciências da Computação, Furb - Universidade Regional de Blumenau, Blumenau, 2006.

OLIVEIRA, Stefano Petrini; MUNIZ, Jorge. **Aplicação do scrum em serviços**: análise em uma fabricante de aeronaves. 2014. 59 f. Monografia (Especialização) - Curso de Sistema de Informação, Universidade Estadual Paulista, Guratinguetá, 2014.

PÁDUA, Paula Filho Wilson. **Engenharia de software**: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2011.

PERINI, Luis Cláudio Perini, Marco Ikuro Hisatomi, Wagner Luiz Berto. **Engenharia de Software**. São Paulo: Pearson Education do Brasil, 2009.

PRESSMAN, Roger S. **Engenharia de Software**, 6 ed. São Paulo. McGraw Hill/Nacional, 2006.

_____. **Engenharia de Software**: Uma abordagem Profissional, 7 ed. São Paulo. McGraw Hill/Nacional, 2011.

QUALITOR. Constat S/A. **Conheça o Qualitor**. Disponível em: <http://www.qualitor.com.br/site/content >. Acesso em: 30 de março 2016.

RAMOS et al. **Modelo cascata ou clássico.** 2010. Disponível em: <http://modelocascata.blogspot.com.br/>. Acesso em: 08 outubro 2016.

ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. **Qualidade de software**: teoria e prática. São Paulo: Prentice Hall, 2001. 303 p.

ROYCE, W. **Software project management** - A unified framework. EUA: Addison Wesley Longman, 1998.

SANTOS, Jads Victor Paiva. **Uso do kanban em um processo de gestão de demandas de manutenção de software por terceiros para um órgão público federal brasileiro**. 2014. 113 f. TCC (Graduação) - Curso de Bacharelado em Engenharia de Software, Universidade de Brasília, Brasília, 2014.

SILVA, Salmo Roberto; PIRES, Daniel Facciolo; CARVALHO NETO, Silvio **Scrum**: um guia prático no gerenciamento de projetos. 2015. Disponível em: <http://periodicos.unifacef.com.br/index.php/resiget/article/view/1011/771>. Acesso em: 20 de maio 2016.

SIMAS et al. **Processos Ágeis** - Aprenda o que são processos ágeis**.** 2014. Disponível em: <http://www.csi.uneb.br/engenharia_de_software/anexos/Artigo-ProcessosAgeis.pdf/>. Acesso em: 18 outubro 2016.

SOMMERVILLE, Ian. **Engenharia de software**. 8. ed. São Paulo: Pearson Education do Brasil, 2007. 552p.

SOUZA, Hugo Vieira L. **Engenharia de software introdução aos processos de software**: modelos e ciclo de vida de software**.** 2015. Disponível em: < http://docplayer.com.br/13404284-Engenharia-de-software.html/>. Acesso em: 25 outubro 2016.

TEIXEIRA, Aguinaldo Aragon. **Fábrica de software**. São Paulo: Atlas, 2007.

TRODO, Lia Degrazia. **Uso de métricas nos testes de software**. 2009. 128 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Federal do Rio Grande do Sul, Porto Alegre.

VLAANDEREN et al. **The agile requirements refeinery**: Applying Scrum principles to software product management. 2011. Disponível em: http://www.sciencedirect.com/science/article/pii/S0950584910001539>. Acesso em: 20 outubro 2016.

WEBER, Pedro Anselmo. **Taskboarddev** - ferramenta para monitoramento e rastreabilidade de atividades de manutenção de software baseada em conceitos ágeis. 2014. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Sistema da Informação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

YONGCHANG et al. **Software maintenance process model and contrastive analysis**. 2011. Disponível em: < http://ieeexplore.ieee.org/document/6116869/>. Acesso em: 20 setembro 2016.

ZHANG, H.F. **Introduction to software engineering analysis**. 2003. Disponível em: <http://www.slideshare.net/blpgirl/introduction-to-software-engineering-1508707>. Acesso em: 30 setembro 2016.