

UNIVERSIDADE ESTADUAL DE GOIÁS
CAMPUS ITABERAÍ
CURSO DE SISTEMAS DE INFORMAÇÃO

LUIZ PAULO DOS SANTOS COSTA

SGP
SISTEMA DE GESTAO PARTICIPATIVA

ITABERAÍ - GO
2017

LUIZ PAULO DOS SANTOS COSTA

SGP
SISTEMA DE GESTAO PARTICIPATIVA

Projeto final, apresentado a Universidade Estadual de Goiás, como parte das exigências para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Juliana Vasconcelos Braga

ITABERAÍ - GO

2017

DEDICATÓRIA

Dedico este trabalho primeiramente a Deus, que até aqui tem me ajudado me concedendo força e coragem durante essa caminhada, à minha família que sempre me apoiou em minhas decisões, aos professores que compartilharam do seu conhecimento comigo e aos meus amigos.

RESUMO

Este projeto apresenta a modelagem de um projeto, SGP (Sistema de Gestão Participativa). Foi realizada uma pesquisa com a população de Itaberaí, através da mesma se obteve um conhecimento mais profundo da importância e necessidade desse projeto. Foram utilizadas técnicas e padrões de implementação de software, que são elas, engenharia de software, engenharia de requisitos, banco de dados, análise orientada a objeto, UML. Colocando em prática as teorias estudadas nas disciplinas presentes no Curso de Sistemas de Informação da Universidade Estadual de Goiás, Campus Itaberaí.

A pesquisa foi de extrema importância para esse projeto, ela esclareceu a importância desse projeto para a população e para a gestão. O uso da UML (Unified Modeling Language), foi muito importante para demonstrar o funcionamento do sistema.

Palavras-chave: Engenharia de Requisitos, Modelagem de dados, Projeto, Java ee.

LISTA DE FIGURAS

Figura 1 – Modelo De Entidade Relacionamento	32
Figura 2 – Modelo Relacionamento Normalizado	33
Figura 3 – Diagrama de Classes SGP	34
Figura 4 – Diagrama de Caso de Uso SGP	35
Figura 5 – Diagrama de Sequência – Cadastrar usuário	36
Figura 6 – Diagrama de Sequência – Acessar Sistema	37
Figura 7 – Diagrama de Sequência – Registrar solicitação	38
Figura 8 – Diagrama de Sequência – Cadastrar categoria	39
Figura 9 – Diagrama de Sequência – Registrar solução	40
Figura 10 – Tela de Login	41
Figura 11 – Tela de cadastro de usuário	42
Figura 12 – Tela de tela de início	43
Figura 13 – Tela de registrar solicitação	44
Figura 14 – Tela de cadastrar categoria	45
Figura 15 – Tela de registrar solução	46
Figura 16 – Tela de Acompanhar solicitação	47
Figura 17 – cronograma	48
Figura 18– lista de requisitos	48
Figura 19 – Entrevista parte 1	49
Figura 20 – Entrevista parte 2	50

SUMÁRIO

1. INTRODUÇÃO	7
1.1. Contextualização	7
1.2. Objetivo geral	8
1.3. Objetivo específico	8
1.4. Justificativa	8
1.5. Metodologia	8
1.6 Estudo de caso	9
1.6.1 Entrevista	10
2. REFERENCIAL TEÓRICO	11
2.1. Engenharia de Software	11
2.1.1. Engenharia de Requisitos	18
2.1.1.1. Requisitos Funcionais	21
2.1.1.2. Requisitos Não-Funcionais	21
2.2. Banco de Dados	22
2.3. Análise Orientada a Objeto	24
2.4. UML	25
2.4.1. Diagramas Caso de Uso	26
2.4.2. Diagramas de Atividade	27
2.4.3. Diagramas de Classe	27
2.4.4. Diagramas de Sequência	28
2.5. Java EE	29
2.6. DESENVOLVIMENTO	30
2.6.1. MODELO ENTIDADE RELACIONAMENTO	30
2.6.2. MODELO RELACIONAL NORMALIZADO	31
2.6.3. DIAGRAMA DE CLASSES	32
2.6.4. DIAGRAMA DE CASO DE USO	33
2.6.5. ESPECIFICAÇÃO DE CASO DE USO	34
2.6.6. DIAGRAMAS DE SEQUÊNCIA	30
2.6.7. TELAS DO PROTÓTIPO	45
2.6.8. CRONOGRAMA E LISTA DE REQUISITOS	52
2.7. CONSIDERAÇÕES FINAIS	53
2.8. REFERÊNCIAS BIBLIOGRÁFICAS	54
2.9. ANEXOS	55

1. INTRODUÇÃO

O presente trabalho se propõe a desenvolver uma solução para promover a comunicação entre a população e a prefeitura municipal de Itaberaí, permitindo ao cidadão relatar um problema relacionado às áreas de interesse, tais como: buracos nas ruas, sinalização de trânsito, ocupação irregular, iluminação pública, limpeza e conservação, segurança, saúde, transporte público, etc.

Dessa forma com base em dados levantados através de uma pesquisa realizada juntamente com a população de Itaberaí, através da mesma notou-se dificuldade da população para solicitar essa manutenção. Com isso, pretende-se despertar a participação popular nas questões relacionadas à cidade e garantir mais qualidade na prestação de serviços ao cidadão por parte da prefeitura.

1.1. CONTEXTUALIZAÇÃO

Com base em entrevistas realizadas com moradores da cidade de Itaberaí, notou-se em grande parte das respostas que a população tem dificuldades em estabelecer um canal de comunicação com a prefeitura que seja efetivo, de forma que o desenvolvimento de uma ferramenta que facilite essa comunicação vem de encontro com a necessidade levantada e permite que a prefeitura trate dos problemas de forma mais organizada e planejada.

Nos dias de hoje, a chamada Gestão Participativa pode ser alcançada com ferramentas que permitam ao cidadão participar ativamente da gestão, apontando os problemas e acompanhando a sua solução.

Diante dos problemas observados é possível verificar as dificuldades por parte da prefeitura em realizar determinadas obras, normalmente existem pessoas que observam as necessidades da cidade, porém com a participação da população esse processo ficaria mais preciso, com isso, a gestão além de economizar com fiscais estaria mais aprofundada nos assuntos e necessidades da cidade e da população

1.2. OBJETIVO GERAL

Desenvolver a modelagem e análise de um projeto de software que sirva de base para o desenvolvimento de uma aplicação, para que a população possa solicitar manutenções na cidade de Itaberaí, manutenções essas realizadas pela prefeitura da cidade.

1.3. OBJETIVOS ESPECÍFICOS

- Analisar o contexto do trabalho, estudando as áreas de gerenciamento de projetos, softwares para gerência de projetos e dispositivos móveis.
- Propor uma solução através de um protótipo, para um possível desenvolvimento de um aplicativo no futuro.
- Permitir uma comunicação entre a população e a gestão.

1.4. JUSTIFICATIVA

Esse projeto está sendo elaborado para beneficiar a gestão e a população, o mesmo facilitará o dia-a-dia de ambas partes. Além de atender as necessidades da população, a gestão estará economizando com funcionários que solicita manutenções, sendo que a população irá realizar esse serviço sem gerar custo para a prefeitura.

A uma necessidade de ambas partes desse projeto, juntas trabalhando no mesmo objetivo alcançaram as metas de forma mais rápida e eficiente, gerando mais conforto, segurança. Algo fácil e prático sem muita burocracia, o usuário pode acessar esse sistema para solicitar a manutenção desejada com ferramentas que utilizam normalmente, um celular ou computador.

1.5.METODOLOGIA

Para o desenvolvimento do projeto foram realizadas leituras de livros, artigos em sites e revistas como referências bibliográficas, também foi feita uma pesquisa de campo com a população da cidade de Itaberaí, foram gerados gráficos, que permitem visualizar a informação de forma mais fácil. Foram utilizadas ferramentas

de desenvolvimento de Software, UML (Linguagem de Modelagem Unificada), Engenharia de Software.

Considerando, por fim os desafios da gestão pública participativa e as potencialidades que as tecnologias da informação oferecem e confrontando-as com opiniões de pessoas de nosso município, fica claro a viabilidade de um projeto que permita o desenvolvimento da cidade e a adequação dos espaços públicos pelos cidadãos ampliando a protagonismo social e ofertando novas possibilidades para a gestão pública municipal.

1.6. ESTUDO DE CASO

A prefeitura municipal de Itaberaí, foi escolhida para a realização da pesquisa de campo. Após pesquisas e leituras em site e revistas, nota-se o qual complexo pode se tornar a participação da população municipal, sem o auxílio de uma ferramenta adequada. Pois sem nenhum sistema de comunicação torna mais dificultoso, essa comunicação ente a população e a gestão.

[...] reclamar de algum problema na cidade via aplicativo instalado no smartphone e receber um posicionamento da prefeitura já é algo que ocorre em uma capital brasileira e está prestes a funcionar em outras duas. [...]. (Gomes, 2014, p.1).

Com a conclusão da pesquisa e análise dos dados colhidos com a população de Itaberaí, nota-se uma grande dificuldade que a população tem para solicitar uma ordem de serviço à gestão, mediante o problema porque não criar um sistema participativo entre a prefeitura e a população, um software simples, porém eficiente.

O público alvo deste estudo é a prefeitura de Itaberaí, e seus munícipes, pois nota-se na relação cidadão-prefeitura grande distanciamento e dificuldade de comunicação. A participação da população será promovida por meio de um software, que permitirá ao cidadão relatar um problema relacionado às áreas de interesse, tais como: buracos nas ruas, sinalização de trânsito, ocupação irregular, iluminação pública, limpeza e conservação, segurança, saúde, transporte público, etc. Com isso, a prefeitura será notificada e procederá ao atendimento à solicitação.

A proposta desse sistema vem trazer uma solução para algumas dificuldades que a população tem em participar ativamente da construção da cidade que ele deseja e permite que a prefeitura compreenda as demandas da população, facilitando o trabalho da própria prefeitura no atendimento ao cidadão, tornando-o assim protagonista no processo de melhoria da cidade em que ele vive e garantindo a cidadania. Com a colaboração da população a prefeitura ao invés de ter alguns funcionários observando onde precisa de uma manutenção, terá centenas de pessoas colaborando com o serviço sem custo.

[...] a gente busca fazer gestão colaborativa: uma gestão da cidade em que as prefeituras procurem a colaboração do cidadão no dia-a-dia", explica Gustavo Maia, um dos sócios fundadores do COLAB [...]. (Gomes, 2014, p.1).

Considerando por fim as opiniões de pessoas de nosso município, proporemos uma tecnologia que permita o desenvolvimento da cidade e a apropriação dos espaços públicos pelos cidadãos ampliando a protagonismo social e ofertando novas possibilidades para a gestão pública municipal.

1.6.1. ENTREVISTA

A pesquisa aponta que a população deseja uma ferramenta para solicitar serviços, 92,5% das pessoas entrevistadas disseram se importar com o uso dessa ferramenta, ela também apontou que 90,8% dessas pessoas tem dificuldades para solicitar serviços a gestão, o uso dessa aplicação facilitara a comunicação tanto da população quanto a gestão o que aponta a pesquisa que onde 90,6% delas acreditam que o uso dessa aplicação poderá facilitar a comunicação.

Prefeituras do brasil, já estão adotando essa forma de comunicação, visando as melhorias na cidade e estar mais próximo a população, tornando assim uma gestão mais eficiente e ágil.

[...] A Prefeitura de Ibaiti, visando receber as principais reivindicações dos ibaitienses está lançando um importante canal de comunicação entre a prefeitura e os moradores do município. Trata-se do aplicativo *Alô Prefeito*. No aplicativo é possível registrar reclamações, denúncias, elogios e sugestões dos serviços prestados pela prefeitura. [...]. (Sarrafho 2017, p.1)

2. REFERENCIAL TEORICO

Esta parte do projeto foi destinada para um estudo um pouco mais aprofundado da teoria em torno dos padrões e ferramentas que foram utilizadas durante o desenvolvimento de todo o projeto. O Estudo foi baseado em leituras de livros de importantes autores que em sua grande maioria são pessoas ativas em desenvolvimento de projetos de softwares.

Por isso a importância da leitura desse tipo de conteúdo, tendo em vista que uma das funções deste projeto é dar a possibilidade de o acadêmico adquirir alguma experiência prática no desenvolvimento de projetos de softwares.

Com o crescimento das cidades e o aumento da população, crescem também a necessidade gestão da estar mais próximo a população, A tecnologia vem avançando e ganhando cada dia mais espaço nas cidades, à medida que aplicativos são usados para uma participação entre a gestão e a população, a qualidade de vidas dos cidadãos melhora, melhorias essas, conforto, segurança, entre várias outras.

A informação, um dos principais ativos das organizações e, por consequência, o uso adequado das tecnologias da informação e comunicação tem sido tomados como fatores determinantes para o sucesso do gerenciamento da gestão pública. (POWACZUK, 2013 p.6)

2.1. ENGENHARIA DE SOFTWARE

A Engenharia de Software é um ramo da engenharia dedicado a softwares. Quando pesquisamos a definição de software, a resposta mais comum encontrada é que software é um programa de computador. Hoje em dia essa definição se tornou um pouco vaga, pois o programa é o resultado de um longo e específico trabalho, que é o de documentação que é tanto ou mais importante do que o programa em si, afinal a qualidade do mesmo está diretamente ligada a qualidade da documentação, senda assim é quase certo que se a documentação for bem feita o software será de qualidade “Software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente” (SOMMERVILLE, 2010 p.4).

A engenharia de software está ligada ao software desde de suas especificações até sua manutenção. Ela é utilizada de forma sistemática e organizada por parte dos engenheiros buscando sempre soluções práticas para problemas em relação ao desenvolvimento do software. Os sistemas que não se utilizam de forma mais específica da engenharia de software são os sistemas para a web que focam mais nas habilidades em projeto gráfico e em software.

É bom deixar claro que existe diferença entre engenharia de software e Ciência da computação, pois a ciência da computação se refere mais especificamente aos computadores e sistemas, já a Engenharia de software se dedica a solucionar problemas que ocorrem durante o desenvolvimento do software.

[...] a ciência da computação diz respeito as teorias e aos métodos que constituem a base de computadores e de sistemas de software, enquanto a engenharia de software se dedica a problemas práticos da produção de software[...]. (SOMMERVILLE, 2010 p.6)

Os processos e modelos de processos são parte importante da Engenharia de Software. Mas o que são? O processo de software é um conjunto de atividades que juntas formam um modelo do processo, ou seja, um modelo de processo nada mais é do que a união e organização dos processos seguindo um padrão mais adequado para cada sistema em si. Os processos são quatro:

- Especificação de software.
- Desenvolvimento de software.
- Validação de software.
- Evolução de software.

“Um modelo de processo é uma descrição simplificada desse processo de software que apresenta uma visão dele”. (SOMMERVILLE, 2010 p.6). Os modelos de processos seguem um modelo e a grande maioria são baseadas em três modelos gerais:

- O modelo cascata: que segue as etapas em sequência, sempre uma após a outra, só atinge a próxima etapa após a conclusão da anterior.
- Desenvolvimento iterativo: este não é tão sistemático quanto a modelo cascata, ele desenvolve o projeto muito rapidamente seguindo

especificações iniciais e depois complementa com informações de clientes e logo em seguida entrega o software para o cliente.

- Engenharia de software baseadas em componentes: este supõe que o sistema já exista e foca muito mais na integração de parte do que no desenvolvimento.

Quando se fala em desenvolvimento de software uma das perguntas principais é sobre o custo. Mas essa pergunta, não pode ser respondida com precisão, pois envolve muitos fatores, um deles é o modelo de processo que seja utilizado e depois vem outras bem mais variáveis, que são o tipo do sistema, sua complexidade e área de atuação.

[...]não existe uma resposta simples para essa pergunta pois a distribuição do custo ao longo das diferentes atividades do processo de software depende do processo e do tipo de software que está sendo desenvolvido[...]. (SOMMERVILLE, 2010. p. 7)

Engenharia de Software busca padronizar o desenvolvimento do software e encontrar formas de avaliar a qualidade deste. Existe uma parte que se dedica a isso, que são os atributos do software, esse tem por objetivo estudar os atributos do software depois avaliá-los, lógico que os atributos variam de um software para outro e de sua aplicação. "O conjunto específico de atributos que você pode esperar de um sistema de software depende, obviamente, da sua aplicação" (SOMMERVILLE, 2010, p. 9)

Atualmente a Engenharia de Software se depara com três grandes desafios. São eles:

- O desafio da heterogeneidade: Com a evolução da tecnologia se tornou necessário a junção de tecnologias mais novas com as mais antigas e ainda a necessidade de que os sistemas trabalhem em rede o principal desafio é construir um software flexível o bastante para se adaptar a essa heterogeneidade.
- O desafio da entrega: Hoje em dia tornou-se necessário a agilidade no desenvolvimento de um projeto de software. Já que grande parte das técnicas de engenharia demandam tempo o desafio é diminuir esse tempo sem comprometer a qualidade do software.

- O desafio da confiança: Esta busca formas de deixar claro que o software é confiável, afinal o mesmo está diretamente ligado à nossa vida cotidiana principalmente os sistemas web.

SOMMERVILLE, 2010. Em seu livro sobre engenharia de software cita um assunto muito importante ligado não somente a engenharia de software, mas também em relação a vida dos profissionais envolvidos ele cita quatro pontos-chaves que devem ser observados, são eles:

- Confidencialidade: Que é o respeito com informações de seus clientes e funcionários do mesmo, não sendo necessário a assinatura de um documento para tal.
- Competência: Este diz respeito ao trabalho em si, pois você não deve aceitar um trabalho em que você não tenha competência.
- Direitos sobre propriedade intelectual: Deve-se estar ciente das leis locais que regem o uso de propriedade intelectual assim como direitos autorais.
- Mau uso de computadores: Não é ético o uso de sua habilidade técnicas para fazer mau uso dos computadores de outras pessoas, isso começa desde de atitudes mais básicas por exemplo execução de jogos até as extremamente graves por exemplo a disseminação de vírus.

Como toda profissão um engenheiro de software possui grandes responsabilidades e grandes dificuldades sem falar no desafio que enfrenta a cada projeto, pois lhe são cobrados sempre os melhores produtos com um prazo muito reduzido para a entrega.

Durante o desenvolvimento das minhas opiniões utilizei as ideias de três autores distintos para pudesse tornar mais abrangente possível nosso ponto de vista sobre Engenharia de software essa primeira parte foi baseada nas informações extraídas do livro sobre Engenharia de software. (SOMMERVILLE, 2010).

Já de acordo com o (Pressman, 1995) houve várias definições para a engenharia de software, na qual todas elas trazem um reforço para a exigência em que ela tem. “A engenharia de software é um rebento da engenharia de sistemas e de hardware.” (Pressman, 1995, p.31.)

Com a utilização desta engenharia se tem uma grande vantagem para a construção de um software, afinal ela é benéfica tanto na projeção como na codificação, entre outras demais vantagens que serão vistas a seguir. Ela possui uma abrangência importante, que é um conjunto de três elementos que serão essenciais para o engenheiro e para o gerente, com este, estarão possibilitados a um maior controle do processo de desenvolvimento do software no decorrer da construção, e também a um melhor gerenciamento dos profissionais que estarão a construir o mesmo, assim eles terão uma boa base para que possam definir bem o trabalho de cada integrante da sua equipe e assim o seu trabalho seja mais organizado e satisfatório.

Este conjunto abrangido pela engenharia de software, é composto pelos seguintes elementos:

- **Os métodos:** estes trazem detalhes para o desenvolvimento do software. Nele é constituído um conjunto de tarefas que é composto por: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto da estrutura de programa e algoritmo de processamento, codificação, teste e manutenção. Em muitos casos os métodos apresentam uma notação orientada a linguagem especial ou gráfica, como também vários critérios que visam uma qualidade desejada do software a ser desenvolvido.
- **As ferramentas:** as ferramentas de engenharia de software, trazem uma determinada “ajuda” tecnológica, podendo ou não ser automatizados os métodos. Foram desenvolvidas ferramentas para o sustento dos métodos citados. Na utilização de uma determinada ferramenta que se cria uma informação em que se possa ser utilizada por outra, se tem o estabelecimento de um sistema de suporte para o desenvolvimento que está sendo feita. Este desenvolvimento é a *engenharia auxiliada por computador* (CASE- computer-Aided software engineering), que faz uma combinação do software, hardware e um banco de dados de engenharia de software. Com isso se pode criar um ambiente de engenharia de software que venha ajudar no projeto.

- **Os procedimentos:** com eles se há uma ligação dos métodos e as ferramentas e então se tem como diz o (Pressman, 1995, p.32) a possibilidade do “desenvolvimento racional e oportuno do software no computador”.

É com os procedimentos que se tem uma definição dos produtos que são exigidos para entrega, a sequência de aplicação dos métodos e os gerentes podem coordenar o controle, mudanças e avaliar os processos.

Cada um desses elementos também pode ser chamado de paradigma da engenharia de software, como outros muito discutidos como:

- O ciclo de vida clássico este também é denominado de modelo cascata;
- Prototipação esta que se executa como mecanismo para apoiar na detecção de requisitos de software.
- Modelo espiral, este na intenção de se ter uma abrangência em características mais específicas no ciclo de vida clássico (cascata) e também na prototipação, inserindo neles a análise de riscos como um novo elemento.

Técnicas de quarta geração este paradigma faz uma abrangência de um determinado conjunto de ferramentas de software que traz uma possibilidade ao profissional de especificar determinadas características do software, como a engenharia de requisitos, que é uma parte da engenharia de software destinada unicamente aos requisitos do sistema, que serão melhor explicadas um pouco mais a frente. Atualmente estas técnicas são consideradas fundamentais no desenvolvimento de software.

Já segundo Pfleeger (2004), a Engenharia de Software caminha em paralelo com os sistemas de informação, ambos temas destinados às organizações, para auxiliar as mesmas a tomarem decisões sob o foco de seu negócio empresarial ou de sua atividade pública. Primeiro é essencial entender a natureza do problema, pois não devemos impor máquinas e técnicas computacionais a toda questão que aparecer em nosso caminho.

Deve-se começar a investigação analisando o problema, isto é, dividindo-o em partes que sejam capazes de entender e manipular. Analisado o problema, deve-se elaborar a solução a partir de componentes que resolvam vários de seus aspectos. Para ajudar a resolver um problema utilizamos uma variedade de métodos, ferramentas, procedimentos e paradigmas.

[...] Método ou técnica: procedimento para a produção de um resultado. Ferramenta: instrumento ou sistema automatizado para realizar uma tarefa da melhor maneira (preciso, eficiente, produtivo) Procedimento: combinação de ferramentas e técnicas, que em harmonia, produzem um resultado específico. Paradigma: estilo de fazer algo, representa uma abordagem ou filosofia em particular para a construção do software (AOO e Procedural) [...]. (PFLEEGER, 2004, p.3).

O papel do engenheiro de software: profissionais na engenharia de software que se utilizam dos computadores e linguagem de programação como ferramentas, no projeto e na implementação da solução de um problema.

Os produtos de software não estão isentos de problemas como defeitos, erros e falhas. Um defeito ocorre quando uma pessoa comete um engano, chamado de erro, na realização de alguma atividade relacionada a um software. EXEMPLO: um projetista pode não compreender corretamente um requisito e criar um projeto que não condiz com a verdadeira intenção do analista de requisitos e do usuário.

Uma falha é uma divergência entre o comportamento requerido para o sistema e o comportamento real. Um defeito é uma visão interna do sistema, observada pelo ponto de vista dos desenvolvedores, enquanto uma falha é vista a partir de um referencial externo ao sistema, ou seja, um problema que o usuário vê.

Principais participantes da Engenharia de Software: Cliente, Desenvolvedor e Usuário.

[...] O **cliente** é a empresa, organização ou pessoa que está pagando para o sistema de software ser desenvolvido. O **desenvolvedor** é a empresa, organização ou pessoa que está construindo o sistema de software para o cliente. Essa categoria necessita de alguns gerentes para coordenar e orientar os programadores e a equipe que realiza os testes. O **usuário** é a pessoa, ou pessoas, que realmente utilizarão o sistema; aqueles que se sentarão em frente ao terminal para inserir dados ou ler os resultados[...]. (PFLEEGER, 2004, p.11).

A primeira coisa que devemos fazer, é entender o que o cliente deseja e necessita, e documentar os requisitos, temos que dividir em componentes para que

fique mais clara as informações. Após o projeto ser documentado, começa o trabalho dos programadores para descrever o sistema, implementaram o que foi especificado nos requisitos. Depois do código gerado por sua vez entrara a parte de teste, os primeiros testes são feitos com testadores da equipe que identificaram as possíveis falhas, após será testado com o cliente.

[...] para muitos sistemas de software, a aceitação do cliente não significa que o trabalho do desenvolvedor acabou. Se forem descobertos defeitos depois do sistema ser aceito, uma equipe de manutenção deve corrigi-los. Além disso, os requisitos do cliente podem mudar com o passar do tempo, e mudanças correspondentes no sistema deve ser realizada[...]. (PFLEEGER, 2004, p. 20).

O ciclo de vida é a estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso.

O modelo de ciclo de vida é a primeira escolha a ser feita no processo de software. A partir desta escolha definir-se-á desde a maneira mais adequada de obter as necessidades do cliente, até quando e como o cliente receberá sua primeira versão operacional dos sistemas.

Não existe um modelo ideal. O perfil e complexidade negócio do cliente, o tempo disponível, o custo, a equipe, o ambiente operacional são fatores que influenciarão diretamente na escolha do ciclo de vida de software a ser adotado. Da mesma forma, também é difícil uma empresa adotar um único ciclo de vida. Podemos citar alguns modelos de ciclo de vida são eles: Cascata, Modelo em V, Incremental, Evolutivo, RAD, Prototipagem, Espiral, Modelo de Ciclo de Vida Associado ao RUP e Modelo em Cascata.

2.1.1. ENGENHARIA DE REQUISITOS

A Engenharia de requisitos é o processo em que se descobre as funcionalidades do sistema suas restrições e atuação do sistema. “A engenharia de requisitos é um processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos de sistema. ” (SOMMERVILLE, 2010 p. 103).

Ela envolve quatro passos que se completam são eles:

- Obtenção e análise.
- Especificação dos requisitos.
- Validação dos requisitos.

O primeiro passo da engenharia é o estudo da viabilidade que é uma visão geral de como o sistema funcionará na organização, como ele pode contribuir.

Os próximos passos envolvem a obtenção e análise de requisitos a documentação e validação de requisitos em outras palavras, pesquisam sobre o tema fazendo entrevistas com pessoas que possuem conhecimento e atuam na área em que o software atuará, entrevistam gerentes e pessoas que possam contribuir com desenvolvimento do projeto.

A engenharia de requisitos, é um processo de criação de software que se baseia em requisitos que também podem ser chamados de serviços e/ou restrições. “O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado de engenharia de requisitos” (SOMMERVILLE, 2010, p. 79).

De acordo com o (Pressman 1995.) A Engenharia de Requisitos contém uma tarefa chamada análise de requisitos ela faz uma ligação entre as pesquisas realizadas para a construção do software. É através desta tarefa em que o engenheiro encontra as funções na qual o software construído terá, e também encontrará uma noção do desempenho do mesmo.

A análise de requisitos também permite ao engenheiro uma representação de como será um determinado software quando pronto, e ela traz uma base para todas as outras atividades da engenharia de software que deverão ser seguidas.

Dentro da análise de requisitos pode haver uma divisão de quatro partes, sendo estas divisões atividades de análise. São elas:

- Reconhecimento do problema
- Modelagem
- Especificação
- Revisão

Segundo ” PFLEEGER, 2004. Um requisito é um levantamento do projeto que iremos trabalhar, levantaremos as principais características para começarmos a montar o projeto.

[...] requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar para atingir seus objetivos três categorias de requisitos: requisitos que devem ser totalmente satisfeitos requisitos que são altamente desejáveis, mas não necessários requisitos que são possíveis, mas poderiam ser eliminados[...]. (PFLEEGER, 2004 p. 20).

Definição dos requisitos listagem completa de tudo que o cliente espera que o sistema proposto faça. Especificação dos requisitos: redefine os requisitos em termos técnicos apropriados para o desenvolvimento do projeto do sistema; gerência de configuração: correspondência direta entre os dois requisitos.

Conjunto de procedimentos e a parte que controlam, os requisitos que definem o que o sistema deverá fazer, os módulos de projeto gerados a partir dos requisitos, o código do programa que implementa o projeto, os testes que verificam a funcionalidade do sistema, os documentos que descrevem o sistema, requisitos funcionais versus não funcionais.

Cada entidade no sistema é um objeto um método ou uma operação é uma ação que pode ser realizada pelo objeto ou pode acontecer com o objeto, encapsulamento maneira pela qual os métodos formam um limite de proteção em torno do objeto, polimorfismo mesmo método para diferentes objetos, cada qual com diferentes comportamentos.

Protótipo descartável é usualmente uma implementação prática do sistema, é produzida para ajudar a levantar os problemas com os requisitos, e depois é descartado. O sistema é então desenvolvido usando algum outro processo de desenvolvimento. Usada para reduzir os riscos com os requisitos, O protótipo descartável NÃO deve ser considerado como um sistema final.

Protótipo evolutivos como o nome sugere, evoluem de uma iteração para a próxima. Enquanto não houver qualidade de produção inicialmente, o código tende a ser retrabalhado conforme o produto é desenvolvido. Para manter o retrabalho gerenciável, eles tendem a ser projetados mais formalmente e testados com uma

certa formalidade mesmo nos estágios iniciais. Conforme o desenvolvimento do produto, o teste torna-se formalizado, como normalmente é o design.

2.1.1.1. REQUISITOS FUNCIONAIS

Segundo (SOMMERVILLE, 2010, p. 79) “Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais”.

Os requisitos podem ser divididos em requisitos funcionais e não-funcionais.

Os requisitos funcionais fazem referência ao que o sistema deve fazer, variam de software para software e dependem dos usuários a que se destina e da abordagem geral do sistema. “(...) os requisitos funcionais descrevem a função do sistema detalhadamente, suas entradas e saídas, exceções”. (SOMMERVILLE, 2010, p. 81).

O estudo dos requisitos é de grande importância, pois é comum o erro na interpretação do mesmo, o que causa muitos transtornos se não descoberto a tempo e antes da entrega do sistema ao cliente. “A imprecisão na especificação de requisitos é o motivo de muitos problemas de engenharia de software”. (SOMMERVILLE, 2010 p. 81).

Já segundo Pressman, 1995 requisitos funcionais são aqueles em que se define uma determinada função de um sistema de software ou até mesmo de seu componente. Esta função é descrita como um conjunto de entradas, comportamento e saídas. Estes requisitos podem ser processamento, detalhes técnicos, manipulação de dados e de processamento e como também funções específicas que trazem uma definição que um sistema, poderá realizar.

2.1.1.2. REQUISITOS NÃO-FUNCIONAIS

Os requisitos não funcionais aplicam-se as restrições do software em geral, e não dos serviços que o software oferece. “Os requisitos não funcionais, como o nome sugere, são aqueles não diretamente relacionados as funções específicas fornecidas pelo sistema”. (SOMMERVILLE, 2010, p. 82).

O não entendimento de um requisito não funcional pode significar o fracasso de um sistema, pois o mesmo envolve muitos fatores importantes para o funcionamento do sistema, como as restrições, que pode ser a portabilidade que é capacidade de um sistema de software ser executado por um Sistema Operacional que é a base onde qualquer software deve rodar, então se o Software for desenvolvido para ser executado em Linux, e o cliente tiver apenas computadores com Windows, isso pode significar o fracasso do projeto já que este pode não ser capaz de rodar em uma plataforma diferente daquela para a qual foi produzido, a menos que utilize alguma linguagem de programação que rode em uma máquina virtual disponibilizada pela empresa responsável pela linguagem, neste caso o projeto mesmo sendo produzido para rodar em Linux, seria capaz de rodar em um Windows, porém fica claro que o projeto sobreviveu graças a ferramenta utilizada e não ao Engenheiro já que este não especificou bem os requisitos não-funcionais.

[...] Os requisitos não funcionais surgem devido às necessidades do usuário, às restrições de orçamento, às políticas organizacionais, à necessidade de interoperabilidade com outros sistemas de software ou hardware ou a fatores externos, como regulamentos de segurança ou legislação a respeito da privacidade[...] (SOMMERVILLE, 2010 p. 82).

Já segundo Pressman (1995) requisitos não funcionais tem uma certa relação aos usos da aplicação do desempenho, confiabilidade, segurança, disponibilidade, manutenção e outras tecnologias que estejam envolvidas. Fica a cargo do desenvolvedor desejar atender estes requisitos ou não. (Pressman, 1995).

Segundo Pfleeger (2004), o requisito não-funcional descreve uma restrição no sistema que limita nossas opções para criar uma solução para o problema, um exemplo, podemos receber a informação de que o sistema deverá ser desenvolvido em um computador ou de contracheques devem ser distribuídos para os funcionários em não mais do que quatro horas depois de os dados iniciais terem sido lidos. (PFLEEGER, 2004, p.115).

2. 2. BANCO DE DADOS

O banco de dados possibilita um armazenamento de dados do nosso dia a dia, ele está presente na grande maioria senão em todos os projetos de sistemas.

Em um banco de dados você pode armazenar, alterar e excluir, por meio de uma linguagem específica que trabalha com manipulação dos dados de um banco de dados, que é a linguagem SQL (Structured Query Language).

Geralmente todas as aplicações utilizadas para o armazenamento e manipulação de dados são compostas por três elementos básicos: campos, registros e tabelas. Os bancos de dados passaram a ser tratados como componente essencial em um bom de projeto de software.

[...] o projeto de banco de dados relacional deixou de ser uma arte para se tornar uma ciência parcialmente implementável por um conjunto de recursos de projeto de software. Muitos desses recursos de projetos surgiram como componentes de banco de dados das ferramentas de engenharia de software auxiliada por computador (Computer-Aided Software Engineering – CASE), e muitos deles oferecem capacidade de modelagem interativa usando uma abordagem simplificada de modelagem de dados [...]. (TEOREY, LIGHTSTONE e NADEAU. 2007. p 01).

A função de um MER (Modelo Entidade-Relacionamento) é separar, trocar os dados entre os depósitos de dados, imagine um supermercado, no supermercado temos várias seções e cada seção tem determinados produtos temos seção somente de arroz e feijão, bolachas, secos e molhados, produtos de beleza e assim sucessivamente, produtos da mesma empresa que não podem ficar na mesma seção, assim e um (MER) ele separa os dados armazenados que não fica no mesmo lugar, ficam em tabelas diferente ou seja em depósitos diferentes e facilitando a comunicação com o usuário. A forma simples do modelo ER e usada como base para a comunicação eficaz com o usuário final sobre o banco de dados conceitual. (TEOREY, LIGHTSTONE e NADEAU. 2007, p 13).

O MER é classificado por três objetos básicos que são ele Entidade, Relacionamento e Atributos.

Entidade representa o objeto a ser salvo em um banco de dados, esta é escrita em forma de retângulo com o nome do objeto em questão escrito com a primeira letra do nome em letra maiúscula. “[...] elas normalmente representam uma pessoa, lugar, coisa ou evento de interesse informativo. (TEOREY, LIGHTSTONE e NADEAU, 2007, p 13).

Relacionamento ele tem a função de relacionar a entidade com outras entidades, ele pode relacionar uma ou mais entidades não possuem existência fixa ou conceitual, e escrito em forma de losango conectando as entidades associadas.

Atributos são características das entidades que são detalhes que descreve elas, existem dois tipos de atributos identificadores e descritores. Um identificador é, por exemplo, a identidade de um produto, pessoa e etc. os descritores são as características no caso de uma pessoa será o nome, endereço, telefone e etc.

2. 3. ANALISE ORIENTADA A OBJETOS

Orientação a objetos é uma perspectiva de desenvolvimento de software que vem ocupando bastante espaço, agora que está comprovado sua eficiência em desenvolvimento nos mais variados níveis de complexidade e tamanho de projetos. “Nessa visão, o principal bloco de construção de todos os sistemas de software é o objeto ou a classe” (BOOCH, RUMBAUCH e JACOBSON, 2005. p. 11).

Mas para compreender melhor essa perspectiva é necessária definir bem o que é um objeto e uma classe. Muito bem, um objeto é algo “do mundo real” que é implementado em sistema, ou seja, o objeto é algo que faz parte do problema ou da solução que é identificado e assim transportado para o sistema. Já as classes é um conjunto de objetos, afinal os objetos são instanciados a partir de uma classe, por isso diz objeto é a instância de uma classe.

[...] explicando de uma maneira simples, um objeto é alguma coisa geralmente estruturada a partir do vocabulário do espaço do problema ou do espaço da solução; uma classe é a descrição de conjunto objetos comuns [...]. (BOOCH, RUMBAUCH e JACOBSON, 2005. p. 11).

A orientação a objetos busca a representação do mundo real, através das classes que são moldes para se definir um ou mais objetos, os objetos possuem estados e ações. O conceito pode parecer um pouco confuso e as vezes complicado pois procura elementos do mundo real para assim representá-lo em parte do software. Não é uma ideia muito jovem, ela vem desde a década de 1960 já com algumas linguagens criadas com o objetivo de difundi-la.

Como dito pode parecer complicado, mas é uma realidade e facilita o desenvolvimento do projeto desde a documentação que demanda um pouco mais de

atenção e tempo até a codificação que fica bem mais simples com toda a documentação em mãos.

De acordo com a conclusão de Eduardo Bezerra em seu livro *Princípios de Análise e Projeto de Sistemas com UML*, o paradigma da orientação a objetos “é uma forma de abordar um problema”.

Através do paradigma de orientação a objetos podemos então ter uma abrangência específica do tema abordado.

Um sistema de software tem em si como atributos objetos e classes, na qual terá funções detalhadas em partes sendo isto uma “obrigação” dos objetos, enquanto as classes guardarão os comportamentos ligados aos objetos.

Para uma conclusão mais completa do assunto abordado acima, observamos uma conclusão geral de (BEZERRA, 2007).

O paradigma da orientação a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada objeto é responsável por realizar tarefas específicas. É pela interação entre objetos que uma tarefa computacional é realizada.

2. 4. UML (Unified Modeling Language)

UML (Unified Modeling Language) é uma linguagem e foi criada com intuito de padronizar a documentação que ocorre durante o desenvolvimento de projetos de softwares, como toda linguagem possui vocabulário e sintaxe.

[...] a UML (Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software[...]. (BOOCH, RUMBAUCH, JACOBSON, 2005. p. 13).

A UML define apenas os vocabulários para a criação e leitura de modelos, porém não define nem quais nem quando esses modelos deverão ser utilizados isso depende do projeto que está sendo desenvolvido.

UML (Unified Modeling Language) é uma linguagem de modelagem segue padrões e é uma importante ferramenta para os desenvolvedores de softwares. A modelagem permite a compreensão de um sistema.

Como toda linguagem possui vocabulários e regras, ela utiliza essas regras para a representação do funcionamento do negócio e do sistema em si. A UML é usada na documentação do projeto nela estão os diagramas que atualmente são amplamente utilizados na fase de codificação, pode e serve de molde para o programa. (Principalmente o diagrama de classes).

A UML é um tipo de linguagem específica em visualização, na qual tem a função de modelar sistemas orientados a objetos. Ou seja, é com esta linguagem que definimos elementos gráficos, que na modelagem dos sistemas de software podem ser utilizados sendo eles visuais o que pode trazer uma facilitação para o engenheiro.

2.4.1 Diagrama Caso de Uso

De forma objetiva podemos definir caso de uso como sendo o cenário de um sistema, nesse cenário estão especificados os requisitos os atores e interações acerca do projeto, que deverá ocorrer dentro do sistema já implementado sem necessariamente especificar como isso irá ocorrer.

Um ator está presente no caso de uso, ele pode ser o usuário ou outro sistema que tem acesso ao seu com o objetivo de obter alguma informação.

[...] um ator representa um conjunto coerente de papéis que os usuários dos casos de uso desempenham quando interagem com esses casos. Os atores podem ser humanos ou sistemas automatizados[...]. (BOOCH, RUMBAUCH e JACOBSON, 2005, p. 229).

Todo sistema possui atores e ações, o caso de uso que é responsável por demonstrar essa interação entre os atores e o sistema, tratando as possíveis ações a serem executadas pelo sistema conforme a solicitação do ator em questão. Os casos de usos são graficamente representados por uma elipse e escritos na forma de verbos no infinitivo (sem conjugação) exemplo: cadastrar cliente.

Um caso de uso é um “personagem” indispensável dentro do desenvolvimento do projeto de sistemas. Ele é definido como uma especificação de

uma sequência completa de interações entre um sistema e/ou um ou mais agentes externos a esse sistema. Ele é uma representação de uma certa funcionalidade de um sistema, ele representa também os agentes externos que tenha alguma interação com o sistema. Este não demonstra de forma alguma o comportamento interno do sistema e nem a estrutura.

2.4.2. DIAGRAMA DE ATIVIDADE

Como o nome sugere um diagrama de atividade é utilizado para especificar como o sistema se comporta diante determinada atividade, ou seja, ele define algumas respostas lógicas que o sistema trará no decorrer de tal atividade. “[..]isso envolve a modelagem das etapas sequenciais (e possivelmente concorrentes) em um processo computacional” (BOOCH, RUMBAUCH e JACOBSON, 2005, p. 268).

[...] os diagramas de atividades são um dos cinco diagramas disponíveis na UML para a modelagem de aspectos dinâmicos de sistemas[...]. (BOOCH, RUMBAUCH, JACOBSON. 2005, p. 268).

O diagrama de atividades é utilizado para a modelagem das ações do sistema, eles dão ênfase ao fluxo de controle de uma etapa para outra. São muito importantes também para a construção de sistemas executáveis. “Um diagrama de atividade é essencialmente um fluxograma que dá ênfase à atividade que ocorre ao longo do tempo. (BOOCH, RUMBAUCH e JACOBSON, 2005, p.270).

Os diagramas de atividades são aqueles que são orientados a fluxos de controle. Ele é semelhante ao diagrama de estados, porém existe uma diferença nas representações, onde o diagrama de atividade representa os estados de uma atividade.

O diagrama de atividades tem dois elementos que o compõe que são, um utilizado para representar fluxos de controle sequenciais e outro para representar fluxos de controle paralelos.

2.4.3. DIAGRAMA DE CLASSE

As classes são a base de um sistema OO (Orientado a Objetos) os diagramas de classes são os mais comumente encontrados em projetos que seguem esse paradigma. Estes Especificam cada uma das classes definidas e suas

relações, são amplamente utilizados na fase de codificação, já que seguem um padrão que é muito bem definido cabendo ao programador apenas codificar o que está escrito não sendo necessário a definição de novas classes, claro levando em conta que o projeto foi muito bem projetado inclusive nos aspectos lógicos, ou seja, não haja inconsistência nem requisitos contraditórios. “Um diagrama de classe mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. (BOOCH, RUMBAUCH, JACOBSON, 2005, p. 107).

Os diagramas de classes são amplamente encontrados nos projetos para a modelagem de sistemas orientados a objetos. Eles são a base para outros diagramas como os diagramas de componentes e os diagramas de implantação. Eles possuem classes, interfaces, relacionamentos de dependência, generalização e associação. De extrema importância este é utilizado inicialmente no nível de análise, até o nível de especificação. O diagrama de classes é o mais rico em termos de notação dentre os demais diagramas da UML. Este diagrama se compõe de classes, associações, multiplicidade, participações, classes associativas, associações ternárias, associações reflexivas, agregação e composições entre outros mais, que são elementos utilizados para a construção do modelo de classe no nível de análise.

Os diagramas são formados por suas classes atributos e métodos, nestes devem especificados tipos de dados e visibilidade no que diz respeito aos atributos, já os métodos devem ser definidos os parâmetros de entrada e os tipos de retorno que o método possua. Todos os relacionamentos entre as classes também devem estar especificados e são estes padrões que fazem com que os diagramas de classes sejam tão importantes na documentação de um sistema de software.

2.4.4. DIAGRAMA DE SEQUENCIA

Cada caso de uso do sistema deve ser representado por um diagrama de sequência, onde nele cada passo envolvendo um determinado caso de uso deve ser especificado, podendo ser uma confirmação ou simplesmente um cadastro simples no sistema, cada ação e resposta deve estar presente nele.

O diagrama de sequência é voltado para o comportamento do sistema, enfatizando a ordem temporal em que as mensagens são trocadas entre os objetos de um determinado processo. Esse diagrama identifica o evento causador de um

processo, bem como os autores envolvidos, e determina como o processo deve atuar e terminar através de mensagens entre os objetos. (GUEDES, 2011. P33).

2.5. JAVA EE

A Java EE (Java Platform, Enterprise Edition) consiste em uma plataforma para desenvolver aplicações Java para internet, possuindo bibliotecas e funcionalidade para ser implementada em software, que se baseia em componente modular, executado em servidores de aplicações. A plataforma Java ee possui diversas especificações, por esse motivo ela é considerada uma plataforma guarda chuva. As especificações mais conhecidas são Servlets, JSP (Java Server Pages) e JSF (Java Server Faces).

Uma aplicação web refere-se a sites que ficam hospedados em grandes servidores, que possibilita o usuário acessá-los em seus celulares ou computadores através de um navegador. Servidores que são chamados de Web Server.

[...] o acesso às páginas desses sistemas é feito utilizando o modelo chamado de request-response, ou seja, o cliente solicita que alguma ação seja realizada (request) e o servidor a realiza e responde para o cliente (response). Na plataforma Java, esse modelo foi implementado através da API de Servlets. Um Servlet estende a funcionalidade de um servidor web para servir páginas dinâmicas aos navegadores, utilizando o protocolo HTTP[...]. (FARIA, 2013. p.10)

Um protocolo HTTP é um protocolo usado para visualizar na sua máquina (celulares, tablets, computadores entre outros aparelhos), os conteúdos que estão armazenados ou processados nos servidores remotos. Quando estamos navegando em um endereço web, geramos uma requisição ao servidor e ele nos devolve para o browser (navegador) o conteúdo da página HTML.

[...] o HTTP é um protocolo stateless de comunicação cliente-servidor: o cliente envia uma requisição para o servidor, que processa a requisição e devolve uma resposta para o cliente, sendo que, a princípio, nenhuma informação é mantida no servidor em relação às requisições previamente recebidas[...]. (FARIA, 2013. p.10).

2.6. DESENVOLVIMENTO

2.6.1. MODELO ENTIDADE RELACIONAMENTO

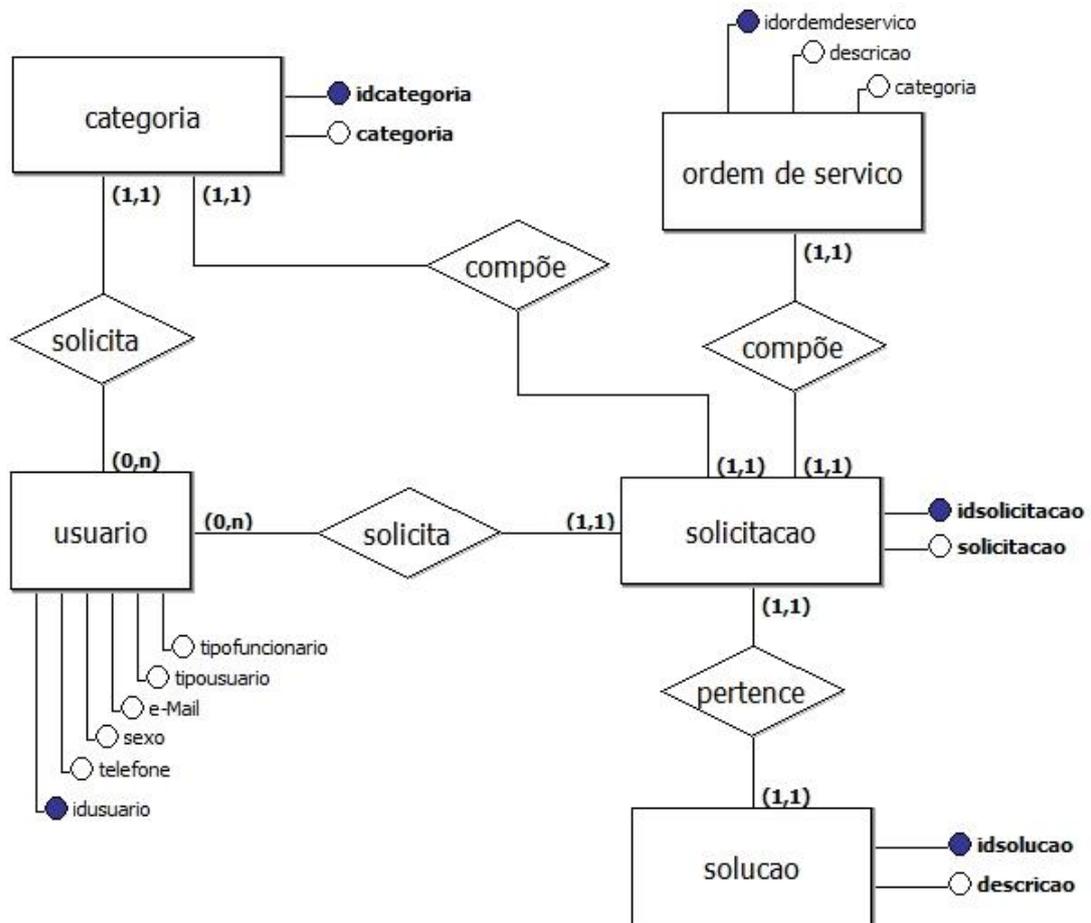


Figura 1 – Modelo Entidade Relacionamento

2.6.2. MODELO RELACIONAL NORMALIZADO

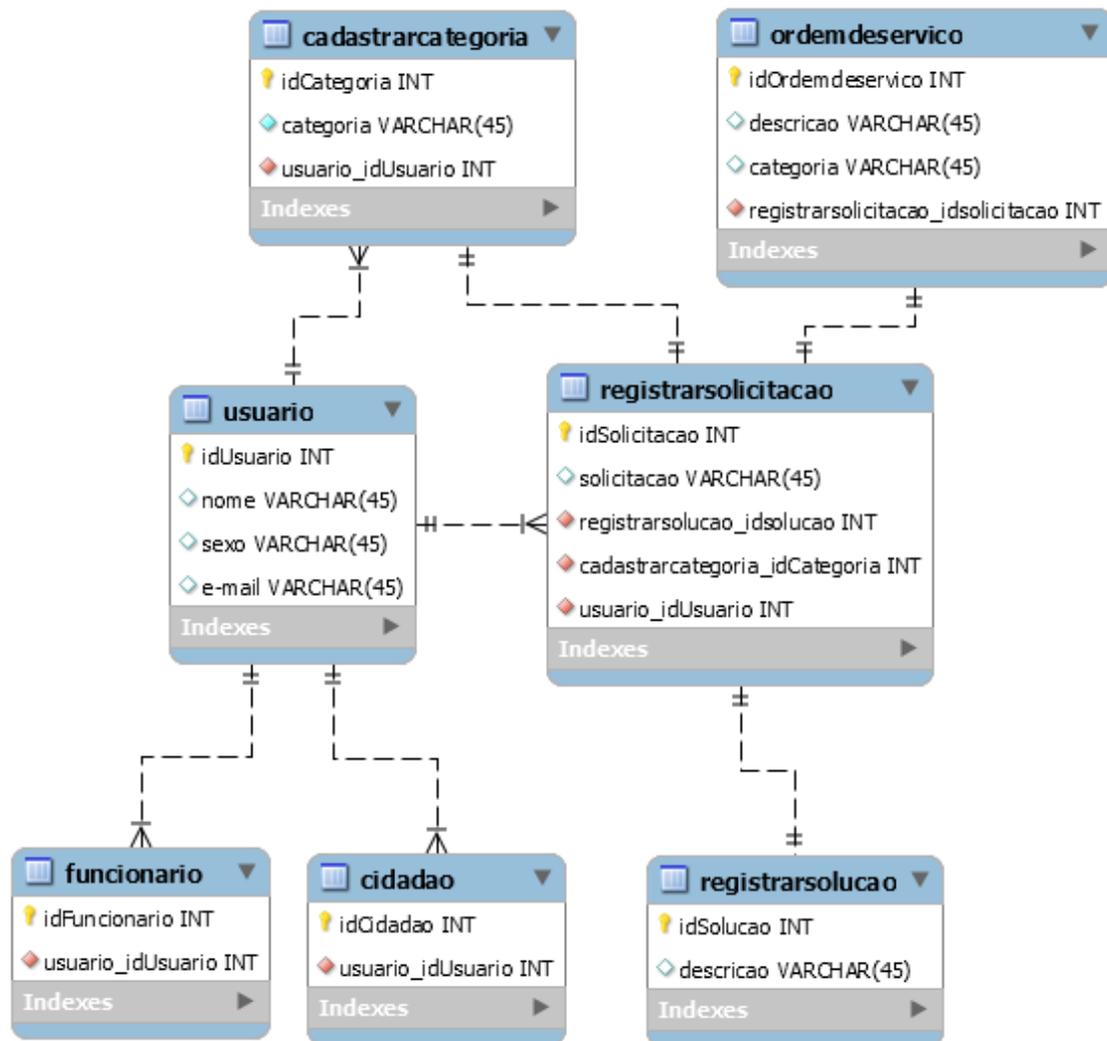


Figura 2 – Modelo Relacional Normalizado

2.6.3. DIAGRAMA DE CLASSES

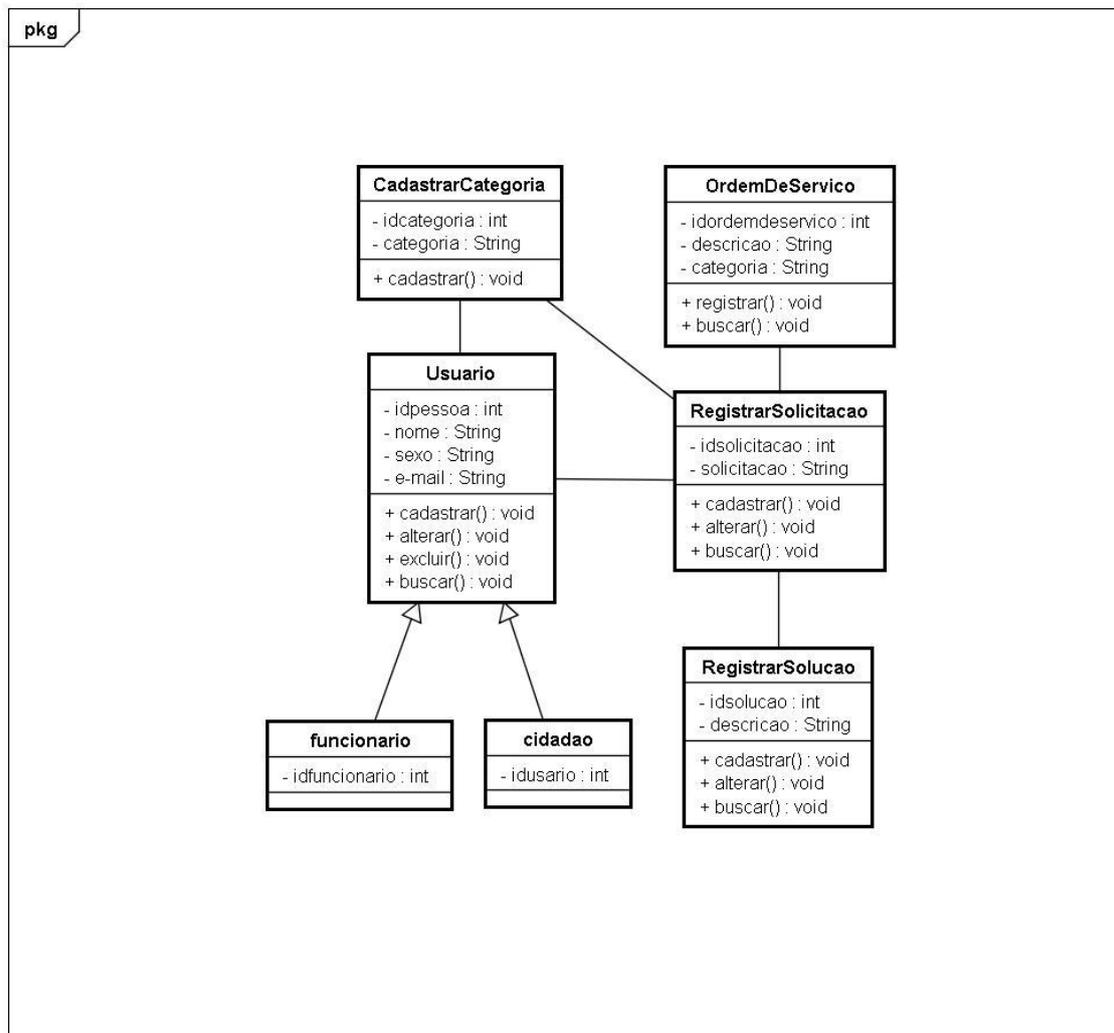


Figura 3 – Diagrama de Classes SGP

2.6.4. DIAGRAMA DE CASO DE USO

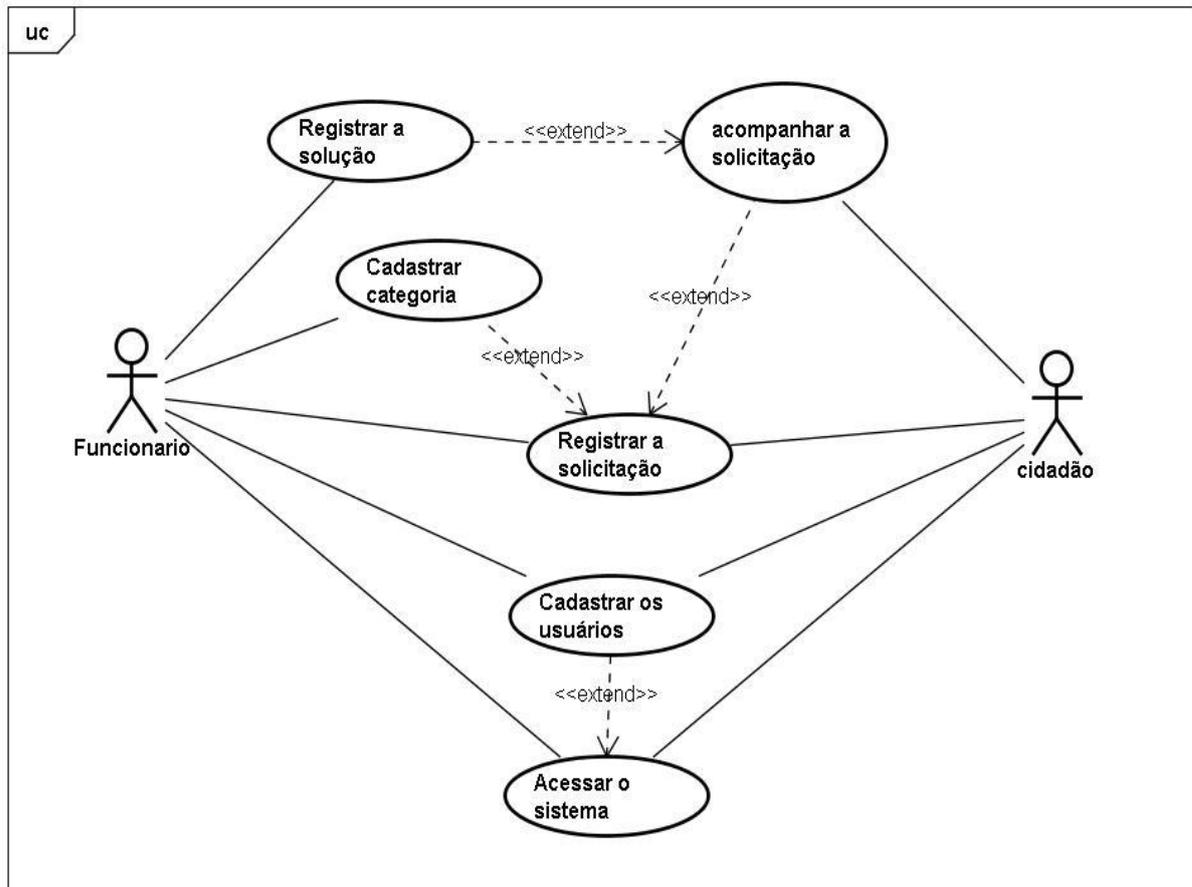


Figura 4 – Diagrama de Caso de Uso SGP

2.6.5. ESPECIFICAÇÃO DE CASOS DE USO

UC 01 - cadastrar usuário	
Atores	Funcionário, cidadão
Resumo	Esta aplicação deve permitir que o usuário se cadastre na plataforma.
Pré-condições	O usuário deve estar conectado à internet, e ter acesso a seus dados pessoais.
Pós-condição	Após o cadastro, o sistema cadastra os dados no banco de dados do sistema.
Fluxo principal	Se inicia quando o usuário seleciona a opção se cadastrar.
Ações do ator	Ações do sistema.
	1. O ator seleciona a opção cadastrar no sistema.
2. O sistema exibe a tela de cadastro.	3. O autor preenche as informações necessárias para cadastrar.
4. O ator confirma as informações necessárias para se cadastrar.	5. O sistema armazena as informações, e confirma o cadastro.
Fluxo alternativo	1 a. o ator clica no botão cancelar. 5 a. se as informações estiverem incorretas, e retornado uma mensagem.
Regras de negocio	RN 01. Não se pode ter duplicidade de cadastro.

Especificação do Caso de Uso “Cadastrar usuário”

UC 02 - Acessar	
Atores	Funcionário, cidadão
Resumo	Esta aplicação deve validar as informações cadastradas, e-mail e senha.
Pré-condições	O usuário deve estar conectado à internet, e ter acesso a seus dados pessoais.
Pós-condição	Após o cadastro, o sistema cadastra os dados no banco de dados do sistema.
Fluxo principal	Se inicia quando o usuário inserir seu e-mail e senha.
Ações do ator	Ações do sistema.
	1. O sistema exibirá uma tela contendo o campo e-mail e senha, e a opção cadastrar usuário.
2. O ator informa seu e-mail e senha.	3. O ator clica no botão entrar.
4. O sistema autentica as informações.	5. O sistema abre então a plataforma.
Fluxo alternativo	1 a. o ator clica na opção cadastrar e o sistema exibe uma tela de cadastro. 5 a. se as informações estiverem incorretas, e retornado uma mensagem.
Regras de negocio	RN 01. A senha do ator não pode ser exibida na tela, deve ser descrita com caracteres especiais.

Especificação do Caso de Uso " Login"

UC 03 - Registrar solicitação	
Atores	Cidadão
Resumo	Esta aplicação deve permitir que o usuário registrar a solicitação.
Pré-condições	O usuário deve estar conectado à internet, e ter acesso a seus dados pessoais para cadastrar a solicitação.
Pós-condição	Após o cadastro, o sistema cadastra os dados no banco de dados do sistema.
Fluxo principal	Se inicia quando o usuário seleciona a opção se cadastrar.
Ações do ator	Ações do sistema.
	1. O ator seleciona a opção registrar solicitação.
2. O sistema exibe a tela de registrar solicitação.	3. O ator preenche as informações necessárias para cadastrar.
4. O ator confirma as informações necessárias para cadastrar a solicitação.	5. O sistema armazena as informações, e confirma o cadastro.
Fluxo alternativo	1 a. o ator clica no botão editar. 5 a. se as informações estiverem incorretas, e retornado uma mensagem.
Regras de negocio	RN 01. Não se pode ter duplicidade da mesma solicitação.

Especificação do Caso de Uso “Registrar solicitação”

UC 04 - cadastrar categoria	
Atores	Funcionário
Resumo	Esta aplicação deve permitir que o usuário cadastrar a categoria, da solicitação que o cidadão fez.
Pré-condições	O usuário deve estar conectado à internet, e ter acesso a seus dados pessoais para cadastrar a categoria.
Pós-condição	Após o cadastro, o sistema cadastra os dados no banco de dados do sistema.
Fluxo principal	Se inicia quando o usuário seleciona a opção se cadastrar.
Ações do ator	Ações do sistema.
	1. O ator seleciona a opção cadastrar categoria
2. O sistema exibe a tela de cadastrar categoria.	3. O ator preenche as informações necessárias para cadastrar.
4. O ator confirma as informações necessárias para cadastrar categoria.	5. O sistema armazena as informações, e confirma o cadastro.
Fluxo alternativo	1 a. o ator clica no botão editar. 5 a. se as informações estiverem incorretas, e retornado uma mensagem.
Regras de negocio	RN 01. Pode ter duplicidade da mesma categoria cadastrada.

Especificação do Caso de Uso "Cadastrar categoria"

UC 05 - Registrar solução	
Atores	Funcionário
Resumo	Esta aplicação deve permitir que o usuário registrar uma solução para que o cidadão possa ter um feedback da solicitação
Pré-condições	O usuário deve estar conectado à internet, e ter acesso a seus dados pessoais para registrar a solução.
Pós-condição	Após o cadastro, o sistema cadastra os dados no banco de dados do sistema.
Fluxo principal	Se inicia quando o usuário seleciona a opção se cadastrar.
Ações do ator	Ações do sistema.
	1. O ator seleciona a opção registrar solução.
2. O sistema exibe a tela de registrar solução.	3. O ator preenche as informações necessárias para cadastrar.
4. O ator confirma as informações necessárias para registrar a solução.	5. O sistema armazena as informações, e confirma o cadastro.
Fluxo alternativo	1 a. o ator clica no botão editar. 5 a. se as informações estiverem incorretas, e retornado uma mensagem.
Regras de negocio	RN 01. O ator só pode registrar a uma solução por vez.

Especificação do Caso de Uso "Registrar solução"

UC 06 - Acompanhar solicitação	
Atores	Cidadão
Resumo	Esta aplicação deve permitir que o usuário acompanhe o feedback da solicitação
Pré-condições	O usuário deve estar conectado à internet, e ter acesso a seus dados pessoais para acompanhar a solicitação.
Pós-condição	Após o cadastro, o sistema cadastra os dados no banco de dados do sistema.
Fluxo principal	Se inicia quando o usuário seleciona a opção acompanhar solicitação.
Ações do ator	Ações do sistema.
	1. O ator seleciona a opção acompanhar solicitação.
2. O sistema exibe a tela de acompanhar solicitação.	3. O ator verifica a solicitação.
Fluxo alternativo	1 a. o ator clica no botão sair ou voltar. 5 a. se as informações estiverem incorretas, e retornado uma mensagem.
Regras de negocio	RN 01. O ator não pode editar nenhuma informação nessa tela.

Especificação do Caso de Uso “Acompanhar solicitação”

2.6.6. DIAGRAMAS DE SEQUÊNCIA

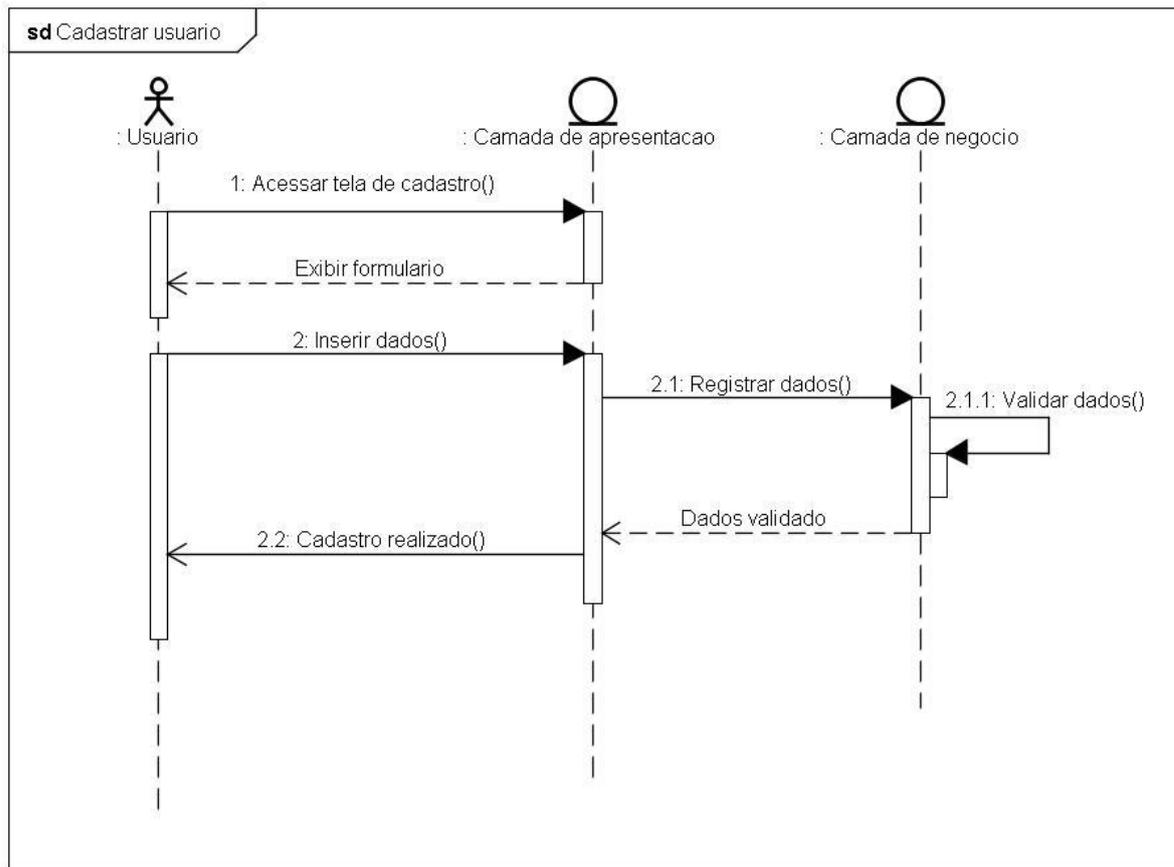


Figura 5 – Diagrama de Sequência – Cadastro Do Usuário

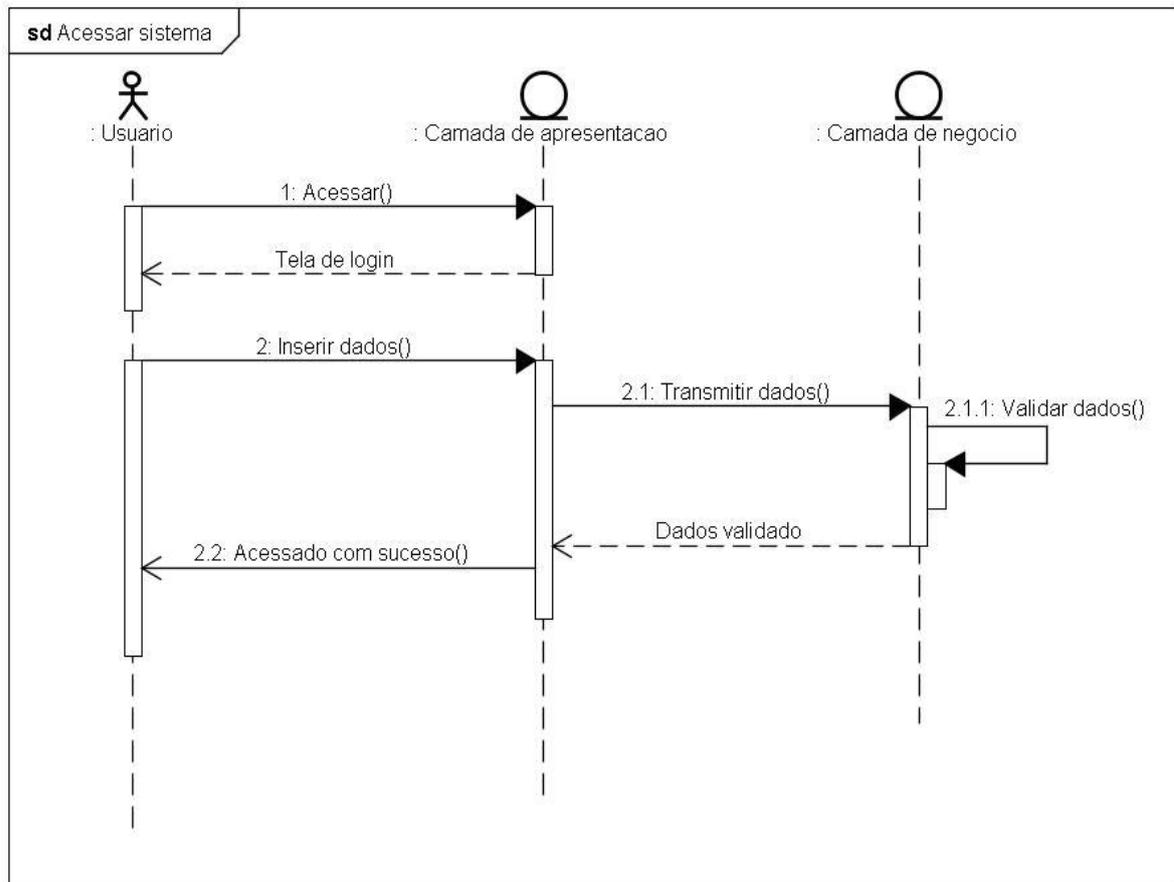


Figura 6 – Diagrama de Sequência – Acessar sistema

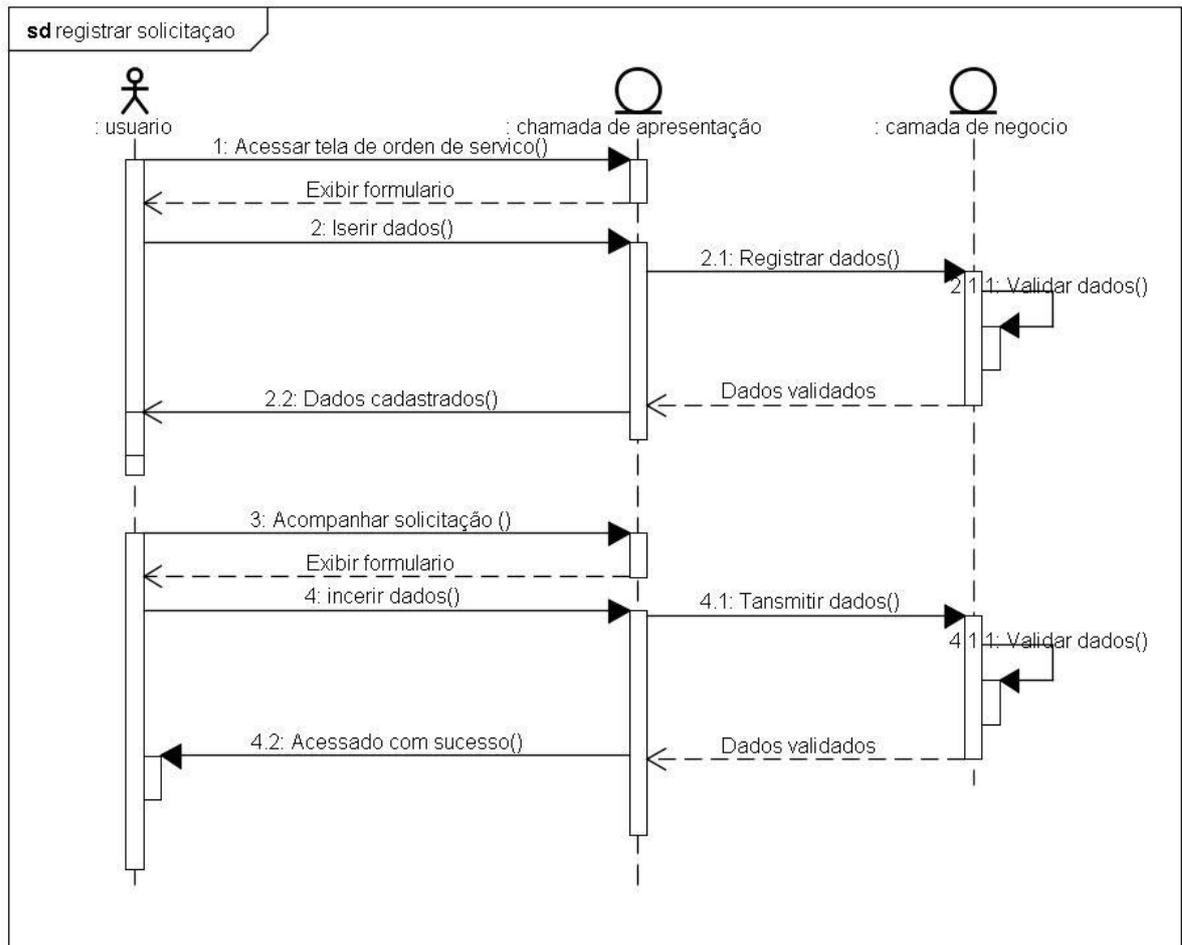


Figura 7 – Diagrama de Sequência – Registrar solicitação

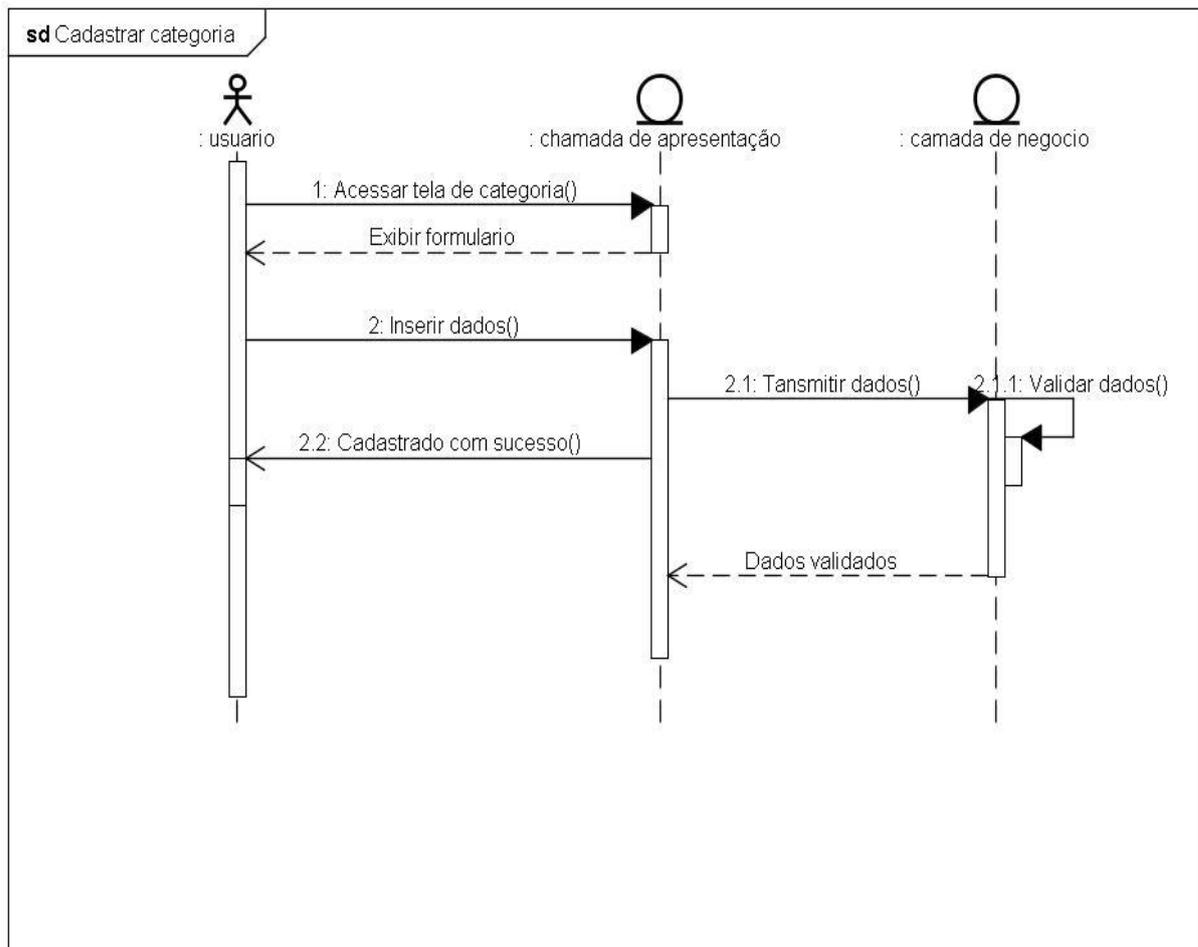


Figura 8 – Diagrama de Sequência – cadastrar categoria

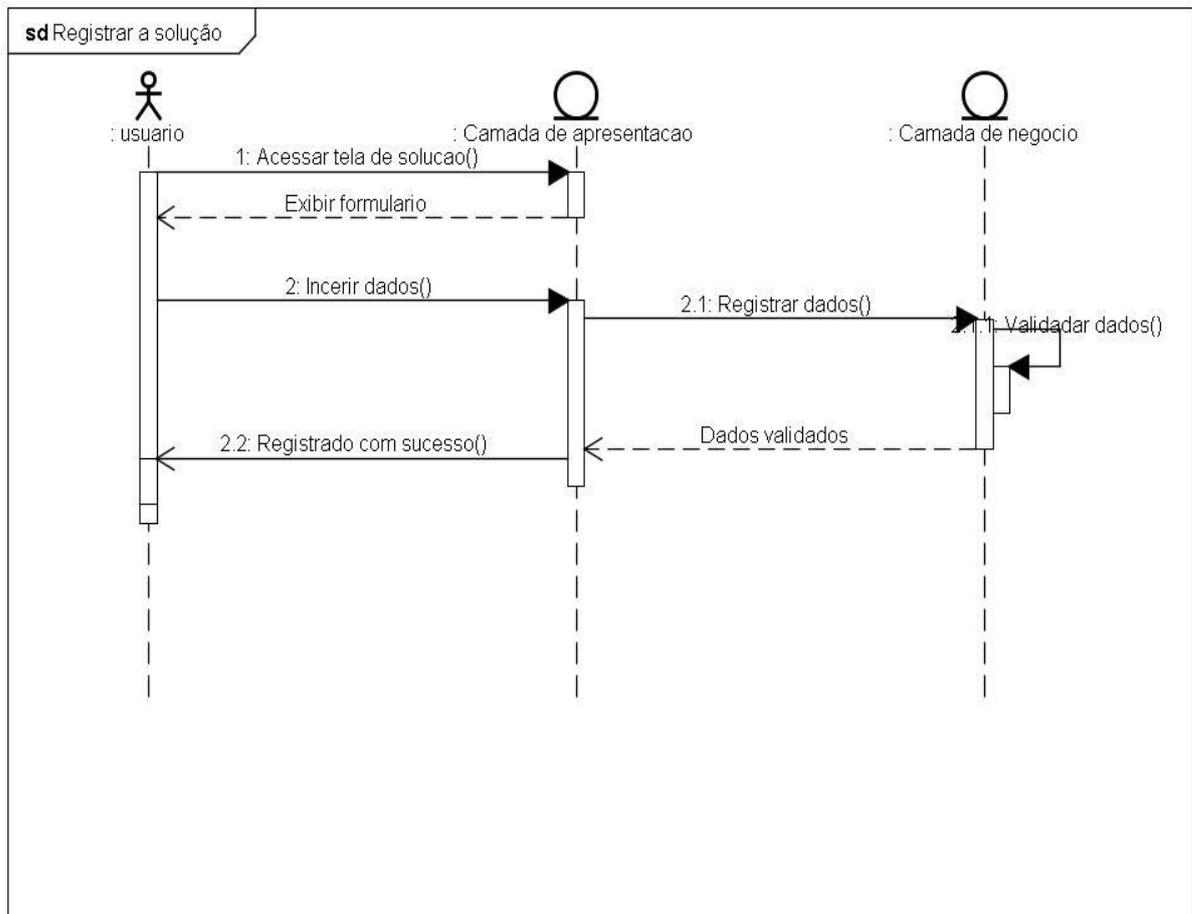


Figura 9 – Diagrama de Sequência – registrar a solução

2.6.7. TELAS DO PROTÓTIPO



The image shows a mobile application login screen. At the top, there is a status bar with a signal strength indicator, the carrier name 'Bell', the time '4:20 PM', and icons for location, lock, and battery. The main content area has a light purple background. The word 'Login' is centered at the top in a large, black, sans-serif font. Below it, there are two white input fields. The first is labeled 'Usuário:' and the second is labeled 'Senha:'. At the bottom center, there is a grey button with the text 'Cadastrar'. Below the button, the text 'Cadastrar usuário e senha' is displayed in a smaller, black, sans-serif font.

Figura 10- login

Bell 4:20 PM

Cadastrar usuário

Nome:

Sexo:

E-mail:

Figura 11 – cadastrar usuário



Figura 12 – tela de inicio

Bell 4:20 PM

Registrar solicitação

Categoria: troca de lâmpada

Descrição

Editar Cadastrar

Figura 13 – registrar solicitação

Bell 4:20 PM

Cadastrar categoria

- Limpeza de rua
- Troca de lâmpada
- Tapa buraco
- Limpeza de lote baldio

Descrição

Figura 14 – cadastrar categoria

Bell 4:20 PM

Registrar solução

- Em andamento
- Realizado com sucesso
- Não realizado

Descrição

Figura 15 – registrar solução

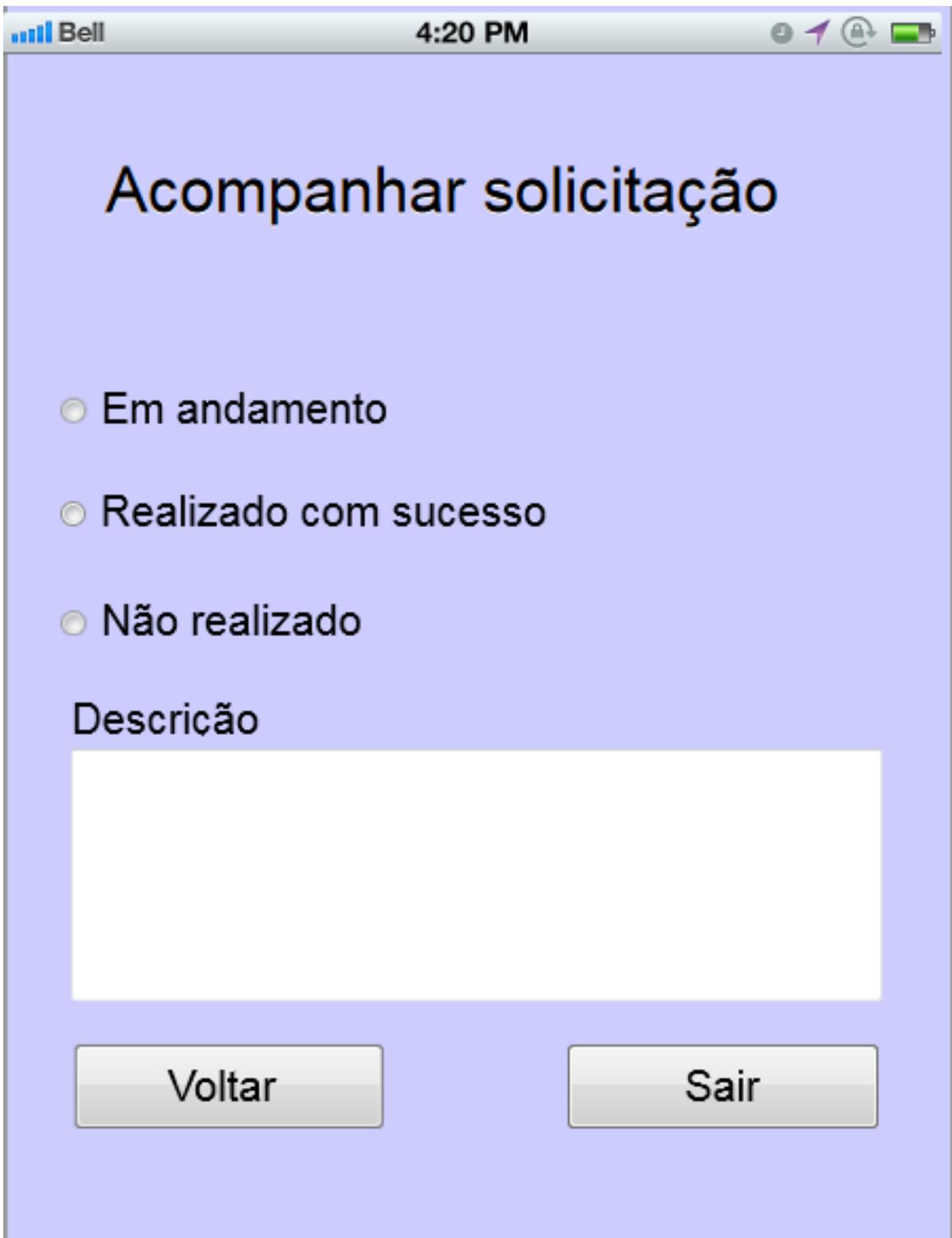


Figura 16 – Acompanhar solicitação

2.6.8. CRONOGRAMA E LISTA DE REQUISITOS

Atividades	Meses – Ano 2017										
	Fev.	Mar.	Abri.	Mai.	Jun.	Jul.	Ago.	Set.	Out.	Nov.	Dez.
Entrega da proposta		X									
Coleta de dados		X	X	X	X	X	X				
Referencial teórico			X	X	X	X	X				
Análise					X	X	X	X			
Modelagem					X	X	X	X			
Prototipação							X	X	X		
Apresentação da Pré-qualificação					X						
Revisão do trabalho escrito									X	X	
Apresentação da qualificação											X

Figura 17 - Cronograma

Nº	Interesse	Motivo	Solicitação	Data	Ator
R1	Cadastrar-se	Nome, endereço, contatos...	Funcionário/cidadão	05/03/2017	Funcionário/cidadão
R2	Acessar o sistema	o cidadão pode acessar o sistema para solicitar um serviço a gestão ou acompanhar a solicitação e o funcionario pode acessar para registrar as solicitações, encaminhar as order de serviços, enviar uma posição para o cidadão e etc.	Funcionário/cidadão	05/03/2017	Funcionário/cidadão
R3	Categoria	Escolher a categoria de serviço ou solicitação	Cidadão	05/03/2017	Cidadão
R4	Registrar Solicitação	Solicitar um serviço a gestão	Cidadão	05/03/2017	Cidadão
R6	Acompanhar solicitação	Acompanhar a solicitação feita pelo o cidadão na aplicação	Funcionário	05/03/2017	Funcionário
R7	Registrar a solução	Manter informações cadastrais das solicitações feita pelos servidores.	Funcionário/cidadão	05/03/2017	Funcionário/cidadão

Figura 18–Lista de requisitos

2.7. CONSIDERAÇÕES FINAIS

Apresentou-se neste projeto as etapas básicas desde o começo do desenvolvimento de um protótipo que dado por nome SGP (Sistema de Gestão Participativa). Viu-se a análise de requisitos, engenharia de requisitos, diagramas, banco de dados, entre outros. Por usar as técnicas mais conhecidas e padrões, tem-se uma ampla área de pesquisa disponível para que possa ser estudado o caso escolhido.

Percebe-se que a pesquisa de campo e entrevistas contribuíram e muito para a determinação da problemática que é o passo inicial deste processo. A mesma mostra a dificuldades que a população de Itaberaí, vem enfrentando no momento de solicitar um serviço a gestão.

Nesse sentido, a utilização de recursos digitais permite que a população solicite uma manutenção na cidade com rapidez e facilidade, e permite a gestão realizar seu trabalho de forma mais rápida e eficiente. Além disso, diminui o tempo de espera da população. Motivando as duas partes envolvidas no evento.

2.8. REFERENCIAS BIBLIOGRAFICAS

BOOCH Grady, RUMBAUCH James, JACOBSON Ivar. UML Guia do Usuário. Editora Elsevier. Rio de Janeiro, 2005.

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. Editora Elsevier Rio de Janeiro, 2007.

FARIA, THIAGO. Java EE 7 com JSF, Prime Faces e CDI. Alga Works Softwares, Treinamentos e Serviços Ltda, 2013.

GUEDES, Gilleanes T. A. UML 2: uma abordagem prática. São Paulo: Novatec Editora, 2011.

Gomes, Helton Simões, Do G1, em São Paulo Prefeituras começam a usar app para receber reclamação de cidadãos. G1. São Paulo, 09/04/2014. <<http://g1.globo.com/tecnologia/noticia/2014/04/prefeituras-comecam-usar-app-para-receber-reclamacao-de-cidadaos.html>> Acesso em: 30 out. 17.

Pressman. DR. Roger. S. Engenharia de Software editora Pearson. São Paulo 1995

PFLEEGER, SHARI LAWRENCE. Engenharia de software: teoria e prática. Editora Prentice Hall. São Paulo, 2004.

POWACZUK, Lucas, Gestão de entidades em cidades digitais: identificação e responsabilização legal dos usuários. Santa Maria, RS, Brasil 2013.

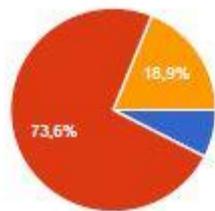
SOMMERVILLE, Ian. Engenharia de Software 8^o edição editora Pearson, São Paulo 2010

Sarrafho, Gilson, em APLICATIVO ALÔ PREFEITO SERÁ MAIS UM CANAL DE COMUNICAÇÃO ENTRE A POPULAÇÃO E A PREFEITURA DE IBAITI. Ibaíti/PR 2017. <[Http://www.ibaiti.pr.gov.br/Aplicativo-Alo-Prefeito-sera-mais-um-canal-de-comunicacao-entre-populacao-prefeitura-de-Ibaiti/](http://www.ibaiti.pr.gov.br/Aplicativo-Alo-Prefeito-sera-mais-um-canal-de-comunicacao-entre-populacao-prefeitura-de-Ibaiti/)> Acesso em: 07 nov. 17.

TOBY, TEOREY, SAM LIGHTSTONE, TOM NADEAU. Projeto de Modelagem e Banco de Dados Editora Elsevier. Rio de Janeiro, 2007.

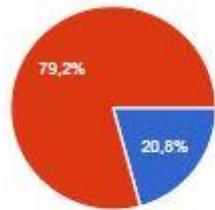
2.9. ANEXOS

1. Qual a sua idade?



Menos de 20 anos	4	7.5%
Entre 20 e 30 anos	39	73.6%
Entre 31 e 50 anos	10	18.9%
Mais de 50 anos	0	0%

2. Você está satisfeito com os serviços prestados pela prefeitura?



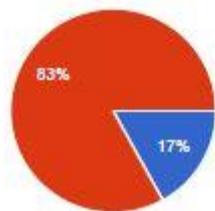
Sim	11	20.8%
Não	42	79.2%

3. Você tem dificuldade para solicitar um serviço da prefeitura? (serviços como: iluminação, tapa buracos, limpeza, roçagem, etc)?



Sim	48	90.6%
Não	5	9.4%

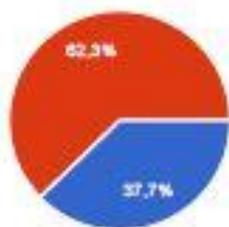
4. Existem alguma previsão de execução dos serviços por parte da prefeitura? A prefeitura informa se o serviço solicitado foi realizado?



Sim	9	17%
Não	44	83%

Figura 19 - entrevista parte 1

5. A prefeitura oferece algum canal de comunicação com a população?



Sim	20	37.7%
Não	33	62.3%

6. Você acessa a Internet com frequência?



Sim	53	100%
Não	0	0%

7. Você possui um telefone celular com acesso a Internet? (smartphone)



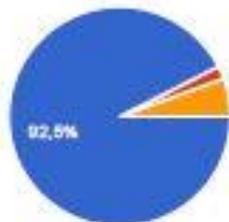
Sim	53	100%
Não	0	0%

8. Você acredita que uma aplicação de celular ou um site facilitaria a comunicação com a gestão municipal para solicitação de algum tipo de serviço?



Sim	48	90.6%
Não	2	3.8%
Não sei	3	5.7%

9. Você usaria esse tipo de aplicativo para solicitar serviços para a prefeitura?



Sim	49	92.5%
Não	1	1.9%
Não sei	3	5.7%

Figura 20 - entrevista parte 2