Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Desenvolvimento de um módulo de help desk para o sistema Sisgera

Lucas Horta Monteiro de Castro

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:

Prof. Me. Euler Horta Marinho

COORIENTAÇÃO:

Silvandro Sergio Martins Oliveira

Junho, 2022
João Monlevade–MG

Lucas Horta Monteiro de Castro

Desenvolvimento de um módulo de help desk para o sistema Sisgera

Orientador: Prof. Me. Euler Horta Marinho

Coorientador: Silvandro Sergio Martins Oliveira

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Junho de 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

C355d Castro, Lucas Horta Monteiro de.
Desenvolvimento de um módulo de help desk para o sistema Sisgera.. [manuscrito] / Lucas Horta Monteiro de Castro. - 2022.
48 f.

Orientador: Prof. Me. Euler Horta Marinho.
Coorientador: Silvandro Sergio Martins Oliveira.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Aplicações Web. 2. Engenharia de software. 3. Sistemas de informação gerencial. 4. Software de aplicação - desenvolvimento. I. Marinho, Euler Horta. II. Oliveira, Silvandro Sergio Martins. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.775

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Lucas Horta Monteiro de Castro

Desenvolvimento de um módulo de *help desk* para o sistema Sisgera

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 15 de junho de 2022

Membros da banca

Mestre - Euler Horta Marinho - Orientador (Universidade Federal de Ouro Preto)
Silvandro Sergio Martins Oliveira - Coorientador (Bio Extratus Cosméticos Naturais)
Doutor - Diego Zuquim Guimarães Garcia - (Universidade Federal de Ouro Preto)
Mestra - Daniela Rodrigues Dias - (Doutoranda em Educação - Universidade Federal de Ouro Preto)

Euler Horta Marinho, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 31/07/2022

Documento assinado eletronicamente por **Euler Horta Marinho**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 31/07/2022, às 18:13, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0370799** e o código CRC **1AE375A5**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado a minha mãe, por sempre depositar sua confiança em mim.
Espero que faça uma boa leitura junto aos anjos!!

Agradecimentos

Agradeço primeiramente a Deus, por sempre guiar as minhas escolhas, por fornecer coragem e força nos momentos mais difíceis.

A minha mãe, Maria da Penha Horta Timóteo da Silva Castro. De tempos em tempos, leio em um papel A4 o seu rascunho deixado. Te defino como esplêndida, a minha base, a mulher que reverencio. Hoje entendo as suas cobranças e puxões de orelha, são os frutos colhidos do meu sucesso. A escola que aprendi a música "A paz do mundo começa em mim"é a mesma que seguro o meu diploma, a escola da vida. Obrigado por me ter gerado em seu útero.

Ao meu pai José Carlos Monteiro de Castro. Obrigado por ser o meu alicerce, por demostrar tanto amor e compaixão, por esboçar no seu olhar tanto orgulho e o sentimento de dever cumprido. Gostaria de poder retribuir ao meu filho tudo que o senhor me proporciona. Obrigado por nunca ter hesitado em investir em mim, por doar-se por inteiro à minha formação pessoal e profissional.

A minha irmã Bárbara Horta Monteiro de Castro. Você é o reflexo da mamãe, minha melhor amiga, que sempre aconselha e faz com que o ambiente esteja propício para eu alcançar o meu limite, sempre regando a semente do amor, da compaixão e da família, dia após dia. Desconheço um ser humano com o coração tão bom quanto o seu, nunca perca a sua essência, pois é ela que mais me cativa.

Aos meus amigos, sempre presentes, únicos. Fico feliz por compartilhar momentos inesquecíveis com vocês. A minha querida República Beco, lugar onde formei minha segunda família, minha casa de companheiros. Nossa querida Clarice Lispector já dizia, "Quem caminha sozinho pode até chegar mais rápido, mas aquele que vai acompanhado, com certeza vai mais longe".

Aos meus orientadores Euler e Silvandro, por toda calma, paciência e sabedoria depositada. Obrigado pelos ensinamentos dentro e fora de sala. Por auxiliarem no desenvolvimento desse trabalho com maestria.

Deixo aqui a última mensagem do rascunho A4 deixado pela minha amada mãe: "Adoção é... Encontro de almas!- Denise Zepter.

"Quer saber o sentido da vida? Pra frente."

— Emicida.

Resumo

Este trabalho tem como finalidade desenvolver um módulo de help desk para o SisGera, um sistema utilizado por corporações de bombeiros voluntários.

Iniciamente, as corporações de Bombeiros Voluntários adveio de duas instituições que prestam serviços voluntários para vítimas envolvidas em ocorrências de trauma e atendimentos clínicos, além de atendimentos para a prevenção e combate a incêndios nos municípios de Barão de Cocais, Dionísio, São José do Goiabal, Santa Bárbara, Catas Altas, MGC120, LMG820, estradas vicinais da área rural e o trecho da BR-262 e BR-381, até o trevo de Boa Vista.

Atualmente, expandiu-se para mais corporações e encontra-se com a viabilidade de escalonar à demais instituições de bombeiros voluntários. A fim de atender as demandas das corporações de bombeiros voluntários, foi criado o Sistema de Gerenciamento e Registro de Atividades (SisGera). Este é um sistema de Informação *Web* para apoio ao registro de ocorrências atendidas, dinamizando o registro das informações de forma rápida e precisa, evitando problemas com registro de dados inconsistentes e incompletos.

Nesse contexto, deparamos com um grande fluxo de processos de armazenamento e gerenciamento das informações que devem ser mantidas na base de dados da organização, a fim de permitir consultas futuras e melhorias na tomada de decisões. Como em qualquer sistema, vale ressaltar a propensão para erros, enfatizando-se algumas falhas comuns, como *crash*, erro funcional, comando ausente e etc. O módulo desenvolvido permite o rastreamento de tais erros, aprimorando a qualidade do suporte prestado pelos mantenedores do sistema. Dessa forma, é possível identificar, gerenciar e tratar o problema de forma ágil.

Palavras-chaves: SisGera, *Help Desk*, Módulo, Sistemas.

Abstract

This job aimed to develop the help desk module, a system used by firefighters. Initiation, such as assistance institutions for the fire service of Volunteers from two institutions, assistance services for assistance to trauma incidents and assistance to clinical treatment incidents, assistance to trauma incidents and fire prevention of Cocais in the municipalities of Dionísio, in addition, Santa Bárbara, Catas Altas, MGC120, LMG820, side roads in the rural area and the stretch of BR-262 and BR-381, up to the Boa Vista interchange. Currently, expand to more and meet with the possibility of over-scaling military firefighters. In order to meet the demands of volunteer firefighter activities, the Management System and Registration Activities (SisGera) was created. This is a *Web* record occurrence support textit information system for fast record support, streamlining the record of data information dynamically inconsistent and accurate, with data record problems and inconsistent and incomplete data. In view of this, we are faced with a large flow of storage and management of information that are kept in the organization's database, in order to allow future consultations and adjustments in decision making. Like any system, error proneness is worth emphasizing, emphasizing some common flaws such as *crash*, functional error, missing command, etc. The module developed allowed the tracking of such errors, improving the quality of support provided by system maintainers. In this way, it is possible to identify, manage and treat the problem in an agile way.

Key-words: SisGera, *Help Desk*, Module, System.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

APIs	Application Programming Interface
COSI	Colegiado do Curso de Sistemas de Informação
DECSI	Departamento de Computação e Sistemas
DPVAT	Danos Pessoais por Veículos Automotores Terrestres
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
ICEA	Instituto de Ciências Exatas e Aplicadas
JM	João Monlevade
MVC	*Model-View-Controller*
PHP	*PHP Hypertext Preprocessor*
REST	Representational State Transfer
RUP	Rational Unified Process
SAC	Serviço de Atendimento ao Cliente
SI	Sistemas de Informação
SQL	*Structured Query Language*
SisGera	Sistema de Gerenciamento e Registro de Atividades
TCP/IP	Transmission Control Protocol/Internet Protocol
UFOP	Universidade Federal de Ouro Preto
URLs	Uniform Resource Locator
WebML	Web Modeling Language
WWW	World Wide Web

Sumário

1 Introdução

Arantes (2018) desenvolveu o projeto "Um Sistema de Informação para apoio ao registro de ocorrências atendidas por grupos de bombeiros voluntários", conhecido como SisGera. Sua fundamentação é um sistema para controle de ocorrências das corporações de bombeiros voluntários de São Domingos do Prata e Barão de Cocais, cidades localizadas no estado de Minas Gerais. Posterior a Arantes (2018), temos o trabalho de Oliveira (2018), "Sistema de Informação para controle de materiais e doações aos bombeiros voluntários", agregando novas funcionalidades ao SisGera, otimizando os processos internos com foco na gestão de atividades administrativas e auxiliando nas tomadas de decisões das corporações.

É natural que um sistema encontre-se em constante manutenção, a fim de corrigir defeitos, adequar-se a novos requisitos e aumentar a suportabilidade. Diante disso, temos o desenvolvimento de um módulo que consiste em depauperar as dificuldades dos usuários do SisGera. Em duas monografias são apresentadas as etapas envolvidas no desenvolvimento dos módulos iniciais do sistema de forma detalhada e articulada (Arantes (2018), Oliveira (2018)).

O trabalho em questão deseja levantar os requisitos necessários e implementar o módulo de *help desk* para o SisGera, permitindo o rastreamento de tais erros, mediante a identificação do problema pelos usuários, o desenvolvimento da sua solução e a apresentação, como também a correlação com outras situações.

1.1 Problema

Os Bombeiros Voluntários prestam serviços para vítimas envolvidas em ocorrências de trauma e atendimentos clínicos, além de atendimentos para a prevenção e combate a incêndios nos municípios de Barão de Cocais, Dionísio, São José do Goiabal, Santa Bárbara, Catas Altas, MGC120, LMG820, estradas vicinais da área rural e o trecho da BR-262 e BR-381, até o trevo de Boa Vista.

A fim de atender as demandas das corporações de bombeiros voluntários, foi criado o SisGera. Este é um sistema de informação *Web* para apoio ao registro de ocorrências atendidas, dinamizando o registro das informações de forma rápida e precisa, evitando problemas com o registro de dados inconsistentes e incompletos. Inicialmente focado nas atividades dos bombeiros das cidades de São Domingos do Prata e Barão de Cocais, atualmente, encontra-se em potencial expansão para outras instituições de bombeiros voluntários.

Diante do serviço prestado pelo grupo de bombeiros, as informações referentes ao

atendimento precisam ser assertivas e registradas de forma rápida, para dar continuidade à assistência. Não obstante, os dados registrados podem ser inverídicos, gerando inconsistências na emissão de relatórios necessários para a concessão do seguro de Danos Pessoais por Veículos Automotores Terrestres (DPVAT) para o acidentado, como também, aumento no tempo médio de atendimento, descentralização da comunicação, aumento de custos, entre outros.

Frente o exposto, deparamos com um grande fluxo de processos de armazenamento e gerenciamento das informações que devem ser mantidas na base de dados da organização, a fim de permitir consultas futuras e melhorias na tomada de decisões. Como qualquer sistema, vale ressaltar a propensão a erros, enfatizando-se algumas falhas comuns, como *crash*, erro funcional, comando ausente e etc. É de suma importância relacionar a gestão da informação por parte dos colaboradores das organizações, sempre visando possuir registros com informações concisas. Dessa forma, o colaborador pode construir conhecimento sobre as ocorrências criadas.

O estudo em questão é em razão da gama de problemas enfrentados pelos usuários do sistema diariamente. Problemas estes que, na maioria das vezes, podem ser tratados utilizando dados coletados previamente e implementando um protótipo especialista, bem estruturado e de forma agilista. Antemão, é possível monitorar as atividades realizadas pelos usuários dentro do sistema, coletando as falhas comuns, fazendo a sua gerência por um fluxo de chamado definido, automatizado e protocolado. Também é possível realizar alterações no SisGera para o próprio sistema realizar a coleta dos dados e com a implementação necessária, de modo que seja possível transportar as informações para a análise do setor responsável por atuar na demanda.

1.2 Objetivos

Nessa seção, serão descritos o objetivo geral e os objetivos específicos do trabalho. Para tanto, faz-se necessário adentrar aos conceitos da Engenharia de *Software*, utilizando tecnologias *Web*.

1.2.1 Objetivo Geral

Desenvolver um módulo de *help desk* para o SisGera, aprimorando a qualidade do suporte prestado pelos mantenedores do sistema.

1.2.2 Objetivos Específicos

Este trabalho tem como objetivos específicos:

- Realizar um breve estudo dos requisitos da aplicação já desenvolvida.

- Identificar e especificar os novos requisitos da aplicação.
- Desenvolver o módulo de *help desk* para o SisGera, com o apoio de ferramentas que auxiliam no processo de desenvolvimento da aplicação.
- Rastreamento de erros, mediante a integração de um sistema de *bug tracker* existente.

1.3 Organização do Trabalho

No Capítulo 1 são apresentados alguns conceitos introdutórios, os quais serão abordados e discutidos de forma detalhada no decorrer da discussão acerca do projeto.

No Capítulo 2 é apresentada a fundamentação teórica, cujo conteúdo tem como base as tecnologias utilizadas para o desenvolvimento do trabalho, abordando cada uma delas, de maneira a estabelecer a relação que resultou no projeto apresentado.

No Capítulo 3 são apresentados os materiais e a metodologia para o desenvolvimento, assim como o módulo criado dentro do sistema, o descritivo dos seus recursos, regras de negócio, ferramentas e as funcionalidades de maneira mais detalhada. A integração do sistema de *bug tracker* também faz referência no capítulo em questão.

O Capítulo 4 destina-se à apresentação dos resultados obtidos com o desenvolvimento do sistema, desde a descrição do sistema final até a elucidação dos problemas e desafios encontrados durante o processo. De antemão, a conclusão obtida é exposta à nível estratégico e operacional.

2 Revisão bibliográfica

Apresenta-se a seguir a revisão bibliográfica, com base nos autores e trabalhos correlatos. Destacam-se os conceitos já levantadas nos trabalhos anteriores, de Oliveira (2018) e Arantes (2018), como também conceitos de Sistemas de Informação, Engenharia de *Software*, Levantamento de Requisitos, Prototipação, *Web*, Aplicações *Web*, Sistemas para *Help Desk* e Sistemas para Rastreamento de *Bugs*.

2.1 Sistemas de Informação

Diante da necessidade de racionalizar a administração da tecnologia no seio das organizações, fomentou-se o conceito de Sistemas de Informação (SI).

Para Laudon e Laudon (2010):

> "Um SI pode ser definido tecnicamente como um conjunto de componentes inter-relacionados que coletam (ou recuperam), processam, armazenam e distribuem informações destinadas a apoiar a tomada de decisões, a coordenação e o controle de uma organização. Além de dar apoio à tomada de decisões, à coordenação e ao controle, esses sistemas também auxiliam os gerentes e trabalhadores a analisar problemas, visualizar assuntos complexos e criar novos produtos".

Cruz (2000) define Sistema como:

> "a disposição das partes de um todo, dentro de uma estrutura organizada, com a finalidade de executar tarefas" e Informação como "o resultado do tratamento dos dados existentes acerca de alguém ou de alguma coisa. A Informação aumenta a consistência e o conteúdo cognoscível dos dados.".

Antecipadamente, é necessário gerenciar a informação, sobretudo para a sua construção e manutenibilidade no tempo. Assim, o desenvolvimento de meios e métodos é crucial para a aplicabilidade dentro de um sistema de informação. A modularidade é um exemplo de subdivisão realizada em sistemas de informação, caracterizando, assim, o grau de alterações de desempenho e funcionalidade que o sistema pode sofrer, satisfazendo um módulo bem definido dentro da arquitetura tecnológica no âmbito geral.

2.2 Engenharia de Software

Segundo Sommerville e Prechelt (2004):

> "O mundo moderno não poderia existir sem o *software*. Infraestruturas e serviços nacionais são controlados por sistemas computacionais, e a

> maioria dos produtos elétricos inclui um computador e um *software* que o controla. A manufatura e a distribuição industriais são totalmente informatizadas, assim como o sistema financeiro. A área de entretenimento, incluindo a indústria da música, jogos de computador, cinema e televisão, faz uso intensivo de *software*. Portanto, a Engenharia de *Software* é essencial para o funcionamento de sociedades nacionais e internacionais."

Mediante isso, a Engenharia de *Software* aplica-se aos processos técnicos do desenvolvimento de *software*, como também ao gerenciamento de projetos de *software*, métodos, teorias para fundamentar a produção e o desenvolvimento de ferramentas, a fim de obter resultados de alta qualidade. Podemos destacar a abordagem sistemática presente na Engenharia de *Software*, principalmente no que diz respeito a produção, metrificando questões como prazo, confiabilidade, custo e as necessidades dos *stakeholders*.

2.2.1 Manutenção de *Software*

Um dos principais objetivos em realizar a manutenção de *software* é o aperfeiçoamento tecnológico do sistema. Ao realizar essa atividade, o sistema se tornará cada vez mais estável, diminuindo sua velocidade de envelhecimento.

Os negócios estão sempre em mudança e o software deve acompanhar esse movimento. Dessa forma, é fundamental que o produto se adapte a novas regras de negócios, e acumule novas funcionalidades de acordo com as necessidades das partes envolvidas.

Um *software* deve ser escrito de modo que possa evoluir para atender às necessidades de mudança dos clientes. A mudança de *software* é uma consequência inevitável de um ambiente de negócios em constante mutação. A manutenção consiste em corrigir erros e acrescentar novas funções à medida que novos requisitos são identificados.

Existem três tipos de manutenção de *software*, sendo adaptativas, corretivas e evolutivas.

As adaptativas visam enquadrar o *software* a uma nova regra de negócio. Ou seja, tem a finalidade de adequar o sistema ao ambiente no qual está inserido. A corretiva tem como objetivo solucionar defeitos encontrados no *software*. Os problemas de funcionalidade são comuns e em alguns casos devem ser corrigidos de forma emergencial. Enquanto a evolutiva visa agregar novas funcionalidades e melhorias para o *software*.

Fundamentando-se nessas definições, podemos classificar o desenvolvimento do módulo de *help desk* como uma manutenção evolutiva.

2.2.2 Levantamento de Requisitos

O levantamento de requisitos diz respeito à fase inicial do processo de engenharia de requisitos e engloba os trabalhos de conhecimento e elicitação dos requisitos.

Nessa fase, especialistas de domínio, clientes e futuros usuários, interagem visando compreender a organização a nível de processos, necessidades, fraquezas dos *softwares* atuais, possíveis melhorias e restrições. Trata-se de um trabalho complexo no qual não se limita ao desejo do cliente, mas sim estudar de forma acurada a organização bem como o domínio da aplicação Kotonya e Sommerville (1998).

Para tal análise, é necessário expor técnicas que remetem a busca pelo *feedback* por parte dos usuários em relação as suas necessidades, visando a captura de informações no contexto do negócio, as necessidades e restrições dos envolvidos, o problema a ser solucionado e o entendimento global da área na qual o sistema irá ser implementado. Pode-se citar algumas técnicas, como entrevistas, cenários, histórias de usuários, prototipagem, casos de uso, etnografia, validação, *workshops* e questionários.

Dentre o leque de técnicas de elicitação, o estudo em questão aplica-se a prototipagem e histórias de usuários.

2.2.3 Prototipação

Segundo Sommerville (2011), "protótipo é uma versão inicial de um sistema usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções".

Decorrente do levantamento de requisitos, juntamente com as histórias de usuários, a prototipação visa facilitar o entendimento de ambos, apresentando de forma simplificada conceitos e funcionalidades do *software*. Além de agregar valor ao produto, vale ressaltar a organização do *layout* proposto, o impacto na experiência do usuário e o melhor alinhamento entre os envolvidos sobre o que vai ser desenvolvido, distanciando-se de possível retrabalho. Ajustes podem ser realizados com frequência, porém, quanto mais fiel o protótipo for, mais assertivo no que diz respeito ao resultado final.

Destaca-se o uso dos protótipos para antecipar as mudanças que podem ser requisitadas, dar suporte a interação do usuário com a interface, evidenciar erros e omitir requisitos propostos, como também visar a viabilidade do projeto proposto. O desenvolvimento do protótipo deve gerar baixo esforço e custo, de forma a ser apresentando logo no início do projeto.

2.3 *Web*

É curioso analisar a velocidade vertiginosa em que a *Web* tem-se desenvolvido, como meio de divulgação e localização de informações variáveis. É comumente aceito as denominações *Web* 1.0, *Web* 2.0 e *Web* 3.0. Porém, é necessário maiores considerações históricas para situar suas denominações no respectivo contexto espaço-temporal.

Com os avanços tecnológicos propostos e impulsionados pelo setor militar norte-americano na década de 1980, temos o advento do termo rede mundial de comunicação. Nos anos finais da década, a abertura da rede baseada no protocolo TCP/IP torna-se de conhecimento comercial, explorando serviços de interconexão entre os correios eletrônicos e provedores, principalmente. Com a expansão da Internet, o amadurecimento do termo faz-se presente, representando o maior conglomerado de redes de comunicações, visando a conexão de computadores à uma escala mundial.

A conectividade abstém da exclusividade, expande em todos os âmbitos da sociedade. Dessa forma, temos o entendimento de que *Web* 1.0, caracterizada por suas páginas estáticas, comportando textos, imagens e links, somente o administrador poderia alterar.

A *Web* 2.0 é refletida na mudança progressiva e irreversível que torna as páginas dinâmicas, abrindo aos usuários a oportunidade de alterar e acrescentar os dados.

Para O'Reilly (2006), "A *Web* 2.0 é a revolução dos negócios e da indústria dos computadores causada pela conversão da Internet em uma plataforma, e pela tentativa de entender as regras do sucesso da nova plataforma."

Já em relação ao uso da termologia *Web* 3.0, pode-se descrever o estado de evolução posterior a *Web* 2.0, sendo uma introdução de uma suposta 'onda' de inovações da Internet, abrangendo conceitos tecnológicos emergentes, que por sua vez, aponta para novos paradigmas na representação e organização do conhecimento e seus desdobramentos práticos, onde a lógica é agregada à *Web*. Podemos também referenciar a *Web* 3.0 como *Web* Semântica.

Com as mudanças tecnológicas presentes, temos a alteração na interação humana com o mundo à volta, objetivando atender as novas demandas dos consumidores e produtos. Essas manifestações se entrelaçam ao conceito *Internet* das Coisas (*Internet Of Things*, IoT), sendo possível que um conjunto de tecnologias associadas permitam que objetos se conectem a uma rede de comunicações, possibilitando que sejam identificados e controlados adentro da conexão de rede.

2.3.1 Aplicações *Web*

Filho (2003) define aplicações *Web* como:

> "produtos de software ou sistemas de informática que utilizam uma arquitetura distribuída, pelo menos parcialmente sob o protocolo HTTP. Em consequência, pelo menos parte das interfaces com o usuário é acessível através de um navegador (*browser*)".

As aplicações *Web* estão presentes no cotidiano, representando tipicamente a sua produção em um ambiente de trabalho multidisciplinar, composto por profissionais que

possuem as mais distintas formações, englobando área gráfica, de comunicação, análise e desenvolvimento de sistemas, *design*, entre outras. Contudo, é fundamental entender como se dá a produção de aplicações *Web* com elevado grau de interatividade com o usuário e o desenvolvimento de funcionalidades complexas.

Ceri, Fraternali e Bongio (2000) define quatro perspectivas ortogonais para a representação de uma aplicação *Web*:

> 1) Modelo estrutural: modela o conteúdo de dados do site em termos de entidades e relacionamentos; 2) Modelo de Hipertexto, descreve a estrutura de hipertexto do site e consiste em dois submodelos: 2a) Modelo de composição: especifica quais páginas compõem a estrutura de hipertexto e quais unidades de conteúdo compõem uma página; 2b) Modelo de navegação: expressa como as páginas e unidades de conteúdo estão vinculadas entre si para compor a estrutura de hipertexto; 3) Modelo de apresentação: expressa o *layout* e aparência das páginas; 4) Modelo de personalização: permite modelar personalizações da aplicação criadas para algum usuário ou grupos de usuários.

Ward e Kroll (1999) sugerem uma proposta para o desenvolvimento de aplicações *Web* buscando unificar o processo criativo de *design* com o processo de Engenharia de *Software* proposto pelo *Rational Unified Process* (RUP).

Identificando os elementos propostos, levanta-se, para o desenvolvimento do SisGera, questões válidas e especificações dos requisitos desenvolvidos e alinhados pela Engenharia de *Software*, direcionado para o modelo de Casos de Uso. É de suma importância definir os requisitos não-funcionais, assim como realizar o *briefing* de *design*, a criação do diagrama de navegação inicial, a composição gráfica (*layout*), a criação dos elementos de *Web design*, como menus, fundos e elementos gráficos. O protótipo inicial de interfaces também se destaca, principalmente por focar em aspectos funcionais, como formulários, janelas, mensagens e links.

2.3.2 APIs REST

Historicamente, as Interfaces de Programação de Aplicações, conhecidas como APIs, existem desde o advento dos computadores pessoais. Podemos caracterizá-las como um conjunto de regras que definem como aplicativos e dispositivos se conectam e comunicam uns com os outros.

Os aplicativos ou serviços que estão realizando o acesso são conhecidos como clientes, mediante os aplicativos ou serviços que contém o recurso, sendo eles denominados servidores.

A chamada de procedimentos remotos e os serviços da *Web* são soluções amplamente aceitas para fornecer uma única interface de programação para vários idiomas. É de interesse literário entender melhor sobre *Representational State Transfer* (REST), padrões baseados

na reutilização da tecnologia HTTP para receber e enviar dados de uma página *Web* por meio das URLs (*Uniform Resource Locator*). Uma API REST modela os princípios de um projeto REST, diante disso, temos o uso do termo APIs RESTful.

Algumas APIs impõem uma estrutura rígida a ser seguida à nível de desenvolvimento, porém, as APIs REST possuem a flexibilidade necessária, oferecendo suporte a uma variedade de formatos de dados e diversas linguagens de programação.

Deve-se alinhar as restrições de arquitetura das APIs REST para cumprir com os princípios. Pode-se discorrer seis restrições, como interface uniforme, desacoplamento do cliente-servidor, sem estado definido, capacidade de armazenamento em cache, arquitetura de sistema em camadas e código sob demanda.

2.3.3 Métodos

As APIs REST se comunicam via solicitações de HTTP (*HyperText Transfer Protocol*). Dessa forma, é possível executar funções como criar, ler, atualizar e excluir registros em um recurso. É possível recuperar um registro realizando solicitação GET, método que permite que os dados sejam pesquisados de acordo com o recurso, retornando as informações consultadas.

Uma solicitação POST é usada para criar um registro. Dessa forma, é possível executar uma carga de novas informações para a base. Uma solicitação PUT é usada para atualizar ou editar um registro existente. Já o método DELETE é usado para excluir um registro.

Todos os métodos HTTP podem ser utilizados em chamadas da API. Contudo, uma API REST bem projetada é semelhante a um *website* em execução em um navegador *Web* com funcionalidades HTTP integrada. Os cabeçalhos e parâmetros de solicitação ganham destaque em chamadas de API REST, devido ao fato de incluírem informações como metadados, autorizações, *cookies*, armazenamento em cache, URLs e mais.

2.4 Sistemas para *Help Desk*

> Um sistema *help desk* possui a função de contribuir para aprimorar o trabalho da equipe de suporte das empresas, auxiliando na coordenação e solução de dificuldades trazidas pelos usuários, com o objetivo de atender e resolver estes incidentes com eficácia e eficiência. Ainda exerce papel de sistema especialista, empregando os registros cadastrados como base para apoio às decisões em todos os níveis de suporte, facilitando os atendimentos e fazendo-os com qualidade MELO A. L.; MENDES (2015).

Cavalari e Costa (2005) definem:

> "Um sistema *help desk* é utilizado para melhorar o gerenciamento das soluções de atendimento. Por meio de um sistema *help desk*, cria-se uma ampla base de dados para a empresa, a qual permite gerenciar os problemas, resolvê-los na sua raiz e diminuir custos operacionais. O *help desk* pode centralizar uma diversidade de informações e áreas de atendimento, tornando-se assim um ponto chave na administração e na solução de problemas. Um sistema *help desk* pode ser aplicado a domínios relativos à: suporte à informática; SAC – Serviço de Atendimento ao Cliente (interno/externo); controle de serviços/manutenção; e centro de informações".

Dessa forma, é possível informatizar o processo de abertura de chamados, facilitando o intercepto das informações, como realizar o efetivo acompanhamento dos reportes de problemas, controlando os processos de forma clara. Pode-se mensurar o nível de satisfação dos clientes com relação ao serviço prestado, como também emitir relatórios gerenciais para acompanhamento e registro.

A metodologia de atendimento e suporte usada, a fim de atender as partes envolvidas no SisGera, em especial os usuários, funcionava mediante o contato via telefone para o responsável pelo desenvolvimento do sistema. Era registrada a solicitação e o profissional responsável realizava o tratamento do problema no tempo sugerido, retornando para o remetente o *status* positivo ou negativo.

Diante da necessidade de melhorar o serviço de atendimento e auxiliar a usabilidade dos recursos do SisGera, tem-se a implementação de um módulo de *help desk*, visando aprimorar a qualidade do sistema, observar os processos descritos, gerar, organizar, armazenar e transmitir informações de diagnóstico de maneira eficiente e sólida.

Como enfoque, é constatado que os objetivos e as necessidades de negócios de uma organização estão atrelados aos serviços disponibilizados pela área de TI. Diante dessa cenário, é importante fornecer o suporte ideal para que tais serviços estejam em sintonia com as necessidades do negócio. Ademais, é fundamental ter conhecimento sobre o gerenciamento de serviços de TI com base na *Information Technology Infrastructure Library* (ITIL), que pode ser definido como um conjunto de normas e procedimentos vigentes, recomendações técnicas que indicam o grau de maturidade de uma organização.

Nota-se que com a implementação do *help desk* é possível unificar o canal de comunicação do sistema, como facilitar as solicitações de suporte dos usuários, sistematizar os processos trabalhados, documentar as soluções expostas em cenários diferentes e reduzir recursos, como custos financeiros e operacionais, priorizando a qualidade do atendimento. Entrelaçando para a análise da ITIL, é visível a melhoria na satisfação dos clientes dependentes de um ou mais serviços, assim como a eficiência operacional, redução nos custos e nos esforços depreendidos, entre outros.

2.5 Sistema para Rastreamento de *Bugs*

O controle de qualidade é essencial para o desenvolvimento e manutenibilidade de aplicações robustas.

Segundo Humphrey (1995), estima-se que os desenvolvedores de *software* cometem de 100 a 150 erros para cada mil linhas de código. O Krasner (2018) enfatiza que, mesmo com uma pequena fração desses erros sendo de caráter grave, totalizando 10%, então uma aplicação relativamente pequena, com 20.000 linhas de código, terá, aproximadamente, 200 erros de codificação sérios.

"O rastreamento de defeitos é um processo importante na engenharia de *software*, pois sistemas complexos e críticos de negócios têm centenas de defeitos", diz Fichman e Kemerer (1993). Diante disso, existe a necessidade de processos de registro e monitoramento de *bugs* ou erros durante a execução do *software*. O gerenciamento se dá pela análise, monitoramento e priorização para depuração.

Um *bug* surge quando a saída esperada para a aplicação não ocorre, sendo falhas ou erros cometidos pelas partes envolvidas. O gerenciamento de defeitos se dá pela prioridade e gravidade.

Sabendo os impactos relativos a serem tratados por certo problema, é possível monitorar e avaliar melhor o *status* do erro e o potencial impacto encontrado, garantindo que os *bugs* sejam tratados. Ferramentas direcionadas a gerenciamento de mudanças e rastreamento de *bugs* permitem que as equipes descubram defeitos, meçam esforços para tratar e analisem os impactos presentes, a fim de resolverem.

Entender os requisitos de desenvolvimento de *software* é fundamental para escolher a melhor ferramenta de rastreamento de *bugs*. Dessa forma, é possível priorizar os recursos presentes na aplicação, a fim de encontrar compatibilidade no ciclo de vida do sistema.

Por sua vez, o SisGera destaca-se no desenvolvimento da estrutura de banco de dados otimizada, atrelando desempenho e escalabilidade às informações ali presentes. É notória a necessidade da interseção entre os recursos disponíveis entre os sistemas, com isso, é essencial realizar um comparativo das ferramentas presentes no mercado e suas características, principalmente no que tange aos recursos elicitados no processo do chamado.

Com o processo definido, é indispensável um sistema de *tickets* para identificar e facilitar a participação do ator no gerenciamento dos chamados. Posto isso, é base gerenciar através de identificadores únicos os atendimentos abertos, a fim de evitar ruído na comunicação do ator com o sistema.

Em razão do SisGera trabalhar com políticas rígidas de proteção de dados, principalmente por serem condizentes à Lei Geral de Proteção de Dados, é fundamental interligar

os recursos de controle de acesso e históricos de mudanças, com a intenção de estruturar as ações prestadas pelos usuários logados, além de obter históricos de mudanças.

Vale ressaltar, que todas as ferramentas informadas abaixo são aplicações *Web*, na qual possuem alto nível de integração com *plugins*, possuem uma área para gerenciar os *tickets* abertos, possibilitam o envio de notificações via *e-mail* para os usuários e possuem suporte para diversos sistemas de banco de dados, como MySQL, MS SQL, PostgreSQL, Oracle, entre outros.

Diante disso, a análise ressalta os recursos voltados para a experiência de navegação por parte do usuário do sistema, diminuindo a curva de aprendizado do operador, como também o acompanhamento do tempo de resposta do chamado, o alto nível de personalização do chamado, a capacidade de gerenciar o controle de acesso em diferentes níveis e a possibilidades de realizar o acompanhamento, solicitação ou atualização do chamado através de um dispositivo móvel.

Segue a tabela 1, mostrando o comparativo de alguns sistemas de rastreamento de *bug* e os recursos discutidos.

Tabela 1 – Comparativo Sistemas de Rastreamento de *Bug*.

Recursos	**Mantis**	**Bugzilla**	**Redmine**
Experiência de navegação simples para usuários	Sim	Não	Não
Acompanhamento do tempo de resposta dos chamados	Sim	Não	Não
Alto nível de personalização	Sim	Não	Sim
Gerenciar controle de acesso	Sim	Sim	Não
Versão móvel multifuncional	Sim	Não	Não

Fonte: Adaptado pelo Autor.

Diante do exposto, observa-se uma qualificação da ferramenta Mantis, motivo pela qual recebe ênfase no próximo capítulo. A escolha do sistema de rastreamento de *bugs* tem um impacto direto na eficiência do processo de desenvolvimento, como também nutre as questões relacionadas a qualidade. O Mantis também sobressai por não existir limite no número de usuários, problemas ou projetos; seu sistema de filtragem é considerado poderoso e pensativo, como também é caracterizado por ter revisão de *bugs* facilmente compreensível.

2.6 Considerações Finais

Neste capítulo foram apresentados conceitos sólidos relacionados ao desenvolvimento deste trabalho, realizando uma introdução sobre a *Web* e como as aplicações *Web* impulsionaram a conectividade entre os dispositivos.

Com o intuito de desenvolver um módulo de *help desk*, é de suma importância enfatizar os principais conceitos e a metodologia usada para atendimento e suporte, unificando os canais de comunicação e elicitando o processo, que antes não era definido.

Além de identificar erros no SisGera, é necessário gerenciá-los de forma sucinta e escalonada, dessa forma, tem-se o entendimento sobre as ferramentas para rastreamento de *bugs* e o comparativo dos principais recursos presentes, a fim de gerar insumos para o próximo capítulo.

O próximo capítulo descreve o desenvolvimento do módulo de *help desk* proposto desde as etapas iniciais, incluindo o levantamento de requisitos, as histórias de usuário, a prototipagem, o modelo de banco de dados, a integração do sistema de rastreamento de *bugs* e a validação do sistema.

3 Desenvolvimento e Resultados

Nessa seção, serão apresentados aspectos do desenvolvimento do módulo de *help desk* do SisGera, como o levantamento de requisitos, elaboração das histórias do usuário, protótipos, modelagem do banco de dados e informações específicas da aplicação *Web*.

3.1 Levantamento de Requisitos

Para o levantamento de requisitos, foi necessário realizar o contato com os *stakeholders* do sistema SisGera. Após o entendimento consolidado do artefato de *software*, conduziram-se questões relacionadas a infraestrutura da aplicação, como as tecnologias a serem utilizadas.

Os requisitos de *software* podem ser divididos em requisitos de usuário e de sistema sendo as funcionalidades do *software* descritas em um nível mais alto de abstração, como também descritos de forma minuciosa. Segue abaixo as operações disponíveis para cada tipo de usuário e funcionalidades gerais:

- Os chamados possuem fluxo de progresso, referente ao fato de que podem transitar entre o usuário, o sistema e o analista de *software*. Essa transição deve ocorrer de forma amena, sem perda de informação e sempre com rótulos bem definidos;
- É importante que o usuário seja informado sobre o *status* do chamado. Porém, com descrições breves para melhor entendimento e retorno;
- A associação de *status* no chamado facilita a análise por parte dos envolvidos, sendo uma referência breve aos tipos de usuários e para controle do sistema, para melhor automatização por parte dele.

Tratando o chamado com *status*, temos a padronização das possíveis definições. Elas são cruciais para associar de forma sucinta a sua vigência. Deste modo, tem-se a segmentação do *status* em seis etapas do processo e cada uma possui uma breve descrição, como se observa na Tabela 2.

Tabela 2 – Status da ocorrência e sua descrição.

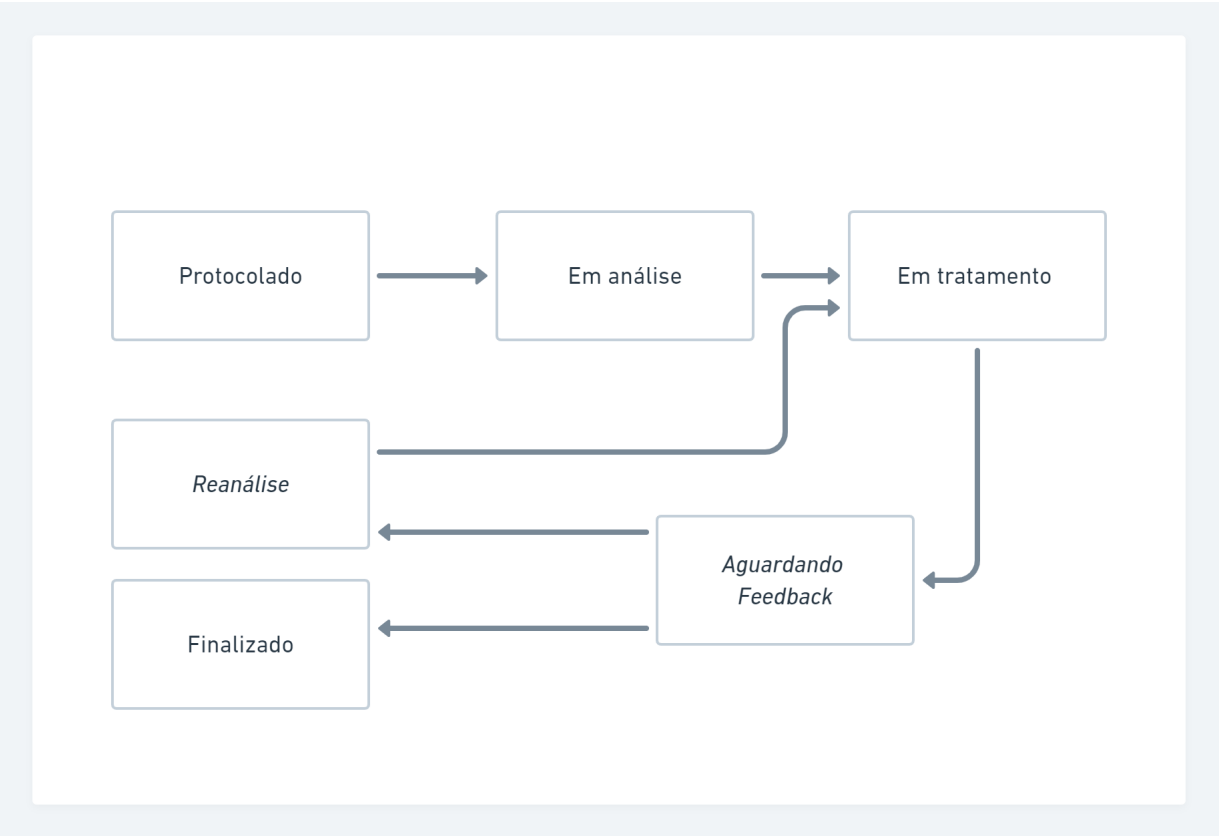
Status	**Descrição**
Protocolado	O chamado é protocolado diante da sua primeira inserção por parte do usuário do sistema. O usuário que cadastrou tem o retorno com o número do protocolo.
Em análise	O chamado encontra-se em análise mediante o aguardo do analista de *software* realizar o resgate.
Em tratamento	Esse nível que o chamado encontra-se é referente ao analista de *software* realizar as devidas verificações e correções relacionadas ao chamado.
Aguardando *feedback*	Quando o chamado necessita de algum retorno por parte do requisitante, o analista informa no chamado o que deseja. Diante disso, seu status é alterado.
Finalizado	O chamado é finalizado, cumprindo a necessidade solicitada por parte do requisitante. Dessa forma, seus dados não podem mais sofrer edições, apenas ser finalizado.
Reanálise	Muitos chamados podem ser incoerentes, mediante a linguagem abordada pelo usuário ou pela falta de informações para dar continuidade. Como também, o analista pode ter uma interpretação errônea perante o chamado. Com isso, é necessário realizar a troca do seu status para reanálise.

Fonte: Elaborado pelo autor (2022).

Podemos, ainda, intitular o módulo na categoria de gerenciamento de soluções de atendimento, sendo possível realizar a interação de uma ferramenta de rastreamento de *bugs* e gerência de chamados, fornecendo informações com base para apoio às decisões em todos os níveis. Contudo, enfatiza-se a necessidade de reestruturar o processo de abertura de chamados por parte dos envolvidos, detendo um fluxo de processo bem definido e interativo.

Para tal, tem-se o seguinte diagrama de fluxo, onde as etapas do chamado são sólidas e estruturadas, sendo elas: protocolado, em análise, em tratamento, aguardando *feedback*, e por fim, finalizado ou reanálise.

Figura 1 – Fluxo proposto do chamado.



Fonte: Elaborado pelo autor (2022)

3.2 História de Usuário

Durante décadas, a utilização de casos de uso, introduzidas por Ivar Jacobson em 1986, foi aplicada para descrever os requisitos funcionais, tendo como principais características a simplicidade e utilidade. Nos anos 2000, a introdução dos processos com base na metodologia ágil ganhou aceitação por parte dos seus adeptos, realizando mudanças significativas nas fases do ciclo de desenvolvimento de *software*, fomentando entregáveis periódicos e a comunicação bilateral frequente entre as partes envolvidas.

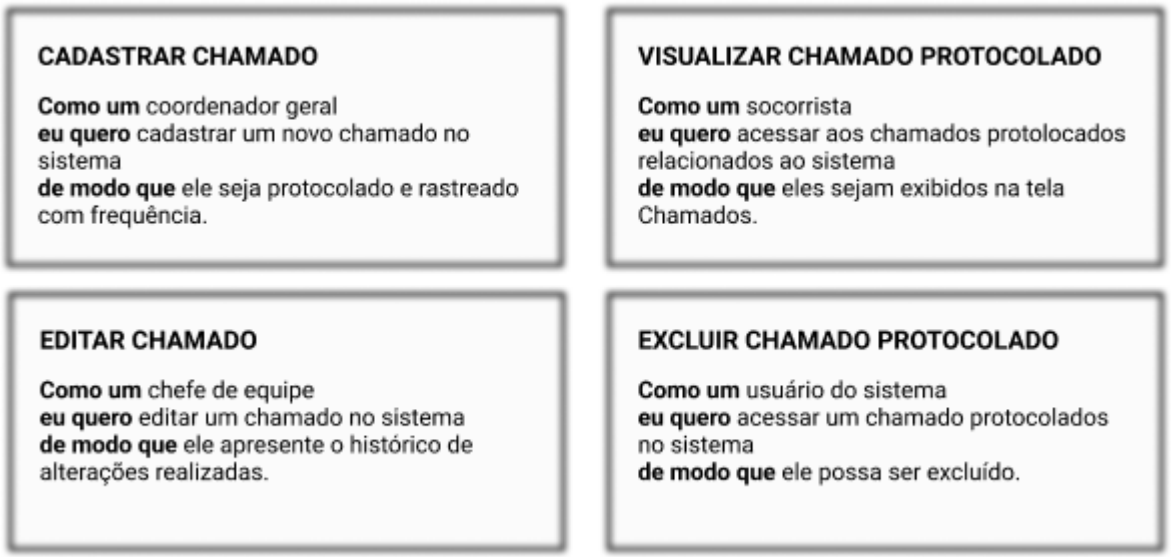
As histórias de usuários expressam, de forma curta, as funcionalidades a partir da perspectiva de um usuário. Podemos basear os valores na simplicidade, *feedback*, comunicação e coragem. Cohn (2004) apresenta de forma clara, o formato de história de usuário para auxílio na comunicação.

A sintaxe de uma história de usuário sugerida por Cohn (2004), assim como a padronizada pelo SisGera, deve conter a seguinte estrutura:

- **Como um**... (papel ou ator) (Quem)
- **Eu quero**... (capacidade ou funcionalidades necessárias)(O que)
- **De modo que**...(porque é de valor do négocio ou benefício)(Por que)

Ciente disso, o SisGera descreve algumas funções internas que fazem referência direta aos usuários finais do processo, destacando-se o coordenador geral, o socorrista e o chefe de equipe. Quando uma funcionalidade não cabe apenas a um tipo específico de usuário, usa-se o termo usuário do sistema, para enfatizar que a história cabe aos demais papéis. A figura 2 enfatiza os exemplos de histórias de usuário do sistema SisGera:

Figura 2 – Relação de histórias com diferentes usuários.



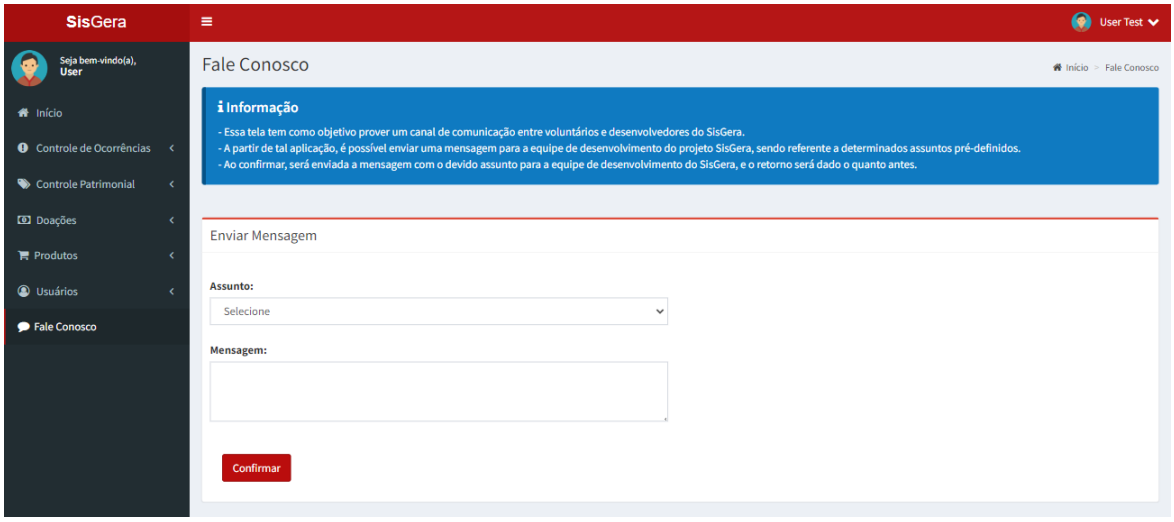
Fonte: Elaborado pelo autor (2022)

3.3 Protótipos

Em face do exposto, foi realizada a criação de protótipos de alto nível sobre a tela Fale Conosco, já existente no SisGera. É de suma importância realizar uma análise do *layout* antes e depois das alterações, a fim de entender melhorias na gama de informações contempladas no protótipo, aumentando a consistência e o conteúdo cognoscível dos dados.

A Figura 3 é um protótipo fiel da interface Fale Conosco do SisGera antes do desenvolvimento do módulo, como pode observar logo abaixo.

Figura 3 – Fale Conosco antes do desenvolvimento do módulo de *help desk*.



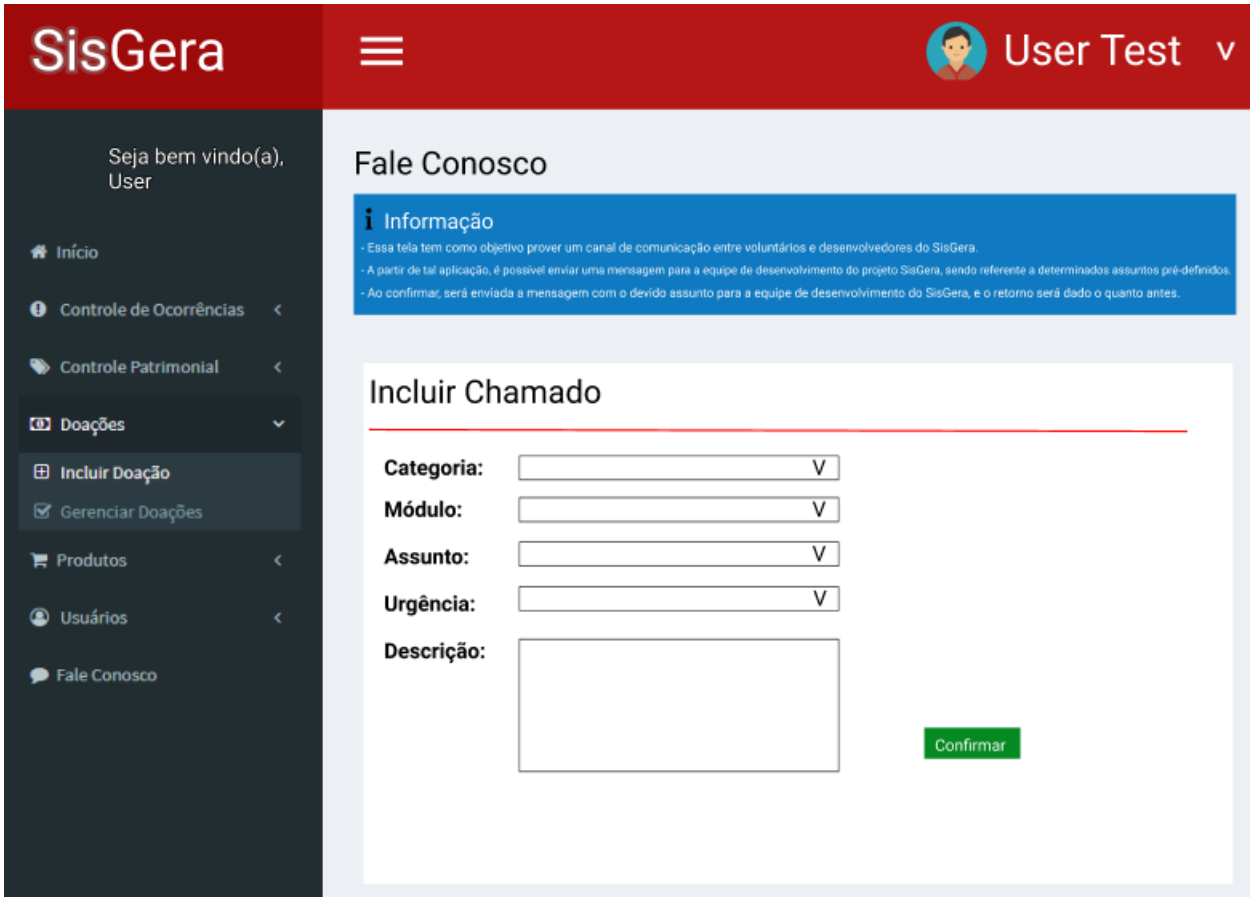
Fonte: Elaborado pelo autor (2022)

A prototipagem redefine a tela Fale Conosco, fazendo com que o usuário final tenha a necessidade de preencher novos campos de dados projetados, como a categoria, o módulo, o assunto, a urgência, e a descrição do chamado proposto. A categoria e o módulo fazem referência direta ao menu principal e ao submenu presente do lado esquerdo da tela, respectivamente.

Para melhor assertividade no processo de criar o chamado por parte do usuário, é necessária a consistência no ato de decisão, ou seja, a busca sistemática dos dados ali presentes. Isso se resume na automação da seleção das opções já presentes no menu e submenu, facilitando a tomada de decisão pelo ator e proporcionando clareza ao destinatário do chamado.

Dessa forma, temos a representação de novos campos na Figura 4, campos pelo qual deseja-se que o usuário selecione adentro de cada tópico o que deseja relatar, sendo possível discorrer na descrição, mais informações sobre o chamado.

Figura 4 – Protótipo do Fale Conosco para desenvolvimento do módulo de *help desk*.

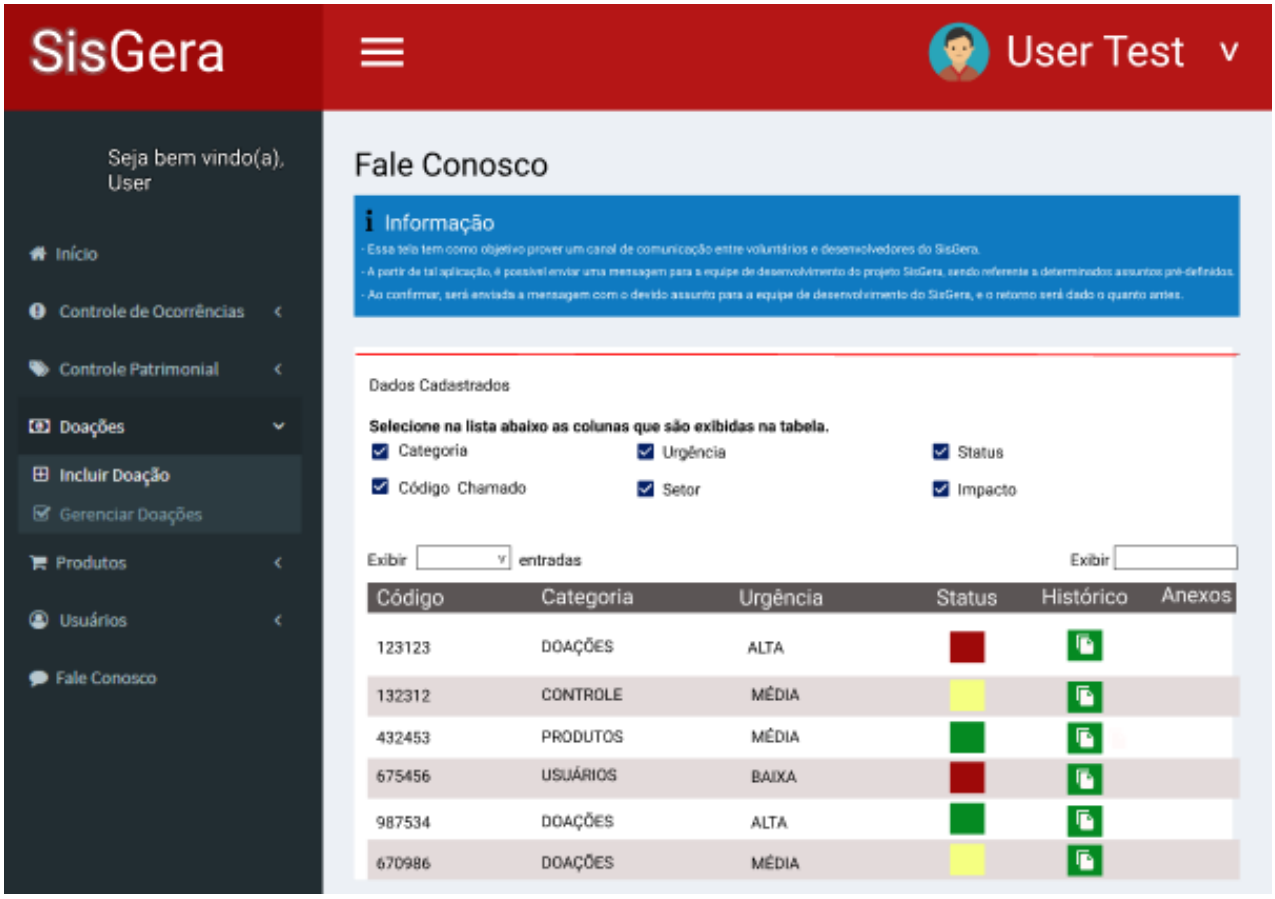


Fonte: Elaborado pelo autor (2022)

Quando qualquer usuário registra um novo chamado, ele é redirecionado para a lista de chamados, que também pode ser acessada a partir do menu lateral do sistema. Nessa lista é possível visualizar todos os chamados gerados com a sua devida descrição, para a validação, atribuição de *feedback*, reanálise, apresentação do histórico de alterações realizadas e seu *status* atual.

A prototipagem simplifica a visualização das histórias de usuários propostas, fazendo com que seja possível metrificar a qualidade dos requisitos levantados e refinar a proposta previamente ao desenvolvimento. Como podemos observar na Figura 5, o protótipo referente à listagem dos chamados é a abstração dos atributos e funções presentes no chamado, sendo possível nortear o processo e simplificar o possível entregável de *software*.

Figura 5 – Chamados listados no módulo de *help desk*.



Fonte: Elaborado pelo autor (2022)

3.4 Modelo de banco de dados

Como necessidade voltada ao armazenamento de dados do SisGera e com o intuito de promover a modelagem dos dados para atender os requisitos organizacionais e técnicos, foi utilizado o banco de dados relacional MySQL. Por sua vez, o MySQL organiza os dados em tabelas, conhecidas no modelo relacional como entidades. As entidades possuem características específicas, que recebem o nome de atributos.

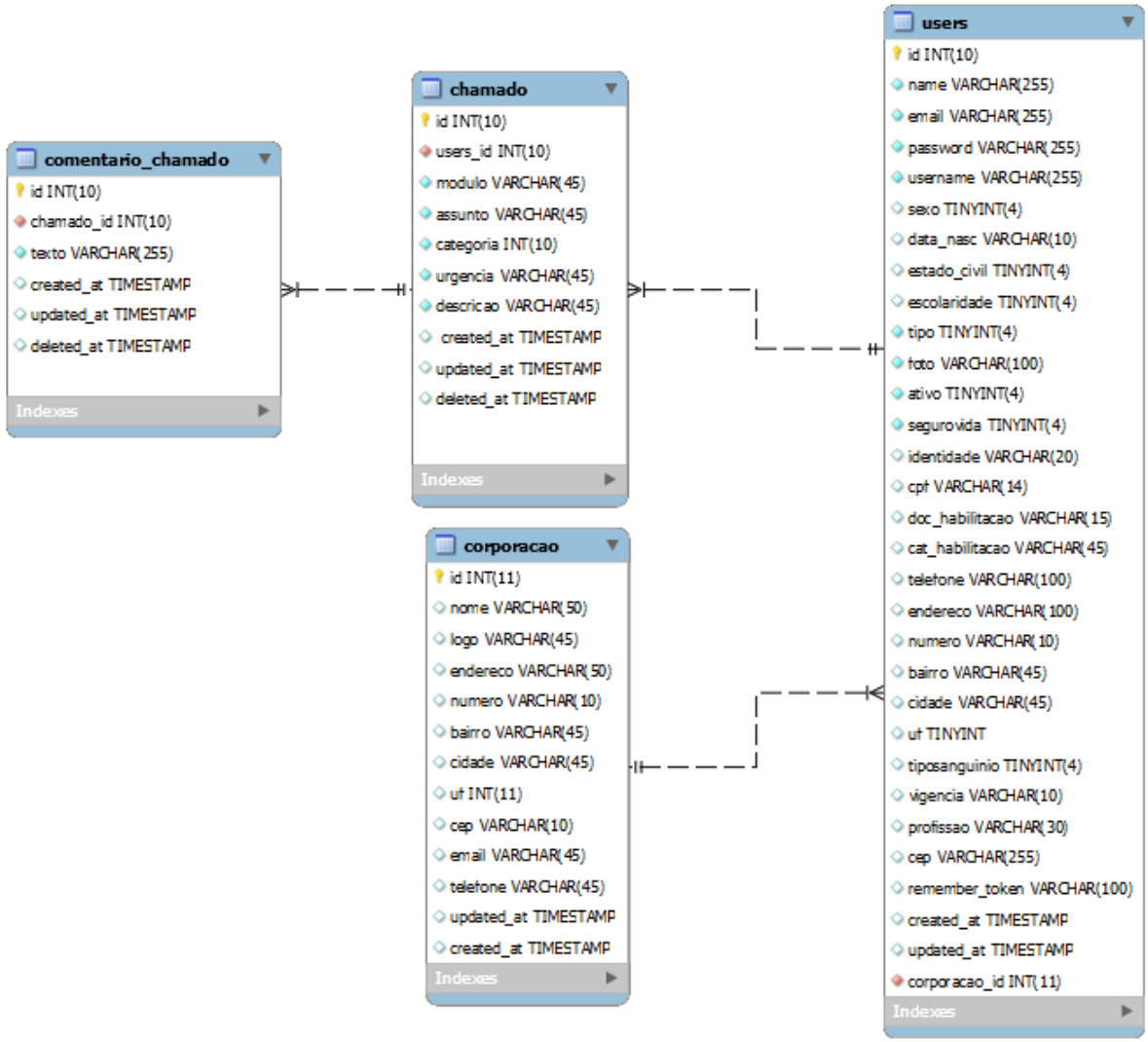
Antemão, a Engenharia de *Software* utiliza o modelo entidade relacionamento para descrever os objetos (entidades) envolvidos no domínio de negócios, e como eles se relacionam entre si (relacionamento).

Diante disso, temos a representação de forma abstrata da estrutura que possuirá o banco de dados da aplicação. A figura 6 mostra as quatro entidades envolvidas no processo, sendo elas o usuário, o chamado, a corporação e o comentário do chamado. Cada entidade possui atributos distintos, que fazem referência às características que deseja-se modelar.

As entidades usuário e corporação foram desenvolvidas nos trabalhos passados,

sendo criadas nesse trabalho as entidades chamado e comentário do chamado.

Figura 6 – Modelo Entidade Relacionamento do Chamado no Sisgera.



Fonte: Elaborado pelo autor (2022)

3.5 Desenvolvimento do módulo

Com os requisitos catalogados e com o modelo entidade relacionamento desenhado, o próximo passo é realizar a configuração do ambiente pelo qual o desenvolvimento do módulo de *help desk* vai ser fundamentado. Como descrito nas monografias de Arantes (2018) e Oliveira (2018), o SisGera foi desenvolvido utilizando o *framework Laravel* e o banco de dados MySQL.

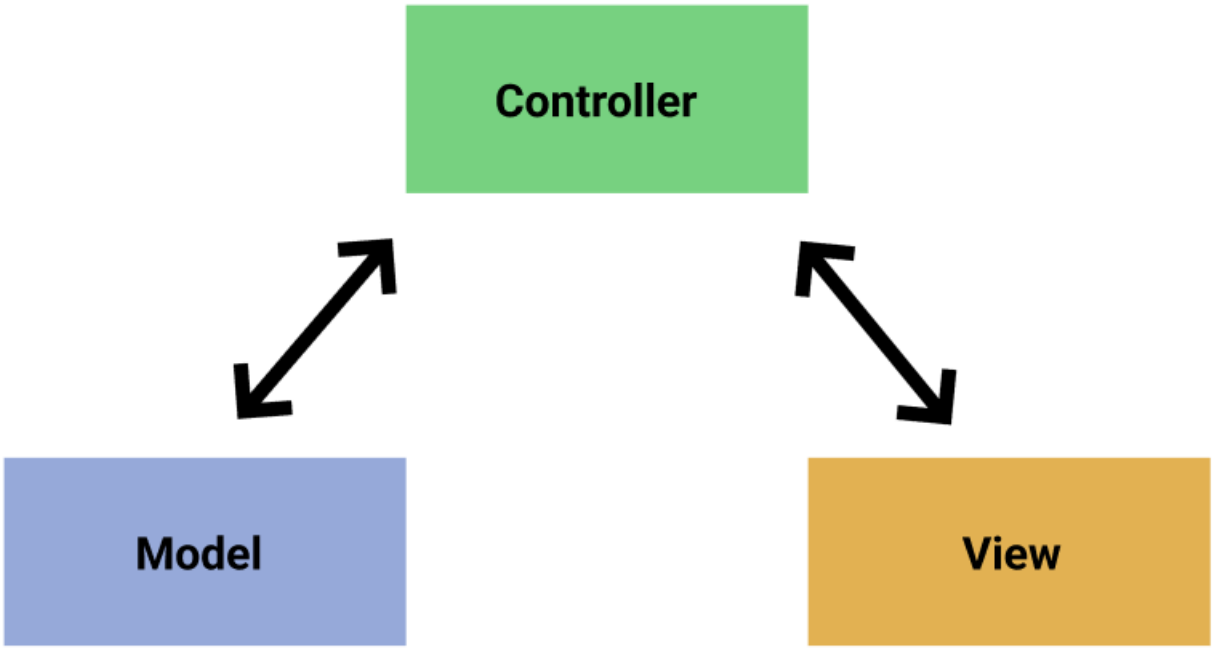
É seguida a mesma linha de raciocínio para o desenvolvimento do módulo, mediante aos recursos e funções consideradas base já desenvolvidas e aplicadas na aplicação em produção. Progredindo em relação a escolha de ambos, é crucial entender a arquitetura de *software* presente na estruturação, sendo ela o padrão MVC (*Model-View-Controller*), sigla pela qual divide em três camadas essenciais a sua identidade: *Model*, *View* e *Controller*.

A camada *Model* é responsável por gerenciar e controlar a forma como os dados se comportam logicamente, por meio das funções e regras de negócios estabelecidas. Ela também é responsável por receber os dados da camada *Controller*, realizando validações e enviando respostas de forma adequada. A camada *Controller* é responsável por intermediar as respostas fornecidas pela *Model* e receber requisições enviadas pela *View*. Por fim, a camada *View* apresenta as informações de forma visual ao usuário. Diante disso, a construção da *View* requer recursos ligados a interface do sistema, como telas, botões ou mensagens.

A arquitetura MVC aumenta consideravelmente a possibilidade de reutilização e manutenção do projeto, pelo fato de separar com aptidão as camadas de apresentação, lógica de negócio e gerenciamento do fluxo da aplicação. Em síntese, a dinâmica entre as camadas é básica, todas as requisições da aplicação são direcionadas para a camada *Controller*, que precisa acessar a camada *Model* para processar a requisição e posterior, exibir o resultado na camada *View*.

O mesmo ocorre para o desenvolvimento do módulo de *help desk*, principalmente pelo tratamento de cada particularidade em sua camada. Podemos visualizar melhor a arquitetura na Figura 7.

Figura 7 – Arquitetura MVC presente no desenvolvimento do módulo.



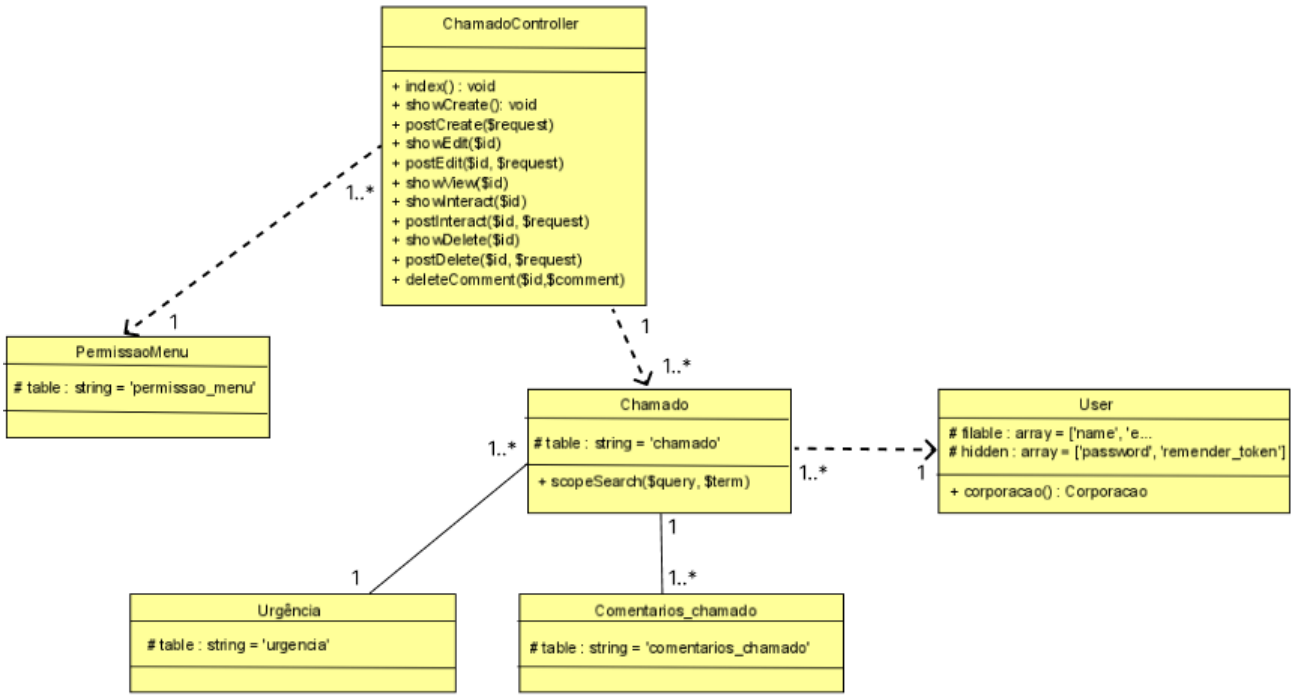
Fonte: Elaborado pelo autor (2022)

Com a visão amplificada da arquitetura, é importante entender como as classes se comunicam. De forma breve, o diagrama de classe é um excelente modelo para esboçar a abstração desejada à nível de implementação. Dessa forma, ele é composto por três partes, sendo elas, o nome da classe que deseja informar, os atributos pertencentes à ela e os métodos relacionados.

Como análise, vários chamados podem ser abertos por um usuário. Tratando o chamado de forma unitária, ele possui o seu grau de urgência, assim como a possibilidade de existir diversos comentários anexos à ele. Como é função da *Controller* administrar as respostas fornecidas pela *Model*, temos o relacionamento entre a *Model* `Chamado` e o `ChamadoController`.

Não tão distante, a mesma relação ocorre entre a *Model*, chamada de `PermissaoMenu` e `ChamadoController`, que referece a *Controller*. A Figura 8 mostra as principais classes trabalhadas no desenvolvimento do módulo, como também suas peculiaridades e o relacionamento entre elas.

Figura 8 – Diagrama de Classes desenvolvimento do módulo no SisGera.



Fonte: Elaborado pelo autor (2022)

Vale ressalta que as entidades `PermissaoMenu` e `User` são referentes aos trabalhos anteriores, sendo fundamental a interpretação do posicionamento de ambos para a construção das demais entidades ilustradas, que fazem parte do desenvolvimento deste trabalho, em questão.

Ciente da arquitetura vigente na aplicação, é esperado o uso de três classes distintas, onde cada uma equivale a uma camada do padrão MVC. Com a comunicação entrelaçada entre elas, temos a apresentação da nova funcionalidade, sendo ela a geração e controle do chamado. Vale ressaltar que a regra de negócio se diferencia, principalmente por existir um processo de chamado bem definido e desenvolvido adentro do SisGera.

Recortando trechos do código, é possível salientar o uso de métodos básicos na aplicação, como o POST, o GET e o DELETE, encorpados no CRUD (*Create, Read, Update, Delete*), acrônimo que faz referência a quatro operações básicas usada em bases de dados relacionais. O CRUD faz referência as operações de criar, ler, atualizar e remover.

Na Figura 9, podemos observar a função `postCreate(PostCreateChamado $request)`, ela realiza a criação do chamado no SisGera.

Figura 9 – Função `postCreate(PostCreateChamado $request)` presente na classe `ChamadoController`.

```php
public function postCreate(PostCreateChamado $request){
    try {

        $chamado = auth()->user()->chamados()->create($this->sanitizer->postCreate($request->all()));

        $request->session()->
            flash('sucessoCadastro', 'Chamado criado com sucesso!');

            return redirect('/chamados');

    } catch (\Throwable $th) {

        $request->session()->flash('errorCadastro', 'Erro ao criar o cadastro!');

        return redirect()->back()->withInput();
    }
}
```

Fonte: Elaborado pelo autor (2022)

Para habilitar a leitura do chamado, desenvolvido na `view.blade`, classe da *View* responsável por visualizar os chamados, foi implementada a função `showView($id)` na `ChamadoController`, classe que recebe as requisições da *View*.

Podemos observar a manipulação do chamado e o uso do método GET na construção da função, como exibido na Figura 10.

Figura 10 – Função `showView($id)` presente na classe `ChamadoController`

```php
public function showView($id){

    $chamado = $this->chamado->findOrFail($id);

    $arrayPagina = [
        'breadcrumb' => 'Visualizar Chamado',
        'cabecalho' => 'Visualizar Chamado',
        'titulo' => 'Visualizar Dados',
        'acao' => "/chamados/{$id}/visualizar",
        'metodo' => 'GET'
    ];

    return view('chamados.view', compact('arrayPagina', 'chamado'));
}
```

Fonte: Elaborado pelo autor (2022)

As funções `postEdit($id, PostEditChamado $request)` e `postDelete ($id, PostDeleteChamado $request)` trabalham com a mesma dinâmica de implementação. Adentro do argumento de cada, é possível observar a chamada de métodos diferentes, sendo eles `PostEditChamado` e `PostDeleteChamado`, respectivamente. Entretanto, é definida de forma clara a chamada do método POST em ambos argumentos para a realização do processo de editar e excluir o chamado.

A Figura 11 auxilia no entendimento do desenvolvimento da função, fazendo referência a função `postDelete($id, PostDeleteChamado $request)`.

Figura 11 – Função `postDelete($id, PostDeleteChamado $request)` presente na classe `ChamadoController`

```
public function postDelete($id, PostDeleteChamado $request){
    try {
        $this->chamado->where('id', $id)->delete();

        $request->session()->flash('sucessoExclusao', 'Chamado excluído com sucesso!');

            return redirect('/chamados');

    } catch (\Throwable $th) {

        $request->session()->
            flash('erroExclusao', 'Erro ao remover o cadastro!');

        return redirect()->back()->withInput();
    }
}
```

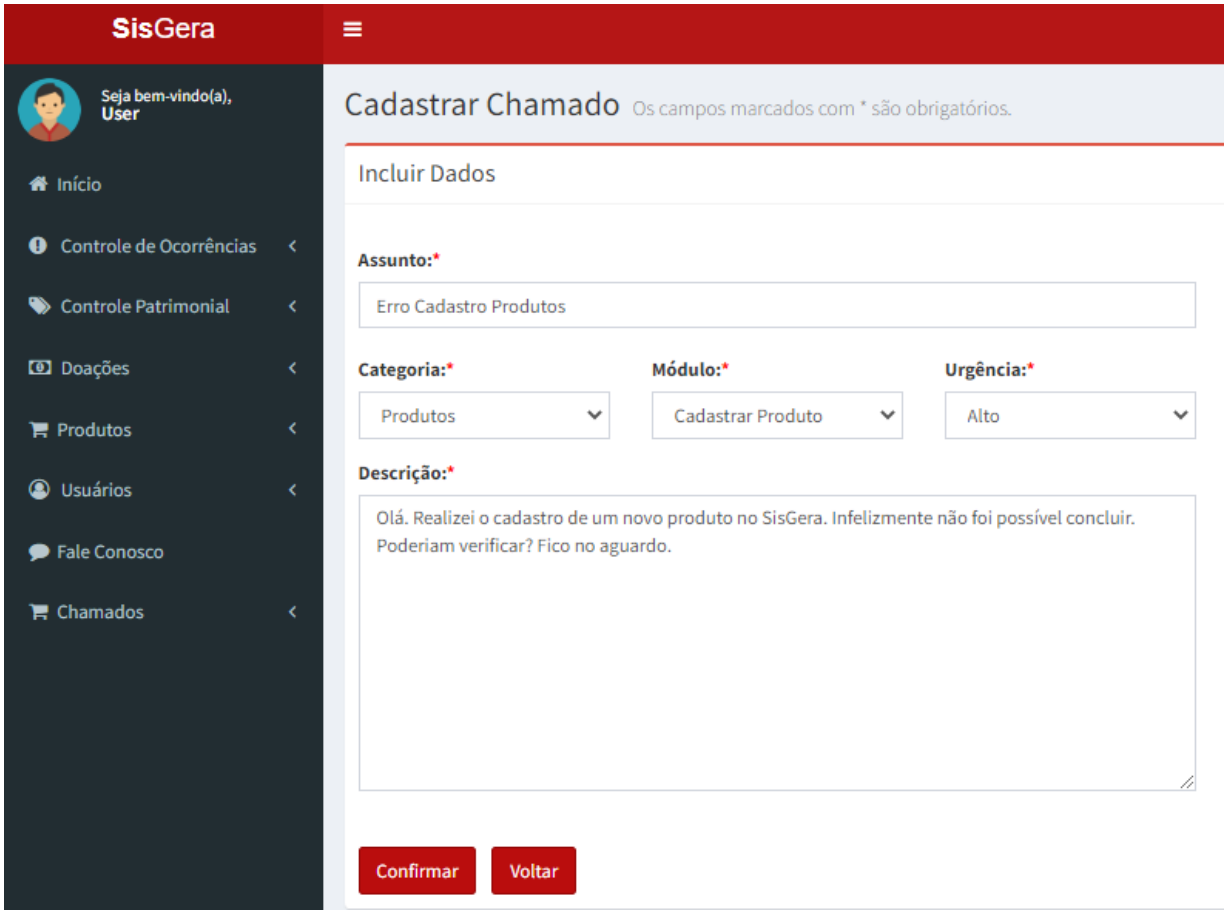
Fonte: Elaborado pelo autor (2022)

Com as devidas demostrações de trechos do código logo acima, podemos focar no resultado encontrado na implementação. Voltando a atenção para a análise das funcionalidades e como a usabilidade delas sucedem. Dessa forma, vamos demostrar algumas funcionalidades presentes no SisGera.

É possível que um usuário do sistema gere um novo chamado, sendo capaz de determinar seus devidos atributos, sendo eles, categoria, módulo, assunto, urgência e descrição. Mede-se a efetividade e objetividade no chamado devido ao fato do usuário selecionar as opções visivelmente presentes na interface do SisGera, fazendo com que ele economize tempo na abertura de um chamado e com uma comunicação única e direta. Ressalta-se o controle de acesso no SisGera, registrando no ato de geração o usuário solicitante.

A Figura 12 demostra a tela `Cadastrar Chamado`, com suas devidas atribuições.

Figura 12 – Tela `Cadastrar Chamado` no SisGera

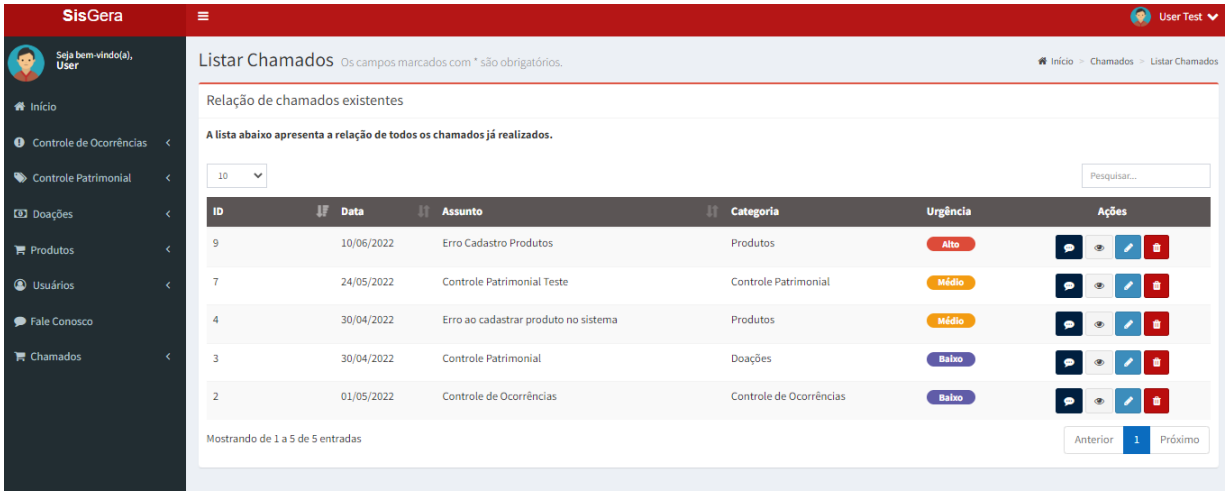


Fonte: Elaborado pelo autor (2022)

A par do cadastro do chamado, é possível realizar outras ações básicas, como listar, editar e excluir. Apresentando a funcionalidade de visualizar o chamado, aplicada na tela `Listar Chamados` tem-se como objetivo constatar todos os chamados do sistema, na qual o usuário tem a possibilidade de editar, excluir, acessar ou aplicar um *feedback* ao chamado.

A Figura 13 contempla a tela citada, sanando também mais duas funcionalidades implementadas, sendo elas editar e excluir.

Figura 13 – Tela `Listar Chamados` no SisGera

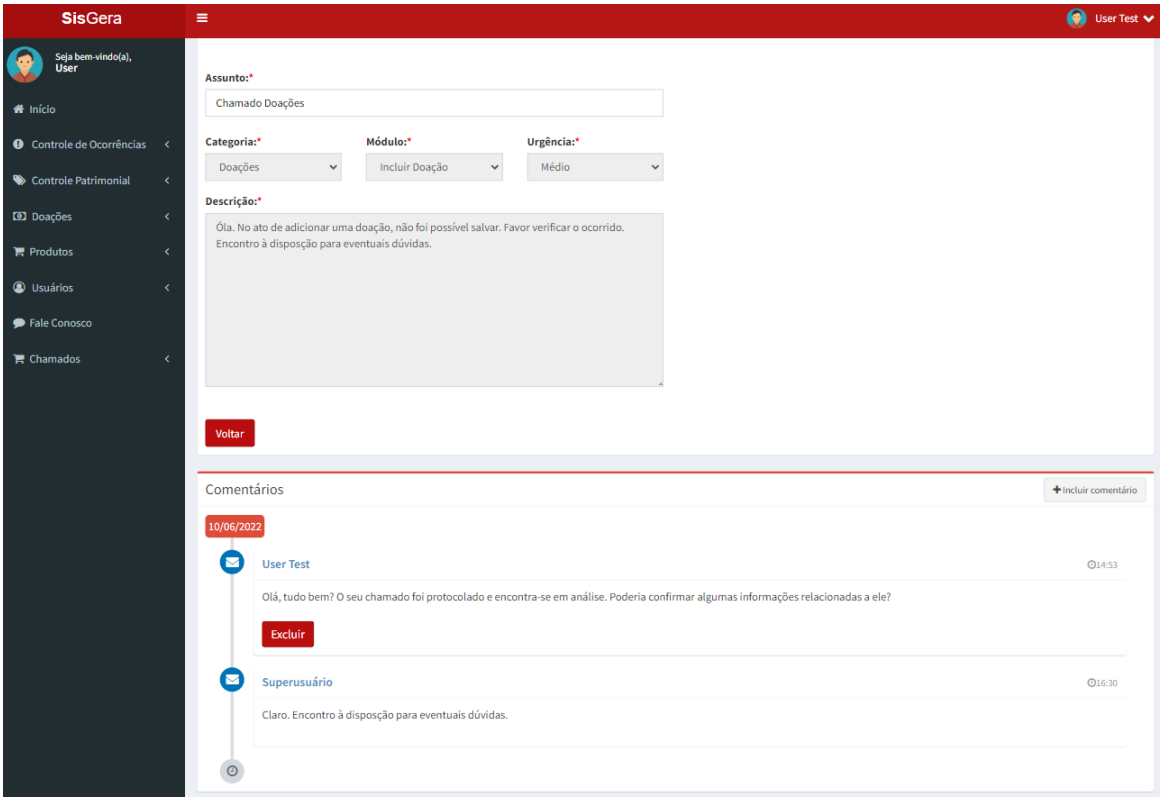


Fonte: Elaborado pelo autor (2022)

O surgimento de dúvidas durante o processo é natural, diante do nível de complexidade presente em cada chamado, principalmente pelos atores envolvidos na dinâmica serem distintos. A fim de melhorar a experiência entre o remetente e o destinatário do chamado, evitando que exista uma lacuna na comunicação de ambos, surge a possibilidade de inserir *feedbacks* ao chamado, evoluindo o entendimento e a desenvoltura do atendimento por meio da percepção dos envolvidos. Condizente a linha de raciocínio, foi desenvolvida a função referente a inserir *feedback*.

Na Figura 14, é possível observar o progresso da comunicação entre os dois atores envolvidos.

Figura 14 – Tela `Atribuir Feedback` ao chamado no SisGera



Fonte: Elaborado pelo autor (2022)

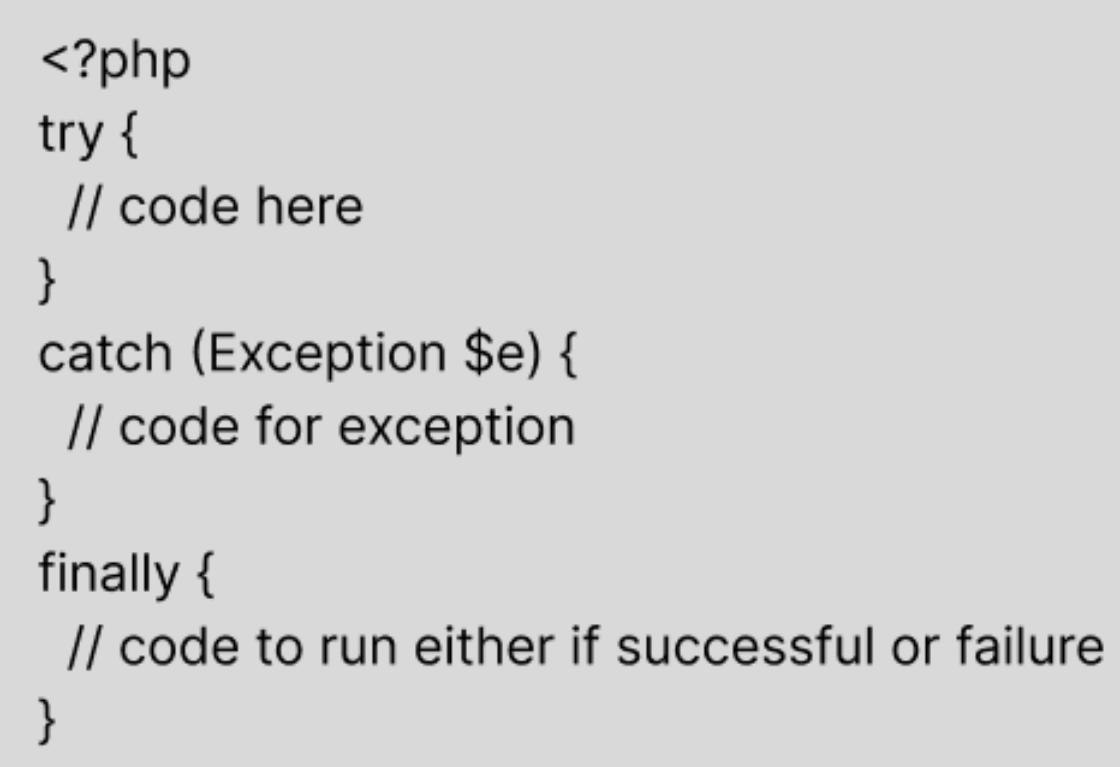
3.6 Integração do Sistema de Rastreamento de *Bug*

Com foco na inteligência operacional e na automação da evolução do chamado, no que sucede o ato da abertura e advém a solução, repercute a integração com o Mantis. De maneira clara e seguindo o processo antes do desenvolvimento do módulo atual, quando ocorre a geração de uma solicitação, é comum realizar o contato com o ator que é responsável por sanar a dor. Essa comunicação muitas vezes ocorre de forma informal, e pode ocasionar na descentralização da informação. Pensando nisso, a integração do SisGera com o Mantis elimina esse circuito de comunicação, padronizando a informação em todas as camadas à nível operacional, tático e estratégico.

Como ilustrado em tópicos anteriores, sistemas estão sujeitos a erros. De forma geral, isso ocorre quando durante o processo de uma ação dentro do código, o esperado não ocorre e requer uma atenção. Podemos nomear o erro como exceção, sendo possível visualizar dentro do PHP a partir do conceito `try...catch...finally`.

A Figura 15 mostra um exemplo de exceção, para melhor entendimento.

Figura 15 – Tela `Atribuir Feedback` ao chamado no SisGera



Fonte: Elaborado pelo autor (2022)

Como as exceções não apontam o caminho a ser seguido, o seu uso de forma genérica não agrega um norte sólido para tratamento e é passível de interpretação por parte do leitor. É conveniente que ela seja personalizada ao máximo, facilitando a compreensão do erro pelo fato de mostrar de forma rápida e simples, atrelada ao motivo e a origem do problema. Informações adicionais podem ser relevantes e auxilia no momento de realizar a correção.

A fim de rastrear e mapear esses pontos para intercessões, temos a coleta de dados no momento exato da exceção e o deslocamento de toda a carga de dados relacionada para o Mantis. Isso é possível devido a classe `App\Exceptions\Hendler` do Laravel. Ela possui duas funções para personalizar o comportamento da exceção, sendo o `report` e o `render`. O `render` exibe de forma amigável ao usuário o tipo de exceção causada como resposta. Enquanto o `report` permite customizar o envio da exceção para um gerenciador de erros, como o Mantis.

Recortando parte do código presente na classe `App\Exceptions\Hendler`, observamos o uso de ambas as funções na Figura 16.

Figura 16 – Uso das funções `render` e `report` na classe `App\Exceptions\Hendler` do Sisgera

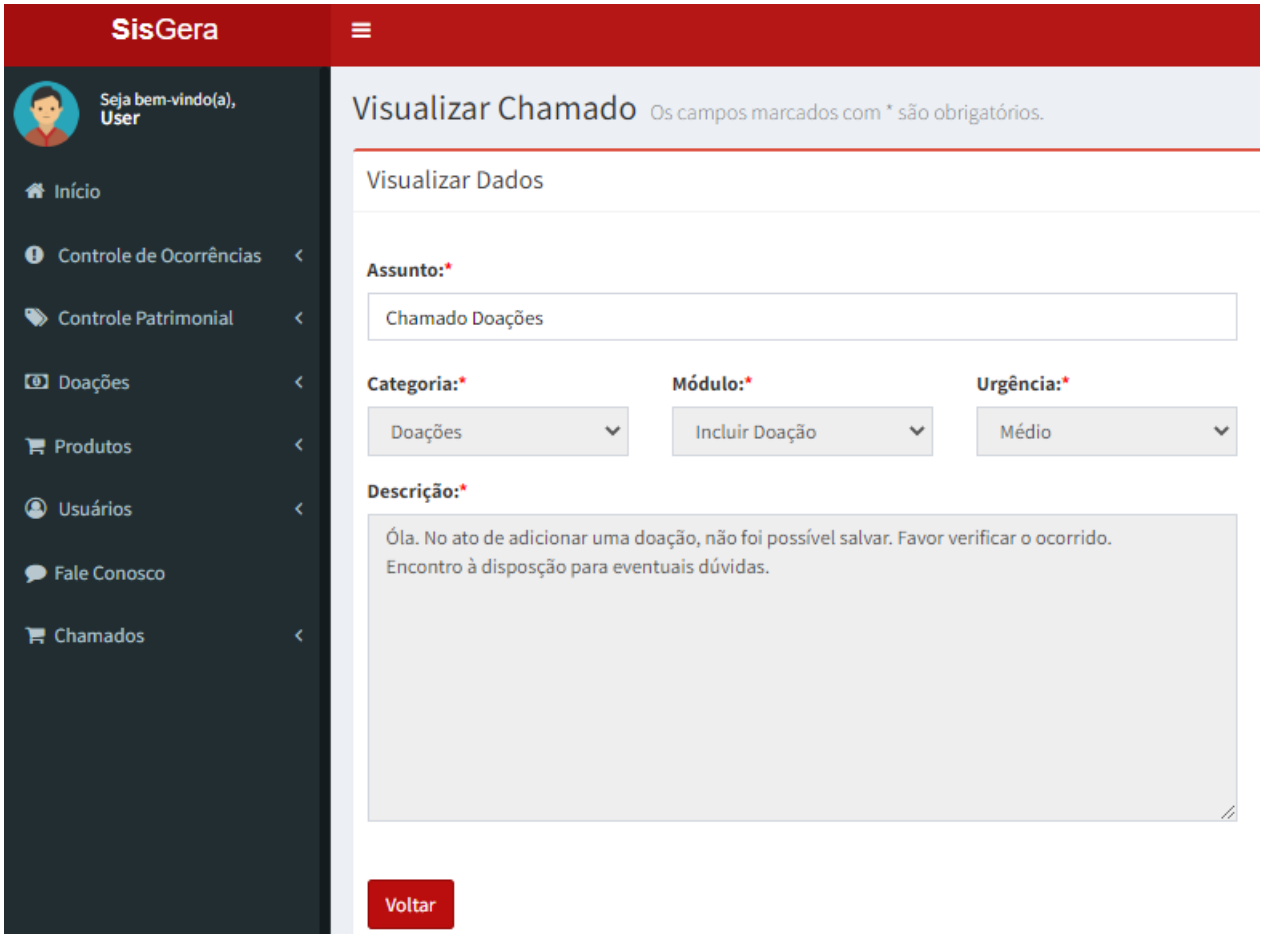
```php
/**
 * Report or log an exception.
 *
 * This is a great spot to send exceptions to Sentry, Bugsnag, etc.
 *
 * @param  \Exception  $exception
 * @return void
 */
public function report(Exception $exception)
{
    //\BugCatcher::createAnIssue($exception);
    parent::report($exception);
}

/**
 * Render an exception into an HTTP response.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  \Exception  $exception
 * @return \Illuminate\Http\Response
 */
public function render($request, Exception $exception)
{
    return parent::render($request, $exception);
}
```

Fonte: Elaborado pelo autor (2022)

Em alguns casos, ao receber uma exceção, é desejado que o processo continue, sem prejudicar a experiência do usuário. Por isso, adentro da classe `App\Exceptions\Hendler`, o *framework* disponibiliza um `helper: report(`*exception*`)`. Isso faz com que seja possível enviar uma exceção para `App` e a partir disso, ela seja enviada para o Mantis como um novo chamado de forma automatizada. O mesmo ocorre para a criação dos chamados na Tela Visualizar Chamado. Para esclarecimento e validação da implementação, vamos realizar a entrada de um chamado e visualizar a sua atribuição no Mantis.

A Figura 17 mostra o chamado na Tela Visualizar Chamado adentro da interface do SisGera. Observe os campos exibidos no chamado e como eles comportam no Mantis, sendo possível assimilar ambos.

Figura 17 – Chamado teste na Tela `Visualizar Chamado` no SisGera



Fonte: Elaborado pelo autor (2022)

Entende-se que de acordo com a Lei Geral de Proteção de Dados, Lei nº 13.709/2018, existe a necessidade de assegurar que os dados sensíveis presentes no chamado sejam protegidos. Como o destinatário do chamado é um usuário à nível técnico, ou seja, presente exclusivamente para solucionar assuntos relacionados ao tema, não existe a necessidade da sua navegação no SisGera ocorrer de forma livre. Também é assegurado como resultado, a economia de recursos na realização das solicitações.

Após a leitura do chamado criado e visualizado no SisGera, temos a transposição de sua cópia para o Mantis, sem a perda das informações e certificando segurança em ambas as extremidades, sendo elas o desenvolvimento do módulo e a integração do sistema de rastreamento de *bug*.

A Figura 18 ilustra o mesmo chamado, porém, adentro do Mantis, como esperado.

Figura 18 – Transposição da cópia do chamado para o Mantis

Fonte: Elaborado pelo autor (2022)

4 Considerações Finais

Este trabalho apresentou o desenvolvimento de novas funcionalidades no projeto SisGera, sendo que tais visam apoiar o operador do sistema a relatar suas dificuldades e manter uma comunicação bilateral entre o operador do sistema e o ator responsável por sanar os problemas encontrados. O módulo ainda não foi implantado, porém, sua expectativa é que otimize o canal de comunicação de chamados do SisGera, organize de forma gradual as demandas vinculadas as melhorias sistemáticas do sistema, fazendo com que seja possível mapear e metrificar o esforço do chamado e seu impacto. Essa informação é de suma importância, pois é com ela que os gestores do SisGera podem alocar recursos disponíveis para tal atividade, com o esforço calculado. Ressalta-se o uso de processos automatizados dentro do desenvolvimento do *help desk*, principalmente por entrelaçar os chamados com o Mantis, ferramenta de rastreamento de *bug* estudada e implementada.

Destaca-se também uma revisão bibliográfica rica em assuntos literários, elicitando temas como Engenharia de *Software*, prototipagem, levantamento de requisitos, o que é *Web* e como ela funciona nos dias atuais, as aplicações presentes no cenário *Web*, APIs REST e seus métodos, sistemas para *help desk* e rastreamento de *bugs*. O desenvolvimento envolveu assuntos voltados a arquitetura `Model-View-Controller`, modelagem de banco de dados, diagrama de classe, trechos de código reforçando a teoria levantada sobre o módulo e também em relação ao Mantis.

A construção dos capítulos ocorreu de forma intrínseca, sendo necessário apresentar o SisGera e o principal problema vigente que o trabalho deseja sanar. Como de esperado, o processo foi definido de forma sólida, existindo um fluxo direcionado para os chamados. Com as tecnologias e abordagens levantadas, foi possível concretizar a criação das funcionalidades, bem como aprimorar as técnicas para modelagem de aplicações. Antemão, foram levantados alguns pontos como propostas para trabalhos futuros, sendo eles apresentados na seção 4.1.

4.1 Trabalhos Futuros

Durante o desenvolvimento deste trabalho foram identificados alguns pontos para trabalhos futuros, sendo eles:

- Testes e avaliações de usabilidade com os usuários do SisGera;
- Implantação do Mantis na aplicação em produção.

Referências

ARANTES, V. M. Um sistema de informação para apoio ao registro de ocorrências atendidas por grupos de bombeiros voluntários. João Monlevade, MG, 2018. Disponível em: <https://goo.gl/Yu2SFe>. Citado 3 vezes nas páginas 13, 16 e 34.

CAVALARI, G. O.; COSTA, H. A. Modelagem e desenvolvimento de um sistema help-desk para a prefeitura municipal de lavras. *Revista Eletrônica de Sistemas de Informação*, v. 4, n. 2, 2005. Citado na página 21.

CERI, S.; FRATERNALI, P.; BONGIO, A. Web modeling language (webml): a modeling language for designing web sites. *Computer Networks*, Elsevier, v. 33, n. 1-6, p. 137–157, 2000. Citado na página 20.

COHN, M. *User Stories Applied: For Agile Software Development*. [S.l.: s.n.], 2004. Citado na página 28.

CRUZ, T. *Sistemas de informações gerenciais: tecnologias da informação ea empresa do século XXI* . [S.l.]: Editora Atlas SA, 2000. Citado na página 16.

FICHMAN, R. G.; KEMERER, C. F. Adoption of software engineering process innovations: The case of object orientation. *MIT Sloan management review*, Massachusetts Institute of Technology, Cambridge, MA, v. 34, n. 2, p. 7, 1993. Citado na página 23.

FILHO, W. de P. P. *Engenharia de software*. [S.l.]: LTC, 2003. v. 2. Citado na página 19.

HUMPHREY, W. S. *A discipline for software engineering*. [S.l.]: Pearson Education India, 1995. Citado na página 23.

KOTONYA, G.; SOMMERVILLE, I. *Requirements engineering: processes and techniques*. [S.l.]: John Wiley & Sons, Inc., 1998. Citado na página 18.

KRASNER, H. The cost of poor quality software in the us: A 2018 report. *Consortium for IT Software Quality, Tech. Rep*, v. 10, 2018. Citado na página 23.

LAUDON, K.; LAUDON. *Sistemas de informação gerenciais*. [S.l.]: Pearson Prentice Hall, 2010. Citado na página 16.

MELO A. L.; MENDES, L. A. M. Sistema help-desk: um estudo de caso da empresa marluvas. 2015. Citado na página 21.

OLIVEIRA, S. S. M. Sistema de informação para controle de materiais e doações aos bombeiros voluntários. 2018. Disponível em: <https://www.monografias.ufop.br/handle/35400000/1188>. Citado 3 vezes nas páginas 13, 16 e 34.

O'REILLY, T. *Web 2.0 compact definition: Trying again*. 2006. Citado na página 19.

SOMMERVILLE, I. *Engenharia de Software. 9ª Edição, 2011. ed*. [S.l.]: Pearson Education–BR, 2011. Citado na página 18.

SOMMERVILLE, I.; PRECHELT, L. Verification and validation. *Software Engineering*, Pearson Addison Wesley Harlow, Essex, v. 7, 2004. Citado na página 16.

WARD, S.; KROLL, P. Building web solutions with the rational unified process: Unifying the creative design process and the software engineering process. *Rational Software Corporation*, 1999. Citado na página 20.