# LISTA DE FIGURAS

[Figura 1 - Diagrama de caso de uso](#_4i7ojhp)

Figura 2 - Diagrama de Classes

Sumário

[LISTA DE FIGURAS 1](#_Toc466999212)

**[1.](#_Toc466999213)****[INTRODUÇÃO](#_Toc466999213)** [3](#_Toc466999213)

**[1.1.](#_Toc466999214)****[Problematização](#_Toc466999214)** [4](#_Toc466999214)

**[1.2.1.](#_Toc466999215)****[Formulação Do Problema](#_Toc466999215)** [5](#_Toc466999215)

**[1.2.2.](#_Toc466999216)****[Solução Proposta](#_Toc466999216)** [6](#_Toc466999216)

**[1.2.](#_Toc466999217)****[Objetivos Gerais](#_Toc466999217)** [7](#_Toc466999217)

**[1.2.1.](#_Toc466999218)****[Objetivos Específicos](#_Toc466999218)** [7](#_Toc466999218)

**[1.3.](#_Toc466999219)****[Metodologia](#_Toc466999219)** [8](#_Toc466999219)

**[2.](#_Toc466999220)****[FUNDAMENTAÇÃO TEÓRICA](#_Toc466999220)** [11](#_Toc466999220)

**[2.1.](#_Toc466999222)****[ITIL](#_Toc466999222)** [11](#_Toc466999222)

**[3.1.1.](#_Toc466999224)****[Operação de Serviço](#_Toc466999224)** [11](#_Toc466999224)

**[2.2.](#_Toc466999227)****[GitHub](#_Toc466999227)** [12](#_Toc466999227)

**[2.3.](#_Toc466999228)****[Qt e Qt Creator](#_Toc466999228)** [13](#_Toc466999228)

**[2.4.](#_Toc466999229)****[C++](#_Toc466999229)** [14](#_Toc466999229)

**[2.5.](#_Toc466999230)****[PostgreSQL](#_Toc466999230)** [15](#_Toc466999230)

**[2.6.](#_Toc466999231)****[Comparação entre Web e Desktop](#_Toc466999231)** [16](#_Toc466999231)

**[3.](#_Toc466999232)****[DESENVOLVIMENTO (MELHORAR TÍTULO DO QUE?)](#_Toc466999232)** [18](#_Toc466999232)

**[3.1](#_Toc466999233)****[Projeto KAIZEN](#_Toc466999233)** [19](#_Toc466999233)

**[3.1.1.](#_Toc466999234)****[Análise de Requisitos](#_Toc466999234)** [22](#_Toc466999234)

**[3.1.2.](#_Toc466999235)****[Modelagem UML](#_Toc466999235)** [23](#_Toc466999235)

**[3.2](#_Toc466999236)****[Implementação](#_Toc466999236)** [32](#_Toc466999236)

**[3.2.1](#_Toc466999238)****[Execução do Sistema](#_Toc466999238)** [32](#_Toc466999238)

**[4.](#_Toc466999239)****[CONCLUSÃO](#_Toc466999239)** [33](#_Toc466999239)

**[BIBLIOGRAFIA](#_Toc466999240)** [34](#_Toc466999240)

# **INTRODUÇÃO**

O mercado tem cobrado cada vez mais valor agregado aos serviços prestados pelas empresas e profissionais de TI. Para auxiliar as organizações a alcançar esse objetivo surgiu o ITIL (*Information Technology Infrastructure Library*), um guia de melhores práticas que possui como principal objetivo a entrega de serviços de TI com melhoria continuada e a geração de maior valor agregado no nos serviços prestados**. (REFERENCIA)**

Para manter essa melhoria continuada é importante manter o controle dos problemas que vão surgindo, bem como eles podem ser solucionados rapidamente, atribuindo a solução desses problemas às pessoas certas. Para isso é importante ter uma ferramenta que centralize a base de conhecimento de problemas e facilite o direcionamento para rápidas soluções.

Com uma base de conhecimento dos problemas bem alimentada e organizada, se torna mais fácil a prevenção de problemas recorrentes, bem como o planejamento de o que deve ser lançado na próxima versão do seu serviço ou o que pode ser adiado por ser um “bug monitorado[[1]](#footnote-1)”.

Para isso é importante uma sintonia em como a organização funciona e como a aplicação mantem os dados, por exemplo, não adianta nada a aplicação ter uma hierarquia de permissões que não reflete como a organização trabalha, dessa maneira utilizar o ITIL como base para construir a aplicação facilita essa sintonia, principalmente por ser uma framework tão utilizada no mercado.

Independente de uma grande ou pequena empresa, a alta competitividade do mercado exige que você tenha um grande controle de tudo que acontece em seu negócio. Quando as coisas vão bem acaba-se deixando de lado um pouco esse controle, mas é quando as coisas começam a dar errado, sem esse controle perde-se tempo, dinheiro e recursos em problemas que poderiam ter sido evitados ou resolvidos de forma bem mais eficiente.

Para ajudar nesse controle existem sistemas de *helpdesk* que ajudam a criar uma base de dados de problemas encontrados e resolvidos e, principalmente, a direcionar e controlar as pessoas que vão cuidar de cada problema.

No presente trabalho é dada uma visão geral das boas práticas apresentadas pela biblioteca ITIL (*Information Technology Infrastructure Library*), uma série de livros que ajudam a organizar o parque tecnológico das empresas de forma controlada e eficiente, nessa biblioteca não é dito o como fazer, mas sim o que fazer, ficando aberto a interpretações para melhor adaptar o que é dito para a real necessidade da empresa.

Através do que está nessa biblioteca o presente trabalho guia de tal forma que auxilie o *helpdesk* da empresa a alinhar as necessidades da mesma com as boas práticas propostas pelo ITIL.

## **Problematização**

Com a falta de conhecimento dos usuários em relação à tecnologia, existem alguns problemas recorrentes, como por exemplo, os usuários não saberem manusear o sistema informativo corretamente, demorarem a entender o funcionamento, entre outros.

Com a falta de organização, os técnicos dos *helpdesks* tem dificuldade de controlar a prioridade dos chamados, tendo dificuldade em saber o que é realmente prioritário, bagunçando a ordem do que deve ser feito e correndo o risco de enviar um problema para um departamento que não é responsável por esse problema, atrasando a sua solução.

Um bom exemplo disso é quando temos vários times cuidando de cada parte do desenvolvimento de um sistema, chega um relato de um cliente sobre um problema no módulo A desse sistema e o helpdesk manda para o time que cuida apenas do módulo B.

Além de todos esses problemas sempre há o problema de duplicidade de chamados, vários chamados tratando um problema crítico e no fim o *helpdesk* acaba perdendo o controle do que foi de fato resolvido.

Na parte técnica, a falta de informação pode ocasionar problemas em relação onde se encontrar os gargalos (baixo desempenho) na infraestrutura. A importância de informar todos os dados relacionados a um problema é fundamental, pois com a ausência de dados estatísticos sobre o problema relatado dificulta ações proativas sobre o mesmo.

### **Formulação Do Problema**

Observando o dia-a-dia de um *helpdesk* é possível notar que realmente há um grande problema de falta de controle na fila de prioridades nos atendimentos dos chamados, muitas vezes as pessoas do atendimento funcionam ou em *FILO*[[2]](#footnote-2), ou seja, um chamado chega, os outros são abandonados e o chamado mais recente é resolvido, ou no modo *FIFO*[[3]](#footnote-3), os problemas são analisados por ordem de chegada, causando uma grande espera. Para isso existem duas opções possíveis:

* Prioridade por categoria: Cada categoria tem uma prioridade pré-definida em seu cadastro, facilitando o manuseio do sistema, porém acaba diminuindo o controle que o usuário tem sobre a fila de prioridades, a gravidade do problema é determinada pela criticidade do componente ou ativo que teve problema.
* Prioridade por chamado: O usuário que abre o chamado é responsável por avaliar o quão urgente é o chamado, nesse caso deve haver o maior nível de instrução do usuário com relação ao que realmente é crítico ou não na organização.

Nos dois casos tem-se resultados completamente diferentes, no caso foi escolhido determinar a criticidade por categoria, sendo assim o gestor da área ficará responsável por determinar qual categoria é mais crítica ou não, em algum momento podemos abrir a opção de alterar a prioridade por chamado aumentando o leque de aplicações do sistema.

Outro ponto fraco de uma grande parte das empresas está em identificar gargalos na infraestrutura e nos sistemas, com os relatórios gerados pelo sistema desenvolvido neste trabalho pode-se localizar uma área ou ativo que está gerando um número maior de chamados, sendo necessária uma análise mais profunda do porque essa área está tendo tantos problemas, essa análise pode começar exatamente pelos chamados abertos por essa área.

Se em determinada área temos sempre chamados relacionados à impressora, podemos entender que a impressora é um gargalo e deve ser substituída visto que está gerando uma carga excessiva de chamados, já se há uma área está tendo problemas de conexão com a *Internet*, a infraestrutura de rede deve ser analisada para levantar onde está a causa de tantos problemas.

No caso de levantamentos estatísticos, o sistema conta com uma área toda dedicada a relatórios onde é possível cruzar dados para facilitar o trabalho do gestor, permitindo que tenha sempre boa parte dos dados às mãos, além da forma como os relatórios são gerados fica bem fácil a criação de relatórios novos sob demanda dos clientes.

Dentre esses relatórios os principais são os que mostram a relação entre chamados, colaboradores e o tempo de resolução dos chamados, com isso pode-se analisar a eficiência geral dos colaboradores do *helpdesk*.

Por fim, com um *helpdesk* centralizado é possível ter os dados de todos os chamados abertos, concluídos e pendentes, facilitando a pesquisa e transmissão de conhecimento para evitar retrabalhos e chamados recorrentes.

### **Solução Proposta**

A solução proposta para cada problema citado acima é passar para os usuários o guia de treinamento do software e oferecer aula de treinamento prática para maior entendimento e aprendizado do software.

Na questão técnica, para facilitar a organização das prioridades dos chamados, é necessária uma opção para que o usuário possa selecionar a prioridade do chamado, por exemplo, “Muito Urgente”, “Urgente” e “Moderado”, e o departamento responsável para a resolução de cada chamado. Cada empresa pode selecionar o quão fragmentado serão as opções de prioridade, podendo haver prioridades intermediárias às do exemplo anterior.

Para reduzir o índice de duplicidade de chamados, o técnico do *helpdesk* deve ter acesso rápido e prático aos chamados abertos, podendo visualizar a prioridade, descrição do chamado e o responsável pela solução do chamado, evitando assim que vários chamados parecidos sejam abertos.

Para solucionar os problemas de falta de controle e conhecimento sobre a infraestrutura devem existir relatórios que relacionem cada ativo à quantidade de chamados abertos e quanto dos chamados abertos são desse mesmo tipo de ativo, além do tempo de solução dos problemas com cada tipo de ativo.

Com essas informações e todas as outras sobre cada ativo, cada responsável técnico e cada problema, o acesso às soluções anteriores se torna muito fácil, assim como o acesso aos problemas a serem solucionados por cada colaborador, facilitando a cobrança e controle.

## **Objetivos Gerais**

Fornecer uma ferramenta alinhada com os processos do ITIL para a gerência de problemas, ou seja, uma ferramenta que auxilie a organização a manter uma base de dados atualizada com relação aos problemas, suas soluções e seus responsáveis, objetivando agilizar a entrega de um serviço com maior valor agregado e funcionando de maneira o mais livre de erro possível.

### **Objetivos Específicos**

O projeto Kaizen[[4]](#footnote-4) visa trazer maior agilidade ao *helpdesk*, mantendo a organização e a fácil visualização dos chamados abertos e dados estatísticos de cada ativo da organização.

O *helpdesk* existe para que os problemas sejam facilmente identificados e corrigidos, porém, sem um sistema que consiga organizar e gerenciar de forma eficiente os dados de cada incidente, fica impossível prever o próximo incidente, não podendo assim tomar medidas preventivas de forma antecipada.

## **Metodologia**

Neste trabalho foi utilizado o método de pesquisa bibliográfico, focando principalmente em fontes ligadas a pesquisas anteriores e às ferramentas utilizadas para o desenvolvimento do mesmo.

Como o trabalho é qualitativo, procurou-se explicar exatamente as dificuldades do cliente, o profissional do *helpdesk*. Apesar de na maioria das vezes o desenvolvimento ser focado no cliente na ponta, deve-se entender que o real objetivo é facilitar o trabalho do *helpdesk,* impactando assim todo o processo e trabalho dentro da organização.

Foi dado um enfoque descritivo no presente trabalho abordando sobre os problemas encontrados em pesquisas relacionadas que buscou-se resolver neste trabalho, classificando assim a pesquisa como qualitativa, focando na relação entre os usuários e o sistema.

Não existe uma empresa contratante para o presente projeto, por isso foram pesquisados os problemas normalmente encontrados nos *helpdesks* de maneira geral, para assim dar uma direção ao trabalho, a fim de se conseguir solucionar os problemas do mundo real e não apenas os que imagina-se que os profissionais enfrentam.

A presente pesquisa é voltada para empresas de todo o porte e tamanho que possuem problemas com o *helpdesk* atual, ou que queiram implantar um sistema de *helpdesk* focado em governança, controle e resultados.

Apesar do foco ser o ITIL, que é uma biblioteca voltada para empresas de TI, qualquer empresa que necessite de um *helpdesk* pode usar o sistema resultante do presente trabalho, para isso basta entender os conceitos de matriz RACI[[5]](#footnote-5)e alguns outros conceitos presentes nesse trabalho.

Durante o desenvolvimento deste trabalho observou-se o quão as soluções apresentadas condizem com o que já existe, se são uma novidade desnecessária, ou se realmente proporciona uma inovação que pode ajudar na relação dos clientes e o *helpdesk*.

1. **Estrutura do Trabalho**

Nos próximos capítulos são expostos mais detalhes do objeto desta pesquisa, no capítulo Fundamentação Teórica é descrito sobre os conceitos usados para o desenvolvimento do KAIZEN, o ITIL que é uma biblioteca de boas práticas. O GitHub, um sistema de repositório de projetos que utiliza o Git para o upload dos projetos e controle de versão. Qt e Qt Creator uma interface gráfica unificada facilitando o desenvolvimento de aplicações multiplataforma.

O C é estruturado pelo C++ uma linguagem de programação orientada a objeto. PostgreSQL, um SGDB (Sistema de Gerenciamento de Banco de Dados) desenvolvido para ser robusto, rápido e confiável, além de ser código aberto, e por final, uma breve comparação entre *web* e *desktop* mostrando o motivo da escolha do sistema ser desenvolvido para a plataforma *desktop*.

No capítulo Desenvolvimento é falado sobre as ferramentas utilizadas, descrevendo cada método escolhido, técnica utilizada e sobre os problemas enfrentados ao longo do trabalho.

No capítulo Projeto fala-se sobre o sistema Kaizen, que irá auxiliar o *helpdesk* que atua na área de gerenciamento de problemas.

No capítulo Análise de Requisitos descreve-se sobre a lista de requisitos funcionais, especificação de requisitos funcionais, lista de requisitos não funcionais e especificação de requisitos não funcionais.

No capítulo de Modelagem UML é descrita a lista de caso de uso, diagrama de caso de uso, especificação do caso de uso e diagrama de classes, mostrando a funcionalidade do sistema Kaizen.

No capitulo implementação **(COMPLETAR)**

No capitulo execução do sistema **(COMPLETAR)**

Nos próximos capítulos conclusão e bibliografia, mostraremos as referências de onde tiramos as citações e opiniões sobre o trabalho.

# **FUNDAMENTAÇÃO TEÓRICA**

1. **ITIL**

ITIL não é uma metodologia, assim “*não é uma regra obrigatória a ser seguida, é um conjunto de recomendações baseadas em boas práticas de Gerenciamento de Serviços de TI.* ” (FREITAS, 2010, p. 57). Investiga o porquê de incidente, como ele acontece, quando ele acontece, desde quando ele acontece, quem está envolvido nisso, e entender o porquê do problema achando a (causa raiz).

O ITIL tem sido cada vez mais utilizado no mercado por já ter provado que as boas práticas apresentadas pelo mesmo trazem resultados positivos e permanentes nas empresas que o implementam. (**PESQUISAR FONTE CONFIÁVEL**

Nessas boas práticas o negócio é guiado para um caminho em que sempre é buscada uma melhoria continuada, sempre procurando uma entrega com maior valor agregado.

Tendo em vista a melhoria continuada com valor agregado, é necessário manter todos os problemas sob controle, e resolvê-los da forma mais rápida possível para não atrapalhar o ciclo de melhoria e não transformar uma entrega em um pesadelo para o negócio.

* 1. **Operação de Serviço**

O livro que trata do tema do presente trabalho é o de Operação de Serviço, este trata dos processos relacionados a manter o Acordo de Nível de Serviço (ANS ou em inglês *Service Level Agreement* - SLA) intacto.

Nesse livro tem-se:

* **Gerenciamento de Incidentes**: Incidente é considerado qualquer evento que causa interrupção no serviço. O Gerenciamento de Incidentes tem como foco principal restabelecer o serviço o mais rápido possível minimizando o impacto negativo do negócio, sendo uma solução de reparo rápido e garantindo melhores níveis de disponibilidade e de qualidade de serviços. É importante que ele conheça e seja informado de quaisquer mudanças.
* **Gerenciamento de Eventos**: Um evento pode ser definido como qualquer mudança de estado que tem importância para a gestão de um item de configuração, gestão dos serviços entregues ao cliente. “Gerencia evento durante todo seu ciclo de vida”.
* **Gerenciamento de Problemas**: Tem como objetivo prevenir a ocorrência de problemas e seus incidentes resultantes, eliminando incidentes recorrentes e minimizar o impacto dos incidentes que não podem ser prevenidos. (FREITAS, 2010, p. 279).
* **Gerenciamento de Acesso**: Processo responsável por permitir que os usuários façam uso do serviço, dados entre outros ativos de TI, ele ajuda a proteger a confidencialidade, a integridade e a disponibilidade de ativos através da garantia que apenas usuários autorizados sejam capazes de acessar.

1. **GitHub**

O Git nasceu da necessidade de Linus Torvalds (pai do kernel Linux) fazer o controle de versão no desenvolvimento do kernel do Linux. Linus estava atrás de uma solução rápida e eficiente para controlar as alterações no seu projeto, com essa ferramenta ele conseguiu manter o trabalho sempre no mesmo diretório e com um eficiente controle de versões.

Em 2008 nasceu o GitHub, um sistema de repositório de projetos que utiliza o Git para o upload dos projetos e controle de versão, além da funcionalidade básica ele ainda conta com várias funcionalidades sociais, facilitando a comunicação entre usuários e desenvolvedores aumentando assim as chances de se obter um ambiente de desenvolvimento colaborativo.

Para manter as versões do sistema e os documentos relacionados atualizados entre os membros do time de desenvolvimento foi utilizado o GitHub com algumas ferramentas gratuitas para facilitar o sincronismo.

Com o GitHub mantém-se o controle de cada alteração no código fonte, não há riscos de alguém da equipe de desenvolvimento estar trabalhando com arquivos desatualizados e tira da equipe a preocupação de manter um repositório de arquivos para esse fim.

Se a equipe toda utilizar o GitHub corretamente o projeto se mantém mais controlado e a chance de um erro ocorrer por arquivos desatualizados vai ser muito baixa.

1. **Qt e Qt Creator**

O Qt nasceu na Noruega em 1995 como uma interface gráfica unificada para Windows, UNIX e Mac, sendo escrita completamente em C/C++.

Pela sua facilidade no desenvolvimento de aplicações multiplataforma e pelo padrão visual bem simples ele foi escolhido como framework para interface gráfica deste projeto.

O Qt é totalmente modular, cada módulo adiciona métodos diferentes que podem ser utilizados para automatizar diversas tarefas relacionadas a interface gráfica e manipulação de dados.

Os próprios desenvolvedores do Qt disponibilizam uma IDE (ambiente de desenvolvimento) que tem todas as opções necessárias para configurar o ambiente Qt e compilar seu programa para as mais diversas plataformas.

Também é possível editar cada um dos arquivos do projeto podendo assim adicionar novas funcionalidades a cada elemento gráfico e mesmo aplicar temas para alterar completamente o visual e a experiência do usuário de forma semelhante ao que o CSS faz pelo HTML.

Quando se trabalha em Qt na verdade os arquivos fonte são arquivos de macro pré-configurados, quando um programa em Qt é compilado primeiramente o Qt roda um programa chamado *moc* para converter esses macros em C++ puro e incorporar ao código editado para só depois compilar o código completo em C++.

Abaixo é possível ver um exemplo de código QML que será interpretado pelo moc para gerar um código em C++:

Button {  
    text: qsTr("Hello World")  
    style: ButtonStyle {  
        background: Rectangle {  
            implicitWidth: 100  
            implicitHeight: 25  
            border.width: control.activeFocus ? 2 : 1  
            border.color: "#FFF"  
            radius: 4  
            gradient: Gradient {  
                GradientStop { position: 0 ; color: control.pressed ? "#ccc" : "#fff" }  
                GradientStop { position: 1 ; color: control.pressed ? "#000" : "#fff" }  
            }  
        }  
   }

Alguns dos clientes mais famosos do Qt são:

* Skype - Comunicador instantâneo oficial da Microsoft
* Google Earth - Aplicativo da Google
* VLC - Reprodutor de mídias

Como visto o Qt é um framework largamente utilizado e que foi escolhido para ser a framework visual do projeto pela facilidade no desenvolvimento e a característica multiplataforma (codifique uma vez, compile em qualquer lugar).

1. **C++**

Segundo Patrícia Feliciano, a linguagem C originou-se da linguagem B nos laboratórios da AT&T entre 1969 e 1973. Ainda segundo Patrícia, boa parte dos sistemas operacionais mais atuais ainda são escritos em C.

O C evoluiu naturalmente para o C++, acrescentando principalmente a orientação a objetos. Os mantenedores dessa linguagem se comprometeram a atualizar a linguagem C++ a cada 3 anos, sendo assim ela nunca ficará desatualizada.

As grandes vantagens do C/C++:

* Alcança níveis de otimização diferentes em tempo de compilação;
* Pode ser usado para as principais plataformas existentes (Windows, UNIX e Mac);
* Faz parte do kernel da maioria dos sistemas operacionais existentes (devido ao alto desempenho e compatibilidade);
* Grande gama de frameworks que facilitam o trabalho do programados sem perder desempenho.

Muitas vezes quando se fala em C++ usa-se o termo C/C++, já que na verdade não existe uma separação real entre as duas linguagens e grande parte do código escrito em C++ é idêntico ao código em C, em outros lugares também é dito que C++ é C orientado a objeto.

1. **PostgreSQL**

O PostgreSQL nasceu em 1986 na Universidade de Berkeley sob o nome de POSTGRES sob encomenda do Departamento de Defesa dos Estados Unidos, ele é o um SGDB (Sistema de Gerenciamento de Banco de Dados) desenvolvido para ser robusto, rápido e confiável, além de ser em código-aberto.

Entre 1986 e 1995 o código foi melhorado principalmente na questão de confiabilidade e em 1995 o código foi completamente reescrito com redução em 25% do tamanho do código-fonte e rebatizado como Postgre95.

Em 1996 foi novamente rebatizado de PostgreSQL já que usava a linguagem SQL como interface padrão.

Hoje o PostgreSQL aceita praticamente tudo que a linguagem SQL propõe e a prova de sua robustez é que os limites hoje que ele aceita são:

* Tamanho máximo do banco de dados: Ilimitado
* Tamanho máximo da tabela: 32 TB
* Tamanho máximo da linha:  1.6 TB
* Tamanho máximo do campo: 1 GB
* Número máximo de colunas por tabela: Ilimitado
* Máximo de colunas por tabela: 250 a 1600 dependendo do tipo da coluna
* Máximo de índices por tabela: Ilimitado

1. **Comparação entre Web e Desktop**

Desenvolvemos em desktop, pelo fato precisar de uma estrutura mais complexa, isso o desktop oferece e será implementada em vários tipos de empresas.

A complexidade atuando em ambientes com grande importância para atuação da aplicação, que é traduzir em Cliente/Servidor.

Vantagem da web, que a interface é conhecida por uma grande gama de usuários que já sabe o funcionamento dos navegadores.

Vantagem em desktop, a uma rica variedade de controles para a interface com o usuário.

WEB desenvolvimento, manutenção e atualizações da aplicação através do servidor.

Em desktop a atualização é um a um tendo o total controle sobre o posicionamento da aplicação, facilitando a interface com integração para vários hardwares. Na questão gráfica em web a aplicação não pode ficar tão elegante quanto é imaginável, já em desktop a parte gráfica é mais fácil de ser manusear.

Em web a segurança é um grande problema, desde que a internet não é um lugar seguro, qualquer coisa feita online está em risco.

Já aplicações desktop são mais confiáveis e confortáveis, pois se acessa um aplicativo enquanto se tem o computador em alcance sem nenhum tipo de ligação com a internet não correndo o risco de perder dados ou o sistema fechar, travar por causa da perda de sinal com a internet.

A aplicação web se torna mais lenta por conta de ter várias pessoas a gravitar em torno de si, por existir grandes problemas como Segurança e Acessibilidade.

Em desktop os dados são armazenados localmente, portanto o risco de vazar dados não existe, Não precisa necessariamente da internet para o seu funcionamento, tende a ter experiência mais complexa do usuário, permitindo mais confiabilidade na estabilidade do software.

# **DESENVOLVIMENTO (MELHORAR TÍTULO DO QUE?)**

Para o desenvolvimento desse projeto foi escolhida a framework visual Qt, que fornece um ambiente de desenvolvimento muito simples de usar, apesar de algumas falhas na fatoração do código quando é feita alguma exclusão de componentes.

O projeto inicial era uma interface simples e direta que pudesse dar ao usuário acesso a todos os dados e ferramentas a partir de uma única tela, mas conforme o desenvolvimento está avançando algumas janelas acabaram sendo mais complexas do que o esperado.

Para manter a equipe de desenvolvimento em sincronia está sendo utilizada a ferramenta Git, GitHub e GitKraken. A primeira uma plataforma de controle de versão, a segunda um servidor que disponibiliza um espaço para utilização do Git em projetos opensource e o último uma GUI para facilitar o uso dos dois primeiros.

Alguns problemas foram enfrentados quando utilizamos o Git, por exemplo, o Qt Creator (IDE do Qt) cria um arquivo com as configurações pessoais do usuário e mesmo colocando esses arquivos na lista de exceções do Git aparece um erro de alterações não submetidas.

Apesar desse pequeno problema o uso desse sistema de controle de versão facilitou muito o trabalho paralelo da equipe, visto que enquanto um codificava, o outro podia ir documentando o que foi codificado sem problemas, basta para isso o colaborador que está codificando lembrar de enviar as alterações para o GitHub.

Com relação a linguagem de programação, o uso do Qt facilita muito o desenvolvimento de interfaces gráficas simples e com todas as funcionalidades no lugar. Com seu bem desenvolvido sistema de signals e slots fica muito simples atribuir ações a qualquer componente da janela.

Além disso, como o Qt é desenvolvido em C++ e o Kaizen está sendo desenvolvido também em C++ os códigos lógicos e da parte visual trabalham muito bem juntos, sem problema algum de compatibilidade entre um e outro.

O principal problema enfrentado no projeto não foi com relação a tecnologia, ferramentas ou linguagem de programação, mas sim em entender o que realmente seria necessário para o usuário do sistema.

Muitas vezes a abordagem que está sendo feita com relação a um requisito ou outro não era bem o que seria mais funcional para o usuário, sendo assim o desenvolvimento geral foi bem atrasado por mudanças drásticas em como o sistema funciona.

O projeto inicial era relativamente pequeno, sem muitos detalhes, mas conforme os estudos e as versões foram sendo feitas muitos detalhes foram surgindo e o sistema tem tomado uma forma muito diferente do que era a ideia inicial.

Um bom exemplo disso é que inicialmente todas as funções eram acessadas pelos menus da janela inicial, mas isso deixou o ambiente muito poluído então muitas funções foram colocadas em outras janelas, criando um caminho um pouco mais longo, mas mais fácil do começo da operação do sistema até o seu objetivo.

## **Projeto KAIZEN**

O sistema batizado de KAIZEN (significa mudança para melhor) irá auxiliar o *helpdesk* que atua na área de Gerenciamento de Problemas, para se obter um *helpdesk* eficiente é muito importante manter um banco de dados de chamados abertos, resolvidos e pendentes, a fim de evitar retrabalho e duplicidade de chamados.

O KAIZEN foi desenvolvido com funcionalidades que irá auxiliar o *helpdesk* a se organizar na abertura de chamados, no controle da prioridade de cada chamado, da atribuição de um responsável por cada chamado e principalmente em criar um banco de dados de problemas e suas soluções que auxiliarão a propor melhorias e correções para a área responsável pelo ativo que for a causa do problema.

Nos SLAs sempre deve-se deixar claro qual o tempo máximo para atendimento de cada demanda do cliente, seja uma correção, uma nova funcionalidade ou simplesmente uma dúvida a ser tirada, todas essas demandas serão encaminhadas para o *helpdesk* que se encarregará de encaminhar a demanda para a área correspondente.

Sem organização essas demandas podem se perder e os SLAs podem ser quebrados, em uma empresa que está focada em entrega de valor e aumento de competitividade dos seus produtos, um SLA quebrado é como um juramento quebrado, ele fere a imagem da empresa e pode causar dificuldades com futuros e atuais clientes.

Com o KAIZEN todo o controle da demanda estará centralizado em um único sistema e o tempo de resposta para demandas tende a cair drasticamente, da forma como o sistema está concebido ele facilmente pode ser adaptado a cada situação principalmente no que diz respeito a relatórios.

Em um *helpdesk* é importante que os dados dos chamados sejam cruzados para gerarem informações que possam ajudar a eliminar causas de problemas tanto no cenário micro em cada chamado individualmente, como no cenário macro na empresa como um todo.

Através do KAIZEN é possível identificar qual área está tendo mais problemas técnicos, ou qual sistema está tendo maior demanda de desenvolvimentos para solucionar problemas críticos.

Uma consequência indireta do uso de um bom sistema de *helpdesk* é que há uma melhoria significativa na forma como são levantados alguns medidores de desempenho.

Em todo o processo dentro de uma organização é necessária uma avaliação durante a execução do mesmo e ao seu final para avaliar se o processo foi executado corretamente e se não são necessárias melhorias durante o processo ou em sua próxima iteração.

Com o *helpdesk* organizado, fazer o levantamento das informações do nível de eficiência de cada processo pode ser muito menos doloroso visto que os dados dos problemas relacionados a cada processo são de mais fácil acesso.

Para os gestores de cada área será bem menos custoso gerar os relatórios de desempenho tendo os relatórios de problemas que podem ser gerados pelo KAIZEN.

De um modo geral o *helpdesk* conversa com todas as áreas da empresa, pois todas geram problemas, que geram demandas, que se tornam chamados e são documentados no sistema de *helpdesk*.

*O sistema de Helpdesk - KAIZEN (mudança para melhor: Hoje melhor do ontem, amanhã melhor do que hoje!)*

O *helpdesk* é conhecido como o departamento que você liga para quando esqueceu a senha de e-mail, precisa configurar a impressora ou porque a “luizinha” da “caixa preta” não acende. Porém a função desse departamento vai muito além de instalar drivers, trocar senhas e outros equipamentos.

No Help Desk(**TERMO EM INGLÊS)** concentram-se profissionais que, com algum conhecimento específico, podem analisar o problema e cortar pela raiz o que está atrasando o fluxo produtivo da empresa.

Segundo VIVIANE:

“Além de atuar como ponto único de contato do usuário/cliente com a TI, o Help Desk registra e administra incidentes, buscando restaurar a operação normal do serviço o mais rápido possível. Assim, pode-se dizer que a “imagem” da TI é transmitida ao usuário/cliente através do Help Desk e, por este motivo, o departamento merece atenção especial. ” (VIVIANE, 2010)

No *helpdesk* que temos a maior parte do processo de Gerenciamento de Problemas, o objetivo desse processo no ITIL é:

· Minimizar os efeitos adversos nos negócios;

· Tratar incidentes e problemas causados por erros na infraestrutura;

· Prevenir para o ativamente a ocorrência dos incidentes, problemas e erros;

· Reduzir o número geral de incidentes;

Para isso é necessário ter um histórico de incidentes e problemas bem descrito, saber separar o que é o que nesses eventos, interpretar corretamente o resultado de cada ação para evitar novos problemas e com isso reduzir os chamados e o tempo para solucioná-los no *helpdesk*.

Com esses objetivos e necessidades um sistema de *helpdesk* eficiente deve ter:

* Formulário de cadastro de eventos no qual seja fácil identificar onde foi o problema, o que causou o problema, quem resolveu o problema e, principalmente, o que foi feito para resolver o problema;
* Sistema de busca que viabilize localizar problemas similares;
* Controle de usuários para que cada um possa cuidar exclusivamente do que foi atribuído a ele, mas que possa consultar tudo que já foi feito;
* Interface simples e que sempre aparenta ser a mesma independente do sistema operacional e dispositivo utilizado para acessá-lo.

### **Análise de Requisitos**

* **Lista de Requisitos Funcionais**

|  |  |
| --- | --- |
| **Número** | **Nome/Breve Descrição** |
| RF1 | USUÁRIO |
| RF2 | INCLUIR USUÁRIO |
| RF3 | ALTERAR USUÁRIO |
| RF4 | EXCLUIR USUÁRIO |
| RF5 | LISTA DE USUÁRIO |
| RF6 | CONSULTA DE USUÁRIO |
| RF7 | CONTROLE DE ACESSO DA APLICAÇÃO |
| RF8 | ALTERAR SENHA |
| RF9 | ESQUECEU SENHA |
| RF10 | CHAMADO |
| RF11 | INCLUIR CHAMADO |
| RF12 | EXCLUIR CHAMADO |
| RF13 | ALTERAÇÃO DO CHAMADO |
| RF14 | CATEGORIA |
| RF15 | INCLUIR CATEGORIA |
| RF16 | ALTERAR CATEGORIA |
| RF17 | EXCLUIR CATEGORIA |
| RF18 | CRIAÇÃO DE GRUPO DE USUÁRIO |
| RF19 | ATRIBUIÇÃO DE PAPÉIS PARA GRUPO DE USUÁRIO |
| RF20 | RELATÓRIO TEMPO DE RESPOSTA X CATEGORIA |
| RF21 | RELATÓRIO TEMPO DE RESPOSTA ESPERADO |
| RF22 | RELATÓRIO CHAMADO ABERTO |
| RF23 | RELATÓRIO CHAMADO ABERTO X RESPONSÁVEL |
| RF24 | RELATÓRIO CHAMADO FECHADO X RESPONSÁVEL |
| RF25 | RELATÓRIO CHAMADO X USUÁRIO X PAPEL |
| RF26 | RELATÓRIO CHAMADO X STATUS DO TICKET |

* **Lista de Requisitos Não Funcionais**

|  |  |
| --- | --- |
| **Número** | **Nome/Breve Descrição** |
| RNF01 | TREINAMENTO PARA O USUÁRIO |
| RNF02 | CRIAÇÃO DO BANCO DE DADOS PARA O SISTEMA DO USUÁRIO |
| RNF03 | SERVIDOR DE BANCO DE DADOS |
| RNF04 | CONEXÃO COM A INTERNET |
| RNF05 | TER O KAIZEN INSTALADO NO COMPUTADOR |
| RNF06 | CONEXÃO COM DISCO LOCAL PARA BANCO DE DADOS |
| RNF07 | BANCO DE DADOS EM NUVEM |
| RNF08 | UM HD POTENTE PARA O SERVIDOR |
| RNF09 | UMA BOA MEMÓRIA RAM PARA O SERVIDOR |
| RNF10 | COMPATIBILIDADE COM SISTEMA OPERACIONAL LINUX E WINDOWS |

### **Modelagem UML**

* **Lista de Casos de Uso**

|  |  |  |  |
| --- | --- | --- | --- |
| **Número** | **Nome** | **Ator** | **Descrição** |
| UC01 | Gerenciar Pessoa | Usuário Técnico de TI | Caso de Uso responsável por manter o cadastro das pessoas no sistema. |
| UC02 |  |  |  |
| UC03 |  |  |  |

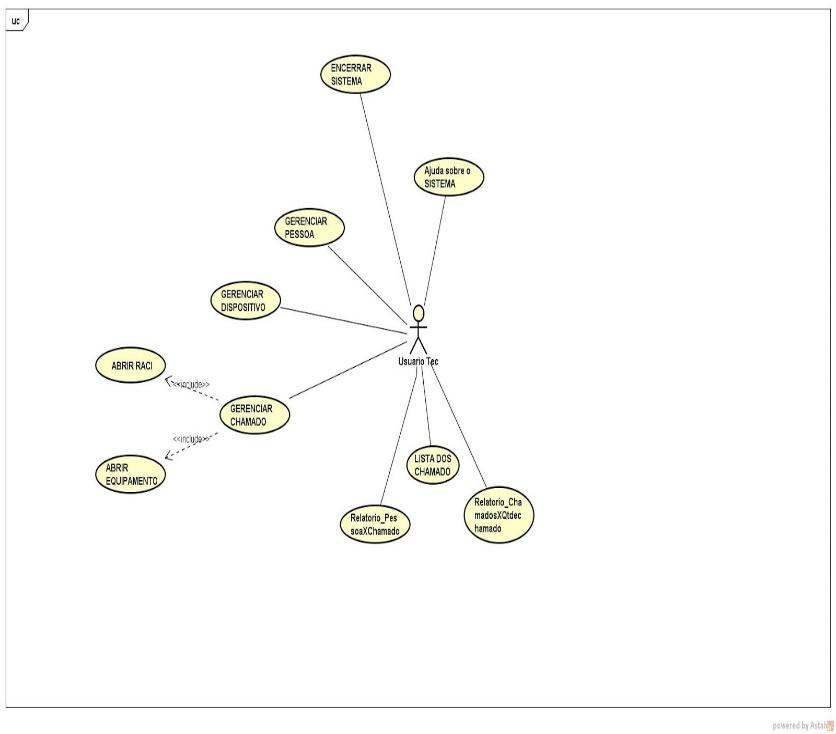
GERENCIAR: PESSOA, DISPOSITIVO, CHAMADO.

RELATÓRIOS: PESSOA X CHAMADO, PESSSOA X QUANTIDADE\_CHAMADO E CHAMADOS

AJUDA SOBRE O SISTEMA

ENCERRAR SISTEMA

* **Diagrama de Casos de Uso**



*Figura 1 - Diagrama de caso de uso*

* **Especificação de Caso De Uso**

**UC01 – Gerenciar Pessoa**

**NOME:** Gerenciar Pessoa

**DESCRIÇÃO SUCINTA:** Usuário cadastra pessoa no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Gerenciar”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Pessoa”

4.      O Sistema abre a tela de cadastramento

5.      O usuário preenche o campo “ID, NOME,TELEFONE, RAMAL,EMAIL,DEPTO, CARGO E GRUPO”

6.     Todos campos são obrigatórios

7.      O usuário preenchendo corretamente os campos tem o botão “Salvar” que é obrigatório a salvar

8.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Gerenciar Pessoa**

1.a. O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário CANCELAR, o sistema fechar a janela, voltando para janela principal

1.b.  O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário EDITAR, o sistema faz uma busca e mostra para o usuário os cadastros de pessoas ativas

1.c.  O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário NOVO, quando o usuário clicar em novo volta para o passo 4.

**(A2) Alternativa ao Passo 8 – Informações Incorretas**

2.a. O sistema retorna uma mensagem de erro “cadastro inconsistente, por favor verificar as informações ” o usuário clica em ok e será redirecionado para a tela anterior.

**(A3) Alternativa ao Passo 9 – A regra RN1 não é atendida**

3.a.  O sistema retorna uma mensagem de erro “cadastro inconsistente, por favor verificar as informações ” o usuário clica em ok e será redirecionado para a tela anterior.

3.b.  O caso de uso é encerrado

**REGRAS DE NEGÓCIO**

**(RN1)** Preenchimento dos campos obrigatórios

FIM DO CASO DE USO PESSOA.

**UC02 – Gerenciar Dispositivo**

**NOME:** Gerenciar Dispositivo

**DESCRIÇÃO SUCINTA:**Usuário cadastra o dispositivo no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Gerenciar”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Dispositivo”

4.      O Sistema abre a tela de cadastramento

5.      O usuário preenche o campo “ID, NOME, TIPO (OPÇÕES) E LOCAL”

6.      os campos são obrigatórios

7.      O usuário preenchendo corretamente os campos tem o botão “Salvar” que é obrigatório a salvar

8.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Gerenciar Dispositivo**

1.a. O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário CANCELAR, o sistema fechar a janela, voltando para janela principal

1.b.  O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário EDITAR, o sistema faz uma busca e mostra para o usuário os cadastro de pessoa ativos

1.c.  O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário NOVO, quando o usuário clicar em novo volta para o passo 4.

**(A2) Alternativa ao Passo 8 – Informações Incorretas**

2.a. O sistema retorna uma mensagem de erro “cadastro inconsistente, por favor verificar as informações ” o usuário clica em ok e será redirecionado para a tela anterior.

FIM DO CASO DE USO DISPOSITIVO.

**Gerenciar – CASO DE USO**

**NOME:** Gerenciar chamado

**DESCRIÇÃO SUCINTA:**Usuário cadastra o chamado no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.      As pessoas relacionadas do chamado precisam já estar cadastradas, os dispositivos dos chamados também.

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Gerenciar”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Chamado”

4.      O Sistema abre a tela de cadastramento

5.      O usuário preenche o campo “ID, MATRIZ RACI, EQUIPAMENTO, TEMPO DECORRIDO, SLA, STATUS (ABERTO, ATRIBUÍDO, EM EXECUÇÃO E CONCLUIDO), REINCIDENTES, ABERTO EM, CONCLUIDO EM E DESCRIÇÃO”

6.      O usuário abre a tela da matriz RACI

7.      O usuário abre a tela de equipamento

8.      O usuário marca a caixa de seleção CheckBox se o chamado já existe

9.      Todos os campos são obrigatórios

10.  O usuário preenchendo corretamente os campos tem o botão “Salvar” que é obrigatório a salvar

11.  E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Gerenciar chamado**

1.a. O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário CANCELAR, o sistema fechar a janela, voltando para janela principal

1.b.  O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário EDITAR, o sistema faz uma busca e mostra para o usuário os cadastro de pessoa ativos

1.c.  O sistema abre a tela de cadastramento de pessoa onde terá a opção alternativa para o usuário NOVO, quando o usuário clicar em novo volta para o passo 4.

**(A2) Alternativa ao Passo 8 – Informações Incorretas**

2.a. O sistema retorna uma mensagem de erro “cadastro inconsistente, por favor verificar as informações ” o usuário clica em ok e será redirecionado para a tela anterior.

**(A3) Alternativa ao Passo 9 – A regra RN1 não é atendida**

3.a.

3.b.  O caso de uso é encerrado

**REGRAS DE NEGÓCIO**

**(RN1)** Preenchimento dos campos obrigatórios

FIM DO CASO DE USO CHAMADO

**Relatório – CASO DE USO**

**NOME:** Relatório pessoa x chamado

**DESCRIÇÃO SUCINTA:**Usuário ver o relatório de pessoa e chamado no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.       Que os dados estejam cadastrados.

**PÓS-CONDIÇÕES:**

1.

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Relatórios”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Pessoa x Chamado

4.      O Sistema abre a tela com a lista de pessoas e chamados

5.      E o caso de uso é encerrado.

**Relatório – CASO DE USO**

**NOME:** Relatório pessoa\_quantidade chamado

**DESCRIÇÃO SUCINTA:**O sistema lista para o usuário um relatório de pessoa e quantidade de chamado.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.       Que os dados estejam cadastrados.

**PÓS-CONDIÇÕES:**

1.

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Relatórios”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Pessoa x Quantidade\_Chamado

4.      O Sistema abre a tela com a lista de pessoas  e quantidade\_chamados

5.      E o caso de uso é encerrado.

**Relatório – CASO DE USO**

**NOME:** Chamado

**DESCRIÇÃO SUCINTA:**O sistema lista para o usuário uma lista de todos os chamados cadastrados.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.       Que os dados estejam cadastrados.

**PÓS-CONDIÇÕES:**

1.

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Relatórios”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Chamado”

4.      O Sistema abre a tela com a lista de Chamados

5.      E o caso de uso é encerrado.

**Ajuda – CASO DE USO**

**NOME:** Ajuda sobre o sistema

**DESCRIÇÃO SUCINTA:**O sistema mostra o FAQ do sistema e o contato do suporte

**ATORES:**

1.      Usuário Técnico de TI

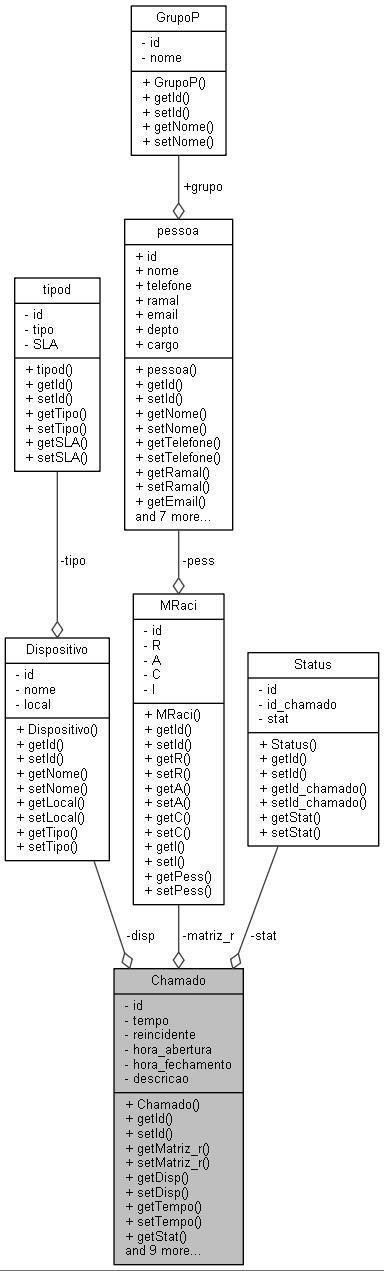
**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Ajuda”

2.      O Sistema exibe o FAQ e o contato do suporte

3.   E o caso de uso é encerrado

(**VERIFICAR CLASES COM AGREGAÇÕES**



*Figura - Diagrama de Classes*

1. **Implementação**

O Kaizen ainda está sendo desenvolvido, como tudo está sendo minuciosamente planejado ele estará disponível em breve

### **Execução do Sistema**

* TESTE DE VALIDAÇÕES

O sistema ainda está em desenvolvidos e não está maduro o suficiente para passar por testes de validações

# **CONCLUSÃO**

No fim do trabalho espera-se entregar um sistema de help desk que de fato ajude o profissional que entrar em contato com ele, no momento não é possível dizer onde o Kaizen vai chegar, mas em breve poderá mudar a forma de se controlar os problemas em todas as organizações que alcançar.

# **BIBLIOGRAFIA**

FERNANDES, Viviane Cardoso. **REDESENHO DE PROCESSOS DO HELP DESK APLICANDO AS BOAS PRÁTICAS DO ITIL.** 2010. Disponível em: <http://www.ulbra.inf.br/joomla/images/documentos/TCCs/2010\_2/ads-viviane%20cardoso%20fernandes.pdf>. Acessado em: 05 out. 2016.

FREITAS, Marcos André dos Santos**. Fundamentos do gerenciamento de serviços TI: preparatório para a certificação ITIL® V3 Foundation**. Rio de Janeiro: Brasport, 2010. 376 p.

MAGALHÃES, Ivan Luizio; PINHEIRO, Walfrido Brito. **Gerenciamento de Serviços de TI (uma abortagem com base na ITIL)**.  NOVATEC, 2007.

ACSOFTWARE. **Webinar - a itil de resolver problemas**. , 2016. Disponível em: <http://https://www.youtube.com/watch?v=Oe4n40fvly4>. Acesso em: 05 out. 2016.

AXELOS. **What is ITIL® Best practice?.** , 2016. Disponível em: <http://https://www.axelos.com/best-practice-solutions/itil/what-is-itil>. Acesso em: 05 out. 2016.

AXELOS. **Disney's ITIL® journey**. , 2016. Disponível em: <http://https://www.axelos.com/case-studies-and-white-papers/disneys-itil-journey-case-study>. Acesso em: 05 out. 2016.

COMPUTERWORLD CIO. **ITIL®: nova versão traz mais complexidade e maiores custos**. , 2016. Disponível em: <http://computerworld.com.br/gestao/2008/01/15/itil-nova-versao-traz-mais-complexidade-e-maiores-custos>. Acesso em: 05 out. 2016.

MEYER, Paul. **The Difference Between a Web Application and a Desktop Application.**, 2011. Disponível em: <http://web-desktop-application.articles.r-tt.com/>. Acesso em: 05 out. 2016.

BYCHKOV, Dmitriy. **Desktop vs. Web Applications: A Deeper Look and Comparison**., 2013. Disponível em <http://www.seguetech.com/desktop-vs-web-applications-a-deeper-look-and-comparison/> . Acesso em: 05 out. 2016.

CAMPISTA, Miguel Elias Mitre. **Linguagens de Programação**., 2016. Disponível em <<http://www.gta.ufrj.br/~miguel/docs/lingprog/qt-f6.pdf>>. Acesso em: 05 out. 2016.

[jryannel](https://github.com/jryannel), [micdoug](https://github.com/micdoug). **QmlBook: Conheça o framework Qt 5**., 2016. Disponível em <<http://qmlbook.github.io/pt/ch01/index.html>>. Acesso em 05 out. 2016.

POZZEBON, Rafaela. **O que GitHub?.,** 2015. Disponível em: <<https://www.oficinadanet.com.br/post/14791-o-que-github>>. Acesso em 05 out. 2016.

QT. **Qt Documentation**., 2016. Disponivel em <<http://doc.qt.io/qt-5/>>. Acesso em: 05 out. 2016.

MENEZES, Antonio Marcio A., 2009. **Introdução a Programação C++ com Qt 4**. Disponível em: <<http://antoniomenezes.net/down/minicursoQt4/oficinaQt4.pdf>>. Acesso em: 05 out. 2016.

MELO, ANTÔNIO Luiz De. MENDES, Luís Augusto Mattos. **Sistema Help Desk: um estudo de caso da empresa Marluvas**., 2009. Disponível em: < <http://www.unipac.br/site/bb/tcc/tcc-963884dac0af21d91a8c802bd288e0df.pdf>>. Acesso em: 05 out. 2016.

BIAZUS, Diogo de Oliveira; COUTINHO, Nabucoonosor. **Introdução e Histórico.**2003. Disponível em: <https://wiki.postgresql.org/wiki/Introdução\_e\_Histórico>. Acesso em: 05 out. 2016.

1. Defeito ou falha em um sistema computacional que é conhecido pelos desenvolvedores e/ou profissionais responsáveis. [↑](#footnote-ref-1)
2. ***First-In Last-Out*** – Priorização de Pilha, onde o primeiro item a entrar em uma pilha, é o último a sair. [↑](#footnote-ref-2)
3. ***First-Int First-Out***– Priorização de Fila, onde o primeiro item a entrar em uma fila, é o primeiro a sair. [↑](#footnote-ref-3)
4. Nome dado ao sistema apresentado e desenvolvido no presente trabalho. [↑](#footnote-ref-4)
5. Matriz RACI – Matriz que organiza as responsabilidades de cada membro da equipe em determinada tarefa [↑](#footnote-ref-5)