**RESUMO**

Em todo negócio de TI (Tecnologia da Informação) é muito importante manter um helpdesk bem informado, para evitar retrabalho com problemas recorrentes e para tornar a operação mais eficiente.

Para garantir a qualidade dos serviços foram criados modelos de melhores práticas como o ITIL (Information Techonology Infrastructure Library), que contem boas práticas que também podem tornar a vida do helpdesk mais fácil na solução dos problemas e promover a satisfação dos usuários, elevando o nível de qualidade dos serviços prestados, reduzindo custos e gerando dados concisos para as tomadas de decisãoda organização.

Nesse trabalho é feito um estudo mostrando um sistema de helpdesk alinhado com as boas práticas do ITIL sempre buscando melhoria continuada e cada vez mais valor agregado ao serviço.

Palavras-chave: ITIL, helpdesk, boas práticas

**ABSTRACT**

In IT(Information Technology) business it’s very important to keep a well informed helpdesk, to avoid rework with recurrent problems and to make the operation more effective.

To ensure the quality of service its been created models of best practices like ITIL (Information Techonology Infrastructure Library), that model have best practices that can make the life of helpesk easier in the solution of the problems and to promote the user satisfaction, elevating the quality level of services provided, lowering costs and generating concise data to make the decision-making easier for the organization.

In this work there is a study of a helpdesk system tuned with the best practices of ITIL always looking for continued improvement and more value to the service.

Keyword: ITIL, helpdesk, best practices

# LISTA DE FIGURAS

[Figura 1 - Diagrama de caso de uso 27](#_Toc468129736)

[Figura 2 - Diagrama entidade de relacionamento do banco de dados 38](#_Toc468129737)

[Figura 3- Diagrama de Classe 39](file:///C:\Users\BrunoSR\Desktop\TCC-Jonatasv2FINAL.docx%20(1).docx#_Toc468129738)

[Figura 4 - Tela de login do Kaizen 41](#_Toc468129739)

[Figura 5 - Tela principal do Kaizen 42](#_Toc468129740)

[Figura 6 - Tela de cadastro de usuário do Kaizen 42](#_Toc468129741)

[Figura 7 - Tela cadastro de status do Kaizen 43](#_Toc468129742)

[Figura 8 - Tela cadastro de prioridade do Kaizen 43](#_Toc468129743)

[Figura 9 - Tela cadastro de grupo do Kaizen 44](#_Toc468129744)

[Figura 10 - Tela cadastro da matriz RACI do Kaizen 44](#_Toc468129745)

[Figura 11 - Tela Cadastro de tipo do Kaizen 45](#_Toc468129746)

[Figura 12 - Tela cadastro de pessoa do Kaizen 45](#_Toc468129747)

[Figura 13 - Tela cadastro de ativos do Kaizen 46](#_Toc468129748)

[Figura 14 - Tela cadastro de chamado do Kaizen 46](#_Toc468129749)

[Figura 15 - Tela de buscar pessoa do Kaizen 47](#_Toc468129750)

[Figura 16 - Tela de buscar ativos do Kaizen 47](#_Toc468129751)

# LISTA DE TABELAS

[Tabela 1 - Requisitos funcionais 25](#_Toc468129752)

[Tabela 2 - Requisitos não funcionais 25](#_Toc468129753)

[Tabela 3 - Caso de uso 26](#_Toc468129754)

Sumário

[LISTA DE FIGURAS 2](#_Toc468129711)

[LISTA DE TABELAS 3](#_Toc468129712)

[1. INTRODUÇÃO 4](#_Toc468129713)

[1.1. Problematização 7](#_Toc468129714)

[1.2.1. Formulação Do Problema 7](#_Toc468129715)

[1.2.2. Solução Proposta 9](#_Toc468129716)

[1.2. Objetivos Gerais 9](#_Toc468129717)

[1.2.1. Objetivos Específicos 10](#_Toc468129718)

[1.3. Metodologia 10](#_Toc468129719)

[1.4. Estrutura do Trabalho 11](#_Toc468129720)

[2. FUNDAMENTAÇÃO TEÓRICA 13](#_Toc468129721)

[2.1. ITIL 13](#_Toc468129722)

[2.1.1 Operação de Serviço 13](#_Toc468129723)

[2.2. GitHub 14](#_Toc468129724)

[2.3. Qt e Qt Creator 15](#_Toc468129725)

[2.4. C++ 16](#_Toc468129726)

[2.5. PostgreSQL 17](#_Toc468129727)

[2.6. Comparação entre Web e Desktop 18](#_Toc468129728)

[3. DESENVOLVIMENTO 21](#_Toc468129729)

[3.1.3 Projeto KAIZEN 22](#_Toc468129730)

[3.1.3 Análise de Requisitos 24](#_Toc468129731)

[3.1.3 Modelagem UML 25](#_Toc468129732)

[3.1.3 Implementação 40](#_Toc468129733)

[4. CONCLUSÃO 48](#_Toc468129734)

[BIBLIOGRAFIA 49](#_Toc468129735)

# **INTRODUÇÃO**

Segundo CUNHA, O conceito de valor agregado não é novo, mas sua utilidade é crescente, vital, para a organização das empresas e para a disputa competitiva. Para auxiliar as organizações a alcançar maior valor agregado nos serviços de TI 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**.**

Segundo SORTICA (2004) o “princípio fundamental do ITIL é o fornecimento de qualidade de serviço aos clientes de TI com custos justificáveis, isto é, relacionar os custos dos serviços de tecnologia e como estes trazem valor estratégico ao negócio.

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 mantém 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 um 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* [[2]](#footnote-2)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.

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 *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 deste departamento vai muito além de instalar drivers, trocar senhas e outros equipamentos.

No *helpdesk* 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 *helpdesk* 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 *helpdesk* e, por este motivo, o departamento merece atenção especial. ”

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.

## **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, demoram a entender o funcionamento, entre outros.

Com a falta de organização, os técnicos dos *helpdesk* 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 em *FILO*[[3]](#footnote-3), ou seja, um chamado chega, os outros são abandonados e o chamado mais recente é resolvido, ou no modo *FIFO*[[4]](#footnote-4), 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.

Outro ponto fraco de uma grande parte das empresas está em identificar gargalos na infraestrutura e nos sistemas, não há relatórios e nem dados para se realizar uma análise mais profunda do porquê determinada área está tendo tantos problemas.

Tal análise pode iniciar tanto por qual área está abrindo mais chamados como qual área está fechando menos chamados, o problema é realizar esse estudo estatístico sem dados de fácil acesso.

Se em determinada área temos sempre chamados relacionados à impressora, podemos entender que a impressora é um gargalo[[5]](#footnote-5) e deve ser substituída vista 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.

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[[6]](#footnote-6) 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 *helpdesk* 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 se imagina 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[[7]](#footnote-7)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*.

## **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, o projeto sobre o sistema Kaizen que auxilia o helpdesk que atua na área de gerenciamento de problema, mostrando análise de requisitos que descreve sobre a lista de requisitos funcionais e não funcionais e especificações de requisitos funcionais e não funcionais, a modelagem UML mostra a lista de caso de uso, diagrama de caso de uso, especificação do caso de uso e o diagrama de classe mostrando a funcionalidade do sistema Kaizen.

No capítulo implementação fala sobre toda experiência do desenvolvimento e de como o sistema funciona, como usar o sistema, tendo prints das telas do sistema, e mostrando a execução do sistema. 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. (Fonte: HARTMAN, Samuel e WOBETO, Edson)**.**

Para GASPAR; GOMEZ; MIRANDA (2010), os negócios têm grande dependência da TI, fazendo com que as boas práticas sejam adotadas com objetivo de trazer resultados positivos, como redução de custos e agilidade em seus processos, visto que estes independem de fornecedores e tecnologias.

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.

**2.1.1 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. “Gerência 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. (FONTE: POZZEBON, Rafaela)

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 está 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**

Segundo Barbosa (2007), seguem algumas características de aplicações web e aplicações desktop:

Algumas vantagens das aplicações Web são:

* Interface HTML reconhecida por uma grande gama de usuários já acostumados com o funcionamento dos navegadores.
* Desenvolvimento, manutenção e atualização centralizada da aplicação. Não é necessário instalar a aplicação em diversos equipamentos diferentes. Basta colocá-lo no servidor para que os usuários a acessem.
* A exportação de dados entre usuários remotos usando o protocolo HTTP é muito mais fácil do que em aplicações desktop.
* Escalabilidade no processamento. Se houver necessidade de aumentar o poder de processamento, basta fazer isto no servidor.

Desvantagens de aplicações Web:

* A interface HTML pode ser um problema, pois não há uma padronização entre os diversos navegadores e sua aplicação poderia ser exibida de uma maneira diferente dependendo do navegador.
* A entrada de uma grande massa de dados é prejudicada na interface HTML, pois não existe uma maneira padrão de criar máscaras de entrada de dados.
* A interface HTML não é rica em controles gráficos e peca no quesito posicionamento. O visual da aplicação pode não ficar tão elegante como você imagina.
* A integração com outros componentes não é tão fácil com HTML.

Algumas vantagens encontradas nas aplicações Desktop:

* Uma rica variedade de controles para interface com o usuário.
* Um total controle sobre o posicionamento dos controles na aplicação.
* O desempenho da interface gráfica é melhor em uma aplicação desktop, já que usa processamento local.
* A interface tem suporte a vários dispositivos de entrada.
* Desvantagens das aplicações Desktop:
* Uma interface gráfica muito carregada deixa a aplicação mais pesada.
* A integração com usuários remotos é mais difícil.
* A distribuição da aplicação é crítica. A aplicação tem de ser instalada na máquina de todos os usuários, que podem ter diferentes tipos de sistemas operacionais.
* A manutenção e atualização da aplicação requerem um esforço extra.

Tendo em vista tudo que foi levantado por Barbosa podemos destacar das aplicações desktop a interface mais rápida, mais rica tanto na questão visual quanto no posicionamento dos controles.

No caso das desvantagens podemos ainda aproveitar algumas delas, por exemplo, na atualização, como não teremos que parar todas as estações para atualizar o servidor, as estações serão de forma controlada atualizadas uma a uma sem grande impacto na operação do sistema.

Com certeza existem dois desafios facilmente superados nas aplicações web que no desktop ainda são muito difíceis, integração de dados entre usuários remotos e escalabilidade no processamento.

Mesmo com esses dois grandes desafios ainda sim, a agilidade na resposta da interface, sua riqueza e a diminuição do impacto nas janelas de manutenção do sistema para a operação pesam a favor das aplicações desktop.

# **DESENVOLVIMENTO**

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[[8]](#footnote-8) 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.

1. **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 |

Tabela 1 - Requisitos funcionais

* **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 |

Tabela 2 - Requisitos não funcionais

1. **Modelagem UML**

* **Lista de Casos de Uso**

|  |  |  |  |
| --- | --- | --- | --- |
| **Número** | **Nome** | **Ator** | **Descrição** |
| UC01 | Login | Todos os Usuário | Caso de Uso responsável por logar o usuário no sistema. |
| UC02 | Criar Usuário | Usuário Administrativo | Caso de Uso responsável por criar usuário para utilizar o sistema. |
| UC03 | Criar Status | Usuário Administrativo | Caso de Uso responsável por criar status do chamado para o sistema. |
| UC04 | Criar Prioridade | Usuário Administrativo | Caso de Uso responsável por criar as prioridade dos chamados do sistema. |
| UC05 | Criar Pessoa | Usuário Técnico de TI | Caso de Uso responsável por manter o cadastro das pessoas no sistema. |
| UC06 | Criar  Ativos | Usuário Técnico de TI | Caso de Uso responsável por manter o cadastro de ativos no sistema. |
| UC07 | Criar  Chamado | Usuário Técnico de TI | Caso de Uso responsável por manter o cadastro dos chamados no sistema. |
| UC08 | Buscar Pessoa | Usuário Técnico de TI | Caso de Uso responsável por fazer a busca de pessoas cadastradas no sistema. |
| UC09 | Buscar Ativos | Usuário Técnico de TI | Caso de Uso responsável por fazer a busca de ativos cadastrado no sistema. |
| UC10 | Buscar Chamados | Usuário Técnico de TI | Caso de Uso responsável por fazer a busca de chamados cadastrado no sistema. |
| UC011 | Relatório Simples Pessoa | Usuário Técnico de TI | Caso de Uso responsável por mostrar relatório de pessoa |
| UC12 | Relatório Simples  Chamado | Usuário Técnico de TI | Caso de Uso responsável por mostrar todos os chamados cadastrados no sistema. |
| UC13 | Relatório Simples Ativos | Usuário Técnico de TI | Caso de Uso responsável por mostrar todos os ativos cadastrados no sistema |

Tabela 3 - Caso de uso

* **Diagrama de Casos de Uso**

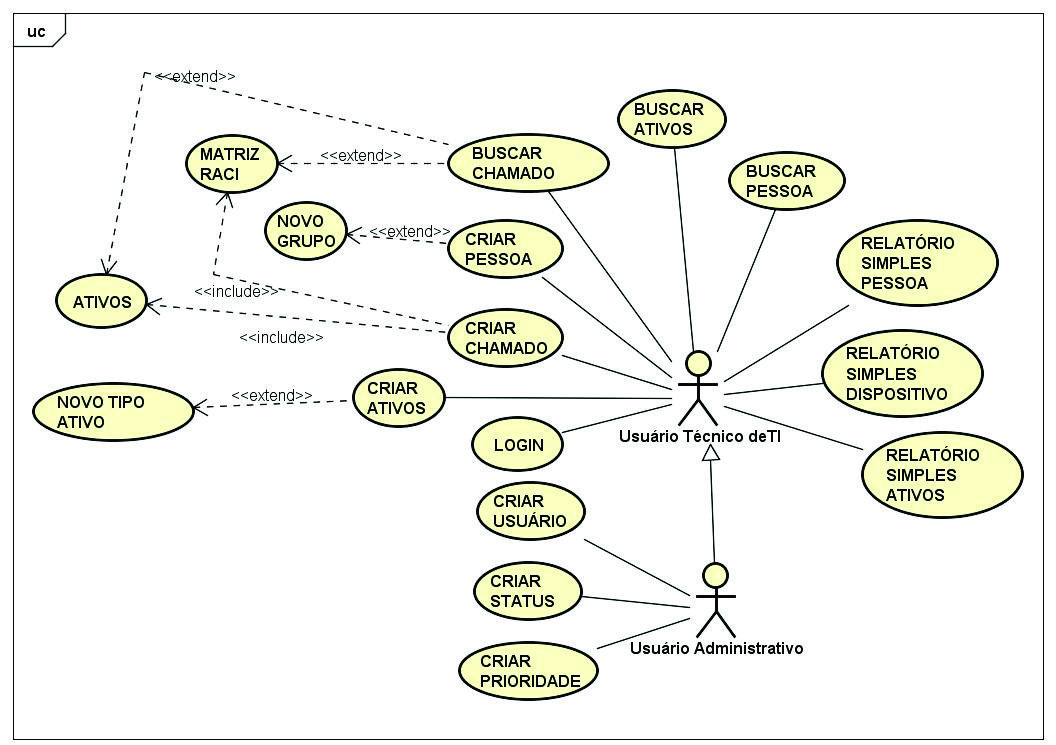


Figura 1 - Diagrama de caso de uso

* **Especificação de Caso De Uso**

**UC01 – Login**

**NOME:** Login

**DESCRIÇÃO SUCINTA:** os usuários fazem login com o sistema.

**ATORES:**

1.      Usuários Técnico de TI e Usuário Administrativo

**FLUXO BÁSICO**

1.      O usuário abrir o sistema e automaticamente aparece a tela de login

2.      O usuário digita seu login e senha

3.      O usuário clica em entrar

4.      O Sistema mostrar as opções

8.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 2 – Login**

1.a. O sistema abre a tela de login o usuário tem a opção CANCELAR, para fechar o sistema.

**(A3) Alternativa ao Passo 3 – A regra RN1 não é atendida**

3.a.  O sistema retorna uma mensagem de erro “"Erro de usuário e/ou senha" o usuário clica em ok e será redirecionado para a mesma tela.

FIM DO CASO DE USO LOGIN.

**UC02 – Criar Usuário**

**NOME:** Criar Usuário

**DESCRIÇÃO SUCINTA:** Usuário Administrativo cadastrar usuário no sistema.

**ATORES:**

1.      Usuário Administrativo

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Administrativo”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Criar Usuário”

4.      O Sistema abre a tela de cadastramento

5.      O usuário preenche o campo “USUÁRIO E SENHA”

6.      O usuário preenchendo corretamente os campos tem o botão “Criar Usuário” e ‘‘ Alterar Senha’’ que é obrigatório a salvar

8.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Criar Usuário**

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

**REGRAS DE NEGÓCIO**

**(RN1)** Preenchimento dos campos obrigatórios

FIM DO CASO DE USO USUÁRIO

**UC03 – Criar Status**

**NOME:** Criar Status

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

**ATORES:**

1.      Usuário Administrativo

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Administrativo”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Status”

4.      O Sistema abre a tela de cadastramento

5.      O usuário preenche o campo “Status [Adicionando Status]”

6.     O usuário fecha a tela no “X” do canto direito

7.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Criar Status**

1.a. O sistema abre a tela de cadastramento de status onde tem a opção alternativa para o usuário SUBSTITUIR STATUS SELECIONADO, o sistema vai substituir pelo status atual.

FIM DO CASO DE USO STATUS.

**UC04 – Criar Prioridade**

**NOME:** Criar Prioridade

**DESCRIÇÃO SUCINTA:** Usuário cadastra a prioridade do chamado no sistema.

**ATORES:**

1.      Usuário Administrativo

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Criar”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Prioridade”

4.      O Sistema abre a tela de cadastramento

5.      O usuário preenche o campo “Prioridade [Adicionar Prioridade]”

6.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Criar Prioridade**

1.a. O sistema abre a tela de cadastramento de prioridade onde tem a opção alternativa para o usuário SUBSTITUIR PRIORIDADE SELECIONADA, o sistema vai substituir a prioridade pela atual.

FIM DO CASO DE USO PRIORIDADE

**UC05 – Criar Pessoa**

**NOME:** Criar Pessoa

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

**ATORES:**

1.      Usuário Técnico de TI

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Criar”

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,E-MAIL,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 – Criar 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 GRUPO, quando o usuário clicar em novo o sistema abre a janela de cadastramento de grupo (NOME DO GRUPO [ADICIONAR GRUPO] tendo a opção de (SUBSTITUIR GRUPO SELECIONADO)

**(A2) Alternativa ao Passo 6 – 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 CRIAR PESSOA.

**UC06 – Criar Ativos**

**NOME:** Criar Ativos

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

**ATORES:**

1.      Usuário Técnico de TI

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Criar”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Ativos”

4.      O Sistema abre a tela de cadastramento

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

6. O usuário tem a opção de criar novo tipo

7.      os campos são obrigatórios

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

9.      E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 6 – Criar Ativos**

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

**(A2) Alternativa ao Passo 4 – Criar Ativos**

1.a. O sistema abre a tela de cadastramento de ativos onde terá a opção alternativa para o usuário ADICIONAR NOVO TIPO, o sistema abre uma janela onde o usuário cadastra o novo tipo, é salvo automaticamente, o usuário fecha a janela no “X”.

FIM DO CASO DE USO ATIVOS..

**UC07 – Criar Chamado**

**NOME:** Criar chamado

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

**ATORES:**

1.      Usuário Técnico de TI

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Criar’’

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, ABRIR ATIVOS,TEMPO DECORRIDO, SLA, STATUS (ABERTO, ATRIBUÍDO, EM EXECUÇÃO E CONCLUÍDO), REINCIDENTES, ABERTO EM, CONCLUÍDO EM E DESCRIÇÃO”

6.      O usuário abre a tela da Matriz RACI

7.      O usuário abre a tela de Ativos

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 – Criar 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.

**REGRAS DE NEGÓCIO**

**(RN1)** Preenchimento dos campos obrigatórios

FIM DO CASO DE USO CHAMADO

**UC08 – Busca Pessoa**

**NOME:** Buscar Pessoa

**DESCRIÇÃO SUCINTA:**Usuário faz busca de pessoas no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.      As pessoas precisam estar cadastradas

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Buscar’’

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Pessoa”

4.      O Sistema abre a tela de busca

5.      O usuário preenche o campo “ID’’

6.      O sistema mostra o dados (NOME,TELEFONE RAMAL,E-MAIL,DEPTO,CARGO,GRUPO)

7.      O usuário tem opção de editar

8.  E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

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

1.a. O sistema abre a tela de busca de pessoa onde terá a opção alternativa para o usuário EDITAR, o sistema libera os campos para edição

1.b.  O sistema libera para o usuário SALVAR, onde as alterações vai ser atualizadas

**(A2) Alternativa ao Passo 5 – Informações Incorretas**

2.a. O sistema retorna uma mensagem de erro “usuário não encontrado” o usuário clica em ok e será redirecionado para a tela anterior.

FIM DO CASO DE USO BUSCA PESSOA

**UC09 – Busca Ativos**

**NOME:** Busca Ativos

**DESCRIÇÃO SUCINTA:**Usuário faz busca dos ativos cadastrados no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.      Os ativos precisam estar cadastradas

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Buscar’’

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Ativos”

4.      O Sistema abre a tela de busca

5.      O usuário preenche o campo “ID’’

6.      O sistema mostra o dados (NOME,TIPO, LOCAL)

7.      O usuário tem opção de editar

8.  E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 4 – Busca Ativos**

1.a. O sistema abre a tela de busca de ativos onde terá a opção alternativa para o usuário EDITAR, o sistema libera os campos para edição

1.b.  O sistema libera para o usuário SALVAR, onde as alterações vai ser atualizadas

**(A2) Alternativa ao Passo 5 – Informações Incorretas**

2.a. O sistema retorna uma mensagem de erro “Ativos’’não encontrado” o usuário clica em ok e será redirecionado para a tela anterior.

FIM DO CASO DE USO BUSCA ATIVOS

**UC10 – Busca Chamado**

**NOME:** Buscar Chamado

**DESCRIÇÃO SUCINTA:**Usuário faz busca de chamado no sistema.

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.      Os Chamados precisam estar cadastradas

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Buscar’’

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Chamado”

4.      O Sistema abre a tela de busca

5.      O usuário preenche o campo “ID’’

6.      O sistema mostra o dados (‘‘ID, MATRIZ RACI, ABRIR ATIVOS,TEMPO DECORRIDO, SLA, STATUS [ABERTO, ATRIBUÍDO, EM EXECUÇÃO E CONCLUÍDO], REINCIDENTES, ABERTO EM, CONCLUÍDO EM E DESCRIÇÃO’’)

7.      O usuário tem opção de editar “STATUS, ACOMPANHAMENTO”

8.   E o caso de uso é encerrado.

**FLUXOS ALTERNATIVOS**

**(A1) Alternativa ao Passo 7 – Busca Chamado**

1.a. O sistema abre a tela de busca de chamado onde terá a opção alternativa para o usuário ADICIONAR ACOMPANHAMENTO

FIM DO CASO DE USO BUSCA PESSOA

34e1

**UC11 – Relatório Simples Pessoa**

**NOME:** Relatório simples pessoa

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

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.       Que os dados estejam cadastrados.

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Relatórios”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “Pessoa’’

4.      O Sistema abre a tela com a lista de pessoas cadastradas

5.      E o caso de uso é encerrado.

FIM DO CASO DE USO RELATÓRIO SIMPLES PESSOA

**UC12 – Relatório Simples Chamados**

**NOME:** Relatório simples chamados

**DESCRIÇÃO SUCINTA:**O sistema lista para o usuário um relatório de todos os chamados cadastrados

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.       Que os dados estejam cadastrados.

**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.

FIM DO CASO DE USO RELATÓRIO SIMPLES CHAMADOS

**UC13 – Relatório Simples Ativos**

**NOME:** Relatório simples ativos

**DESCRIÇÃO SUCINTA:**O sistema lista para o usuário um relatório de todos os ativos(tudo que vêm da problema) cadastrados

**ATORES:**

1.      Usuário Técnico de TI

**PRÉ-CONDIÇÕES:**

1.       Que os dados estejam cadastrados.

**FLUXO BÁSICO**

1.      O usuário seleciona a opção “Relatórios”

2.      O Sistema exibe as opções

3.      O usuário escolhe a opção “ATIVOS’’

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

5.      E o caso de uso é encerrado.

FIM DO CASO DE USO RELATÓRIO SIMPLES ATIVOS

* **Diagrama de entidade de relacionamento de banco de dados**

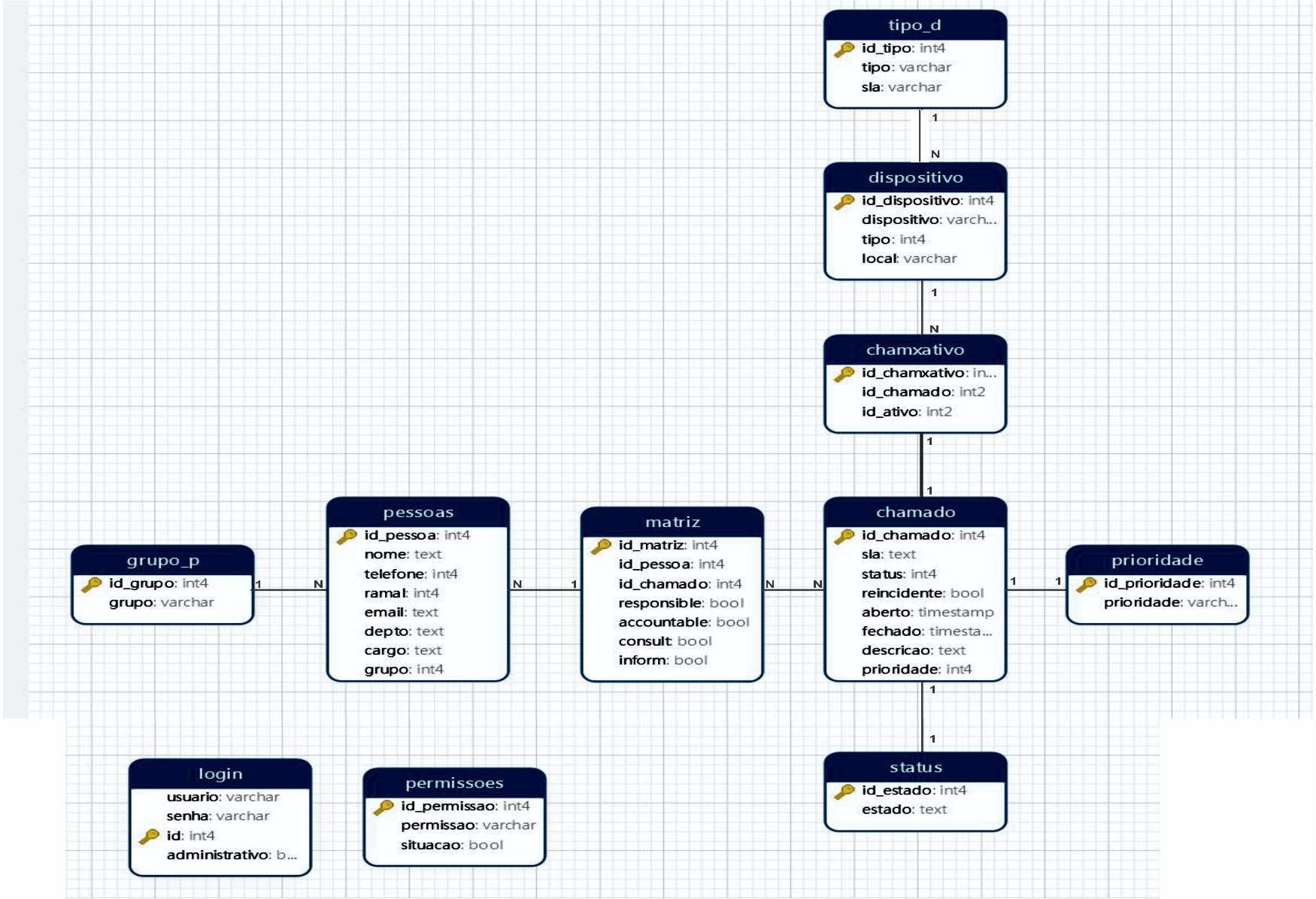
****

Figura 2 - Diagrama entidade de relacionamento do banco de dados

* **Diagrama de Classe**

Figura 3- Diagrama de Classe

1. **Implementação**

Na implementação do sistema foi escolhida a linguagem C++ com a framework Qt, inicialmente por questões práticas, a equipe tem mais experiência com o C/C++ e o Qt é um framework que veio para facilitar a portabilidade do sistema para vários sistemas operacionais, como visto anteriormente.

Durante o desenvolvimento outras vantagens apareceram, como o baixo consumo de memória, vasta documentação online tanto do Qt quando do C/C++ e o fato do Qt ter reimplementado algumas funções da linguagem C/C++ exatamente para facilitar sua integração com a parte visual feita em Qt.

O banco de dados PostgreSQL, escolhido para o sistema se mostrou inicialmente uma decisão que poderia ter complicado o trabalho, apesar de ser um SGBD[[9]](#footnote-9) (Sistema Gerenciador de Banco de Dados) muito robusto e eficiente, ter uma licença livre de custos muito interessante para os futuros clientes do sistema, algumas desvantagens surgiram.

A principal delas é que o driver de conexão do Qt para esse banco necessita ser compilado pois o que acompanha a framework mostra algumas limitações.

O processo de compilação desse driver não é tão trivial quanto parece, muitos arquivos devem ser colocados na mesma pasta, é necessário a instalação do PostgreSQL na máquina de desenvolvimento e usando o MinGW (série de ferramentas de desenvolvimento opensource) é feita a compilação utilizando alguns arquivos do banco de dados e outros do *framework* Qt.

Superado esse desafio o desenvolvimento fluiu muito mais rápido, com a utilização do Qt as telas foram desenvolvidas de forma muito rápida e prática, porém, problemas foram surgindo conforme novas camadas de janelas foram sendo adicionadas.

O código se torna muito complexo quando você tem uma janela inicial e três ou mais níveis de janelas após alguns cliques de botão, várias linhas complicadas foram surgindo o que provocou uma mudança no design da aplicação no meio do desenvolvimento.

Mais menus e menos camadas, além de simplificar o uso do sistema vai facilitar muito a manutenção do mesmo, já que é muito mais fácil e prático acrescentar um menu com uma janela, do que uma série de janelas que começam de um botão em uma janela que vem de outra janela.

Após todas as janelas redesenhadas no novo padrão era necessário dar funcionalidade a essas janelas, fazendo o acesso ao banco de dados, extraindo informações de lá, incluído e atualizando o banco de dados de problemas.

Nesse momento surgiram algumas dúvidas com relação ao funcionamento do banco de dados escolhido, nos outros SGBDs normalmente o índice de cada tabela já é auto incremental, no PostgreSQL esse incremento deve ser especificado, dando a nós a experiência de mais uma característica interessante do SGBD escolhido, os índices podem ter incrementos de acordo com o gosto do usuário de maneira bem prática.

Todos os cadastros e consultas começaram a se integrar e só ficaram os relatórios por último.

* EXECUÇÃO DO SISTEMA

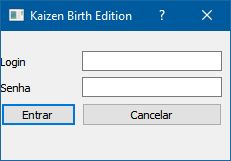


Figura 4 - Tela de login do Kaizen

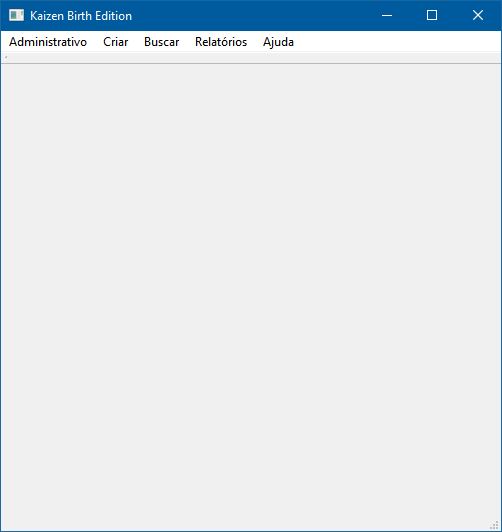


Figura 5 - Tela principal do Kaizen

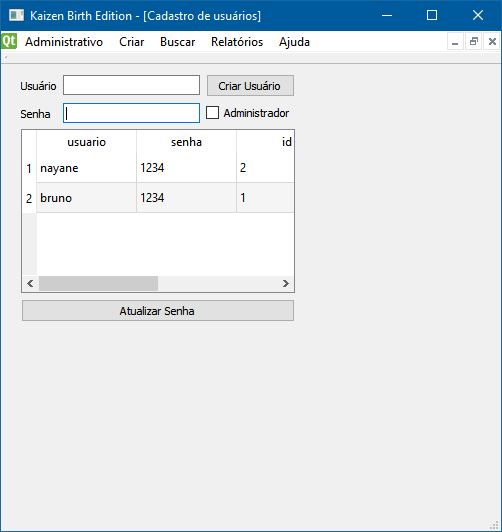
**

Figura 6 - Tela de cadastro de usuário do Kaizen

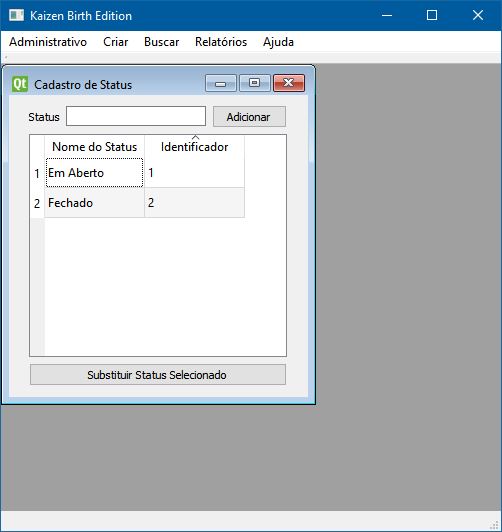


Figura 7 - Tela cadastro de status do Kaizen

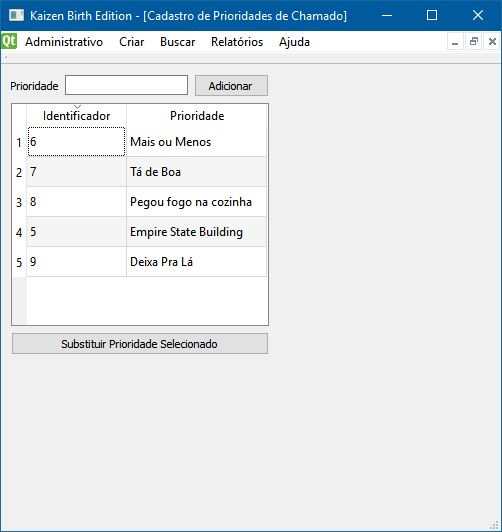


Figura 8 - Tela cadastro de prioridade do Kaizen

**

Figura 9 - Tela cadastro de grupo do Kaizen

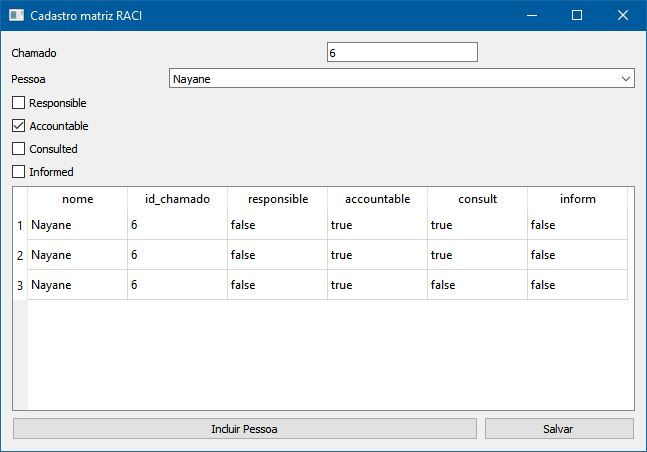
**

Figura 10 - Tela cadastro da matriz RACI do Kaizen

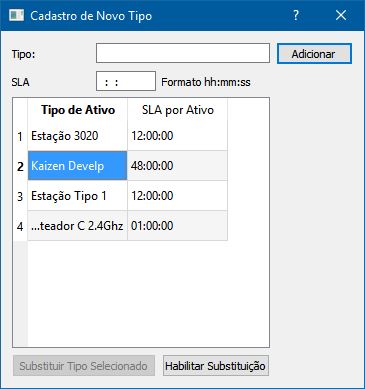
**

Figura 11 - Tela Cadastro de tipo do Kaizen

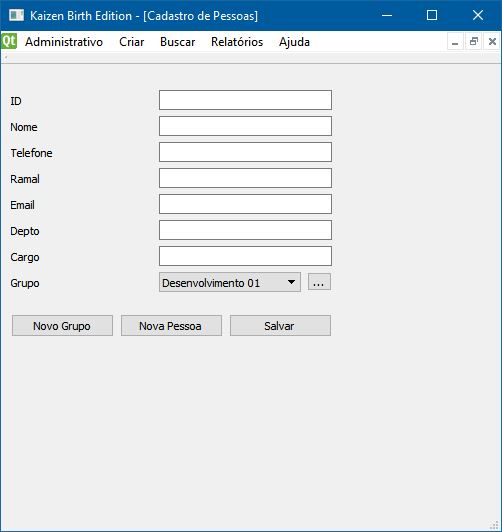


Figura 12 - Tela cadastro de pessoa do Kaizen

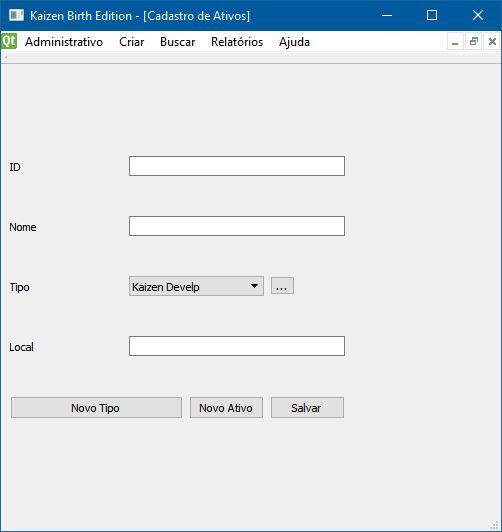


Figura 13 - Tela cadastro de ativos do Kaizen

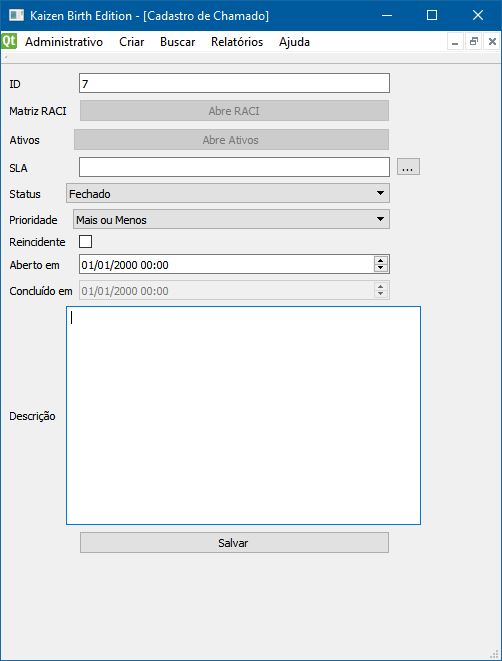


Figura 14 - Tela cadastro de chamado do Kaizen

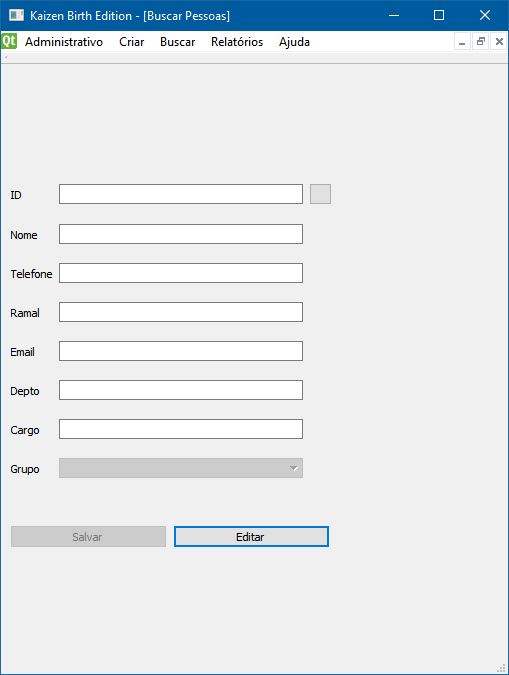


Figura 15 - Tela de buscar pessoa do Kaizen

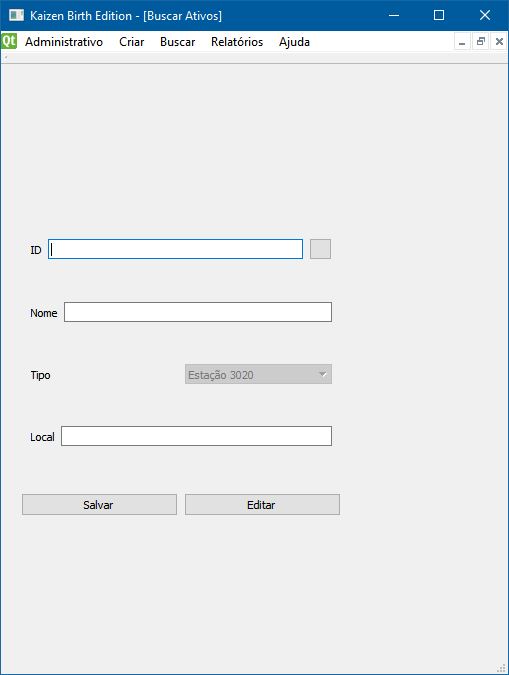
**

Figura 16 - Tela de buscar ativos do Kaizen

# **CONCLUSÃO**

No fim do trabalho espera-se entregar um sistema de *helpdesk* 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.

Em uma segunda fase do desenvolvimento do Kaizen há planos que ele tenha uma busca heurística pelos chamados abertos e fechados para que o usuário tenha uma forma mais eficiente de evitar a abertura de chamados duplicados e a disseminação de conhecimento. E o usuário vai ter a escolha de usar temas para customizar com cores e colocar fundo nas janelas do programa deixando assim o sistema com a personalização da empresa, e também será liberado a busca por outro campo exemplo: Nome facilitando o usuário na hora que for fazer a busca especifica, por enquanto só está liberado para fazer a busca por ID.

Além disso não vai ter limitações para relacionar um chamado a outro por enquanto cada chamado só suporta um único ativo, ainda não é possível mais de um ativo por chamado, caso o mesmo problema afete mais de um ativo é necessário abrir mais de um chamado, mesmo com essas limitações e algumas funções ainda a desenvolver nosso trabalho demonstrou que, de forma simples, é possível trabalhar o helpdesk em sintonia com as boas práticas do ITIL.

Para isso será realizado uma série de estudos dentro da inteligência artificial para, com aprendizado de máquina, possamos “ensinar” o Kaizen a evitar situações indesejadas pelos usuários.

Além disso será implementado um sistema de temas para que cada empresa possa utilizar o Kaizen com sua identidade visual, agregando valor ao sistema.

# **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.

GASPAR, Marcelo; GOMEZ, Thierry; MIRANDA Zailton. **T.I. Mudar e Inovar Resolvendo conflitos com ITIL® V3**. Brasília: Senac, 2010.

SORTICA, Eduardo A.; CLEMENTI, Sérgio; CARVALHO, TCMB. Governança de TI: Comparativo entre COBIT e ITIL. In: Anais do Congresso Anual de Tecnologia da Informação-CATI. 2004. Disponível em: http://187.7.106.14/andre/redes/topicos-esp/2012-2/roteiros/roteiro3-ITIL/docs-ITIL/Icaro-Governanca-de-TI-Comparativo-entre-Cobit-e-Itil.pdf. Acesso em: 21 nov. 2016

Korth, H. F., Silberschatz, Sistema de Banco de Dados, 3ª edição, Makron Books, 1999

HARTMANN,Samuel : WOBETO, Edson. Melhores práticas em TI baseado em ITIL. São Paulo, Abanet 23/06/2015 Disponível em: http://www.abranet.org.br/Artigos/Melhores-praticas-em-TI-baseado-em-ITIL-para-provedores-de-acesso-a-Internet-no-Vale-dos-Sinos-682.html#.WDrd9PkrLIU.

1. Defeito ou falha em um sistema computacional que é conhecido pelos desenvolvedores e/ou profissionais responsáveis. [↑](#footnote-ref-1)
2. Helpdesk – “Balcão de ajuda” em português, termo adotado da língua inglesa que segundo o Dicionário Michaelis significa: “ Serviço de apoio aos usuários para solucionar problemas técnicos.” [↑](#footnote-ref-2)
3. ***First-In Last-Out*** – Priorização de Pilha, onde o primeiro item a entrar em uma pilha, é o último a sair. [↑](#footnote-ref-3)
4. ***First-In First-Out***– Priorização de Fila, onde o primeiro item a entrar em uma fila, é o primeiro a sair. [↑](#footnote-ref-4)
5. ***Gargalo***– Perde de desempenho. [↑](#footnote-ref-5)
6. Nome dado ao sistema apresentado e desenvolvido no presente trabalho. [↑](#footnote-ref-6)
7. Matriz RACI – Matriz que organiza as responsabilidades de cada membro da equipe em determinada tarefa [↑](#footnote-ref-7)
8. Service Level Agreement – Um acordo de nível de serviço, um contrato entre fornecedor de TI e um cliente especifico. [↑](#footnote-ref-8)
9. SGBD – Sistema Gerenciador de Banco de Dados – *“um Sistema Gerenciador de Bancos de Dados é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados* (SILBERSCHATZ, 1999) [↑](#footnote-ref-9)