UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

SILVIO POLLI GHEDIN

VALIDAÇÃO DE REQUISITOS NO MODELO DE PROTOTIPAÇÃO EM PROJETOS WEB: CONSTRUÇÃO DE UMA FERRAMENTA

CRICIÚMA
2016

SILVIO POLLI GHEDIN

VALIDAÇÃO DE REQUISITOS NO MODELO DE PROTOTIPAÇÃO EM PROJETOS WEB: CONSTRUÇÃO DE UMA FERRAMENTA

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação, da Universidade do Extremo Sul Catarinense, UNESC, com linha de pesquisa em Desenvolvimento de Software Web.

Orientador: Prof. *Esp*. Gilberto Vieira da Silva

CRICIÚMA

2016

SILVIO POLLI GHEDIN

VALIDAÇÃO DE REQUISITOS NO MODELO DE PROTOTIPAÇÃO EM PROJETOS WEB: CONSTRUÇÃO DE UMA FERRAMENTA

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Desenvolvimento Web.

Criciúma, 28 de novembro de 2016.

BANCA EXAMINADORA

Prof. Esp. Gilberto Vieira da Silva - UNESC - Orientador

Prof. MSc. Ana Claudia Garcia Barbosa - UNESC

Prof. Esp. Fabrício Giordani - UNESC

Dedico este projeto a todos aqueles que de alguma forma estiveram próximos de mim e me motivaram a continuar.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me dar o dom da vida e conceder a oportunidade de realizar este sonho de desenvolver este trabalho.

Agradeço minha família, em especial aos meus pais, Silvio e Jaqueline, por sempre se esforçarem para me proporcionarem o melhor e também a todos os demais membros que estiveram presente.

Agradeço ao meu namorado Ramires e a nossa gata Dina pela compreensão dos momentos difíceis e pelo carinho que sempre me proporcionaram quando eu precisei.

Agradeço aos colegas de profissão Wellington Knabbenn, Paulo Roberto, Tiago Tessmann e Ramon Tessmann por fazer meu trabalho melhor e consequentemente me ajudaram a render mais.

Agradeço ao meu orientador Gilberto Vieira da Silva **que sempre me apoiou e sempre me deu forças para continuar e fez um papel fundamental no desenvolvimento deste trabalho.**

Aos demais colegas do curso de Ciência da Computação, que partilharam de momentos únicos nestes quatro anos e meio, também um salve para meu amigo Marlon Zilli que sempre esteve pronto para me ajudar quando eu precisei.

Meus amigos em geral, que indiretamente e diretamente fizeram parte desta história.

RESUMO

Com a popularização da internet e com o crescimento dos usuários que passaram a utiliza-la, grandes empresas e softwares viram a oportunidade e necessidade de migrarem para a Web. Os principais motivos de insucesso e comprometimento dos projetos são: falta de envolvimento do usuário, mudanças nos requisitos ou problemas na definição. Diante desses problemas ainda vigentes, foi criada uma ferramenta. Durante a fase de prototipação, a ferramenta criada é utilizada e tem como objetivo validar requisitos e proporcionar uma melhor interação entre cliente e desenvolvedor. A ferramenta foi utilizada em um ambiente real, na empresa Blueberry - Agência de Marketing Digital, permitindo a validação dos requisitos com clientes reais e melhorando a comunicação com eles. Os resultados obtidos no desenvolvimento foram satisfatórios e houve uma melhora de até 40% na finalização dos projetos desenvolvidos.

Palavras-chave: Requisitos, Desenvolvimento Web, Prototipação, Engenharia de Software

ABSTRACT

With the polarization of Internet and the growth of users using it, big companies and softwares saw both the opportunity and the need to migrate to the Web. The main factors of failure and commitment in these projects are: lack of user involvement, changes in the requisites or problems in definition. A tool was created to meet the needs of these current problems. During the prototyping, this new tool is used and aims at validating requisites and provide a better interaction between the cliente and the developer. The tool was used in a real environment at Blueberry – Agência de Marketing Digital, allowing such requisites to be validated with real clientes and improving communication with them. The results gathered in the development were satisfactory and there was an improvement of up to 40% in the completion of the projects developed.

Keywords: Requisites, Web Development, Prototyping, Software Engineering

LISTA DE ILUSTRAÇÕES

Figura 1 – Ciclo de vida de software 18
Figura 2 – Disciplinas que englobam o IHC 21
Figura 3 – Exemplo de um *storyboard* 23
Figura 4 – Exemplo de um *wireframe* 25
Figura 5 – Processo de desenvolvimento de protótipo 26
Figura 6 – O processo da e-prototipagem como parte do ciclo de desenvolvimento 29
Figura 7 – Requisitos não articulados 32
Figura 8 – Uma visão espiral do processo da engenharia de requisitos 34
Figura 9 – XWebProcess sob ponto de vista dinâmico 40
Figura 10 – Sublime Text 2 52
Figura 11 – Filezilla 53
Figura 12 – Diagrama de casos de uso 54
Figura 13 – Modelagem do banco 54
Figura 14 – Modelagem conceitual 55
Figura 15 – Estrutura do servidor web 56
Figura 16 – Comunicação entre servidor e banco de dados 57
Figura 17 – Fluxograma da aplicação 57
Figura 18 – Tela de Login 58
Figura 19 – Ajax chamando a tela de login 58
Figura 20 – Parâmetros do login 59
Figura 21 – Campos de selecionar no modal de criação de novo projeto 60
Figura 22 – Ajax de cadastro de imagem 60
Figura 23 – Cadastro de projeto e modal de envio de e-mail 62
Figura 24 – Envio de e-mail para o cliente 63
Figura 25 – Salvar comentários no banco de dados 63
Figura 26 – Carregamento dos comentários 64
Figura 27 – Carregamento dos comentários 02 64
Figura 28 – Upload no cadastro de usuário 65
Figura 29 – Cadastro de usuário 65
Figura 30 – Cadastro de cliente 66
Figura 31 – Endereçamento das imagens 67
Figura 32 – Tela de login do protótipo 68

Figura 33 – Modal de cadastro de novo usuário 68
Figura 34 – Modal de cadastro de novo cliente.....69
Figura 35 – Modal de cadastro de novo projeto 69
Figura 36 – Tela de projeto 70
Figura 37 – Fazer comentário 71
Figura 38 – Finalizar projeto.....71
Figura 39 – Modal de cadastro de novo imagem 72
Figura 40 – Modal de cadastro de novo imagem 73

LISTA DE TABELAS

Tabela 1 - Princípios dos métodos ágeis. 19
Tabela 2 - Vantagens e desvantagens dos protótipos de baixa e alta fidelidade. 24
Tabela 3 - Técnicas para levantamento de requisitos. 35

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CASE	*Computer-aided software engineering*
CSS	*Cascading Style Sheets*
ES	Engenharia de Software
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IHC	Interação Homem Computador
PHP	*Hypertext Preprocessor*
PMI	*Project Management Institute*
WEB	*World Wide Web*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Objetivo Geral 13
1.2 Objetivos Específicos 14
1.3 Justificativa 14
1.4 Estrutura do Trabalho 15
2 ENGENHARIA DE SOFTWARE 17
2.1 Software 17
2.2 Ciclo de Vida do Software 17
2.3 Metodologias de Desenvolvimento de Software 18
2.4 Design Rationale 19
2.5 Interação Homem Computador (IHC) 20
2.6 Prototipação 22
2.6.1 Prototipação de Baixa Fidelidade 22
2.6.2 Prototipação de Alta Fidelidade 23
2.6.3 A importância de utilizar protótipos 25
2.6.4 Abordagens de prototipação 27
2.6.5 Prototipagem baseado na Web 28
3 ENGENHARIA DE REQUISITOS 31
3.1 Estabelecendo Requisitos 31
3.2 Conceito de Requisito 33
3.3 Requisitos Funcionais e Não Funcionais 33
3.4 O Processo de Engenharia de Requisitos 34
3.4.1 Levantamento de Requisitos 35
3.4.2 Análise de requisitos 36
3.4.3 Validação de requisitos 36
3.5 Gerenciamento de Requisitos 37
3.6 Processo de Desenvolvimento Web 38
3.6.1 XWebProcess 39
3.6.2 Modelo WebML 41
4 TECNOLOGIAS PARA DESENVOLVIMENTO WEB 43
4.1 Vantagens das Aplicações Web 43
4.2 HyperText Markup Language (HTML) 44
4.3 Cascading Style Sheets (CSS) 45

4.4 PHP 46
4.5 JavaScript em Aplicações Web 46
5 TRABALHOS CORRELATOS 48
5.1 METODOLOGIA PARA EQUIPES DE DESENVOLVIMENTO DE REQUISITOS DE SISTEMAS DE INFORMAÇÃO 48
5.2 UMA ABORDAGEM SISTÊMICA PARA O PROCESSO DE PRODUÇÃO EM ENGENHARIA WEB, NA FASE DE CONCEPÇÃO 48
5.3 AS COMPETÊNCIAS DA EQUIPE DE PROJETO NO PROCESSO DE DESENVOLVIMENTO DE APLICAÇÕES WEB 49
5.4 ELABORAÇÃO DE UMA PLATAFORMA PARA METANÁLISE DIAGNÓSTICA UTILIZANDO CONCEITOS E ANÁLISE DE REQUISITOS DA ENGENHARIA DE SOFTWARE 49
6 FERRAMENTA PARA VALIDAÇÃO DE REQUISITOS NO MODELO DE PROTOTIPAÇÃO EM PROJETOS WEB 51
6.1 METODOLOGIA 51
6.2 FERRAMENTAS UTILIZADAS 52
6.3 MODELAGEM DE DADOS 53
6.4 DESENVOLVIMENTO DA FERRAMENTA 55
6.4.1 Estrutura do servidor web 56
6.4.2 Aplicação web 57
6.4.3 Funcionamento do sistema 67
6.5 RESULTADOS OBTIDOS 72
7 CONCLUSÃO 75

1 INTRODUÇÃO

A crise do software, termo utilizado desde o final dos anos 1960, expressa as dificuldades do desenvolvimento de software frente à complexidade dos problemas a serem resolvidos, crescente demanda de sistemas computacionais e o uso de técnicas e métodos adequados na sua implementação (SOMMERVILLE, 2008). Os efeitos da crise se manifestam em diferentes situações, dentre elas cita-se o não cumprimento de prazo e orçamento; produção de softwares de baixa qualidade; ingerência em projetos de software; não utilização de padrões de desenvolvimento; e não cumprimento de requisitos.

Para superar a “crise do software”, a Engenharia de Software (ES) tem proposto caminhos de solução que envolvem ferramentas, metodologias, padrões, artefatos, entre outros recursos para dar apoio ao processo de desenvolvimento, avaliação, implantação e manutenção de softwares. Apesar dos esforços da ES, muitos projetos de software apresentam as situações citadas ainda na atualidade (PRESSMAN, 2006).

A confirmação dessa afirmação vem dos dados do estudo intitulado de *Chaos Research* (STANDISH, 2015), realizado desde 1994 pela *Standish Group International*, que relata o sucesso e o fracasso de projetos na área da Tecnologia da Informação (TI). No relatório *Chaos Research* os projetos são enquadrados em três categorias distintas: mal sucedido (o projeto é cancelado em algum momento do desenvolvimento por uma ou mais razões); bem-sucedido (o projeto é concluído dentro do prazo previsto e do orçamento estimado); e comprometido (o Projeto é concluído, porém entregue com atraso, com orçamento além do estimado, e em alguns casos, o software não possui todas as funcionalidades especificadas).

Os resultados do relatório *Chaos Research* de 2015 apontam que dos projetos de desenvolvimento de software de empresas e agências governamentais dos Estados Unidos apenas 16,2% foram bem-sucedidos, 52,7% foram classificados como comprometidos e 31.1% dos projetos foram considerados malsucedidos e consequentemente cancelados. Dentre os motivos referentes à classificação de comprometimento do projeto em relação ao prazo, orçamento e não alinhamento dos resultados do software com as necessidades dos usuários, no relatório são citados: falta de envolvimento do usuário (12.8%); requisitos e especificações incompletos

(12.3%); mudanças em requisitos e especificações (11.8%); falta de apoio executivo (7.5%); incompetência/imaturidade tecnológica (7.0%); falta de recursos (6.4%); expectativas não realistas (5.9%); falta de clareza nos objetivos (5.3%); tempo estimado não realista (4.3%); novas tecnologias (3.7%); outros (23.0%).

Portanto, é possível dizer que a crise de software continua vigente ainda na atualidade sendo um tema importante e recorrente em pesquisas tanto científicas, na área de Engenharia de Software, como tecnológicas, aplicadas em empresas de desenvolvimento de software (PRESSMAN, 2011). Apesar dos dados do relatório Chaos Research serem provenientes dos Estados Unidos, pode-se projetar este resultado para cenários similares onde tem-se o problema de comprometimento ou insucesso em projetos de software.

Considerando que uma das razões, citada pelo PMI (2013), para que um projeto seja classificado como comprometido é a não entrega dos resultados alinhados com as necessidades dos usuários, pode-se perceber a relação direta com os três primeiros motivos relatados sendo os que mais afetam o comprometimento: falta de envolvimento do usuário, problemas na definição ou mudanças em requisitos e especificações.

Deficiências no alinhamento entre as necessidades do usuário e o resultado do software podem ser decorrentes do pouco tempo dedicado ao levantamento de dados, falta de método ou ainda não adoção de estratégias de gestão de projeto e acompanhamento das expectativas dos interessados (*stakeholders*). A etapa de levantamento de requisitos consome tempo, e por consequência, recursos humanos e financeiros (MENEZES, 2001).

Há uma tendência na abreviação ou descuido nessa etapa, comprometendo a qualidade do trabalho como um todo (REZENDE, 2005). As alterações de requisitos são frequentes durante o projeto, incluindo novas possibilidades ou alterando as existentes. A falta de registro formal dessas alterações, que considere técnicas e métodos de ES e de gerenciamento de projeto, impede o acompanhamento correto e a visualização de impacto das alterações no cronograma e o comprometimento no prazo de entrega (PRESSMAN, 2011). Além disso, algumas alterações podem modificar o resultado final impactando no objetivo e atendimento de expectativa do cliente/usuário.

Problemas na etapa de levantamento de requisitos comprometem os resultados de um projeto gerando um escopo mal definido e muitas vezes sem a validação necessária do cliente/usuário (PMI, 2013). Diante do cenário apresentado, esta pesquisa visa estudar técnicas e métodos de engenharia de software e de gerenciamento de projeto que embasam o levantamento de requisitos de aplicações Web e o processo de comunicação entre cliente/usuário e equipe de desenvolvimento com propósito de validação e aprovação dos requisitos e do escopo de um projeto.

A partir do estudo será desenvolvida uma ferramenta para gerenciar o processo de comunicação entre cliente/usuário e equipe de desenvolvimento a partir do registro formalizado dos requisitos de um projeto Web, suas alterações e respectivas aprovações e reprovações do cliente/usuário. Com os registros visa-se a geração de um histórico que poderá ser utilizado para acompanhar a evolução, as alterações, as aprovações e reprovações, além de possibilitar informações para identificação dos impactos desses registros no sucesso, comprometimento ou insucesso do projeto.

O foco da proposição da ferramenta é o gerenciamento do processo de comunicação dos requisitos e escopo do projeto Web entre cliente/usuário e equipe de desenvolvimento. Nesse trabalho este processo é identificado como *feedback*, pois visa exprimir os registros de alimentação de retorno nas comunicações. Para validação da ferramenta será realizado um estudo de caso abrangendo um projeto de aplicativo Web e uma equipe de desenvolvimento.

Como resultado de aplicação da ferramenta, busca-se na perspectiva do cliente/usuário o atendimento de suas necessidades e expectativas contribuindo para a rápida identificação e resolução de questões e descontentamentos de forma a potenciar o seu envolvimento positivo no decurso da realização do projeto. Já na perspectiva do desenvolvedor, o propósito da ferramenta é dar suporte ao processo de comunicação com o cliente/usuário a partir do registro e acompanhamento da etapa de levantamento de requisitos e escopo com vistas ao sucesso do projeto.

1.1 OBJETIVO GERAL

Realizar a validação de requisitos em projetos Web a partir de técnicas de engenharia de software, visando o desenvolvimento de uma ferramenta para gestão de *feedback* entre cliente e desenvolvedor.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) compreender a validação de requisitos em projetos Web;
b) demonstrar técnicas de levantamento de requisitos para desenvolvimento Web;
c) desenvolver um protótipo para gestão de *feedback* visando a validação de requisitos;
d) validar a prototipação em um estudo de caso a partir do desenvolvimento de um projeto Web.

1.3 JUSTIFICATIVA

O avanço e o constante desenvolvimento da Internet e dos projetos que são criados especificamente para Web, tem modificado a forma de interação com as pessoas e os meios digitais. Nosso mundo mudou e com isso nossa forma de trabalhar também mudou. Há diversas formas de encantar e mudar a forma com que as pessoas interagem com o mundo e consigo (LOWDERMILK, 2013).

Conforme Norman (2013), mecanismos e formas para coletar *feedback* do usuário raramente existem. A busca incansável na tentativa de desenvolver algo melhor ao que foi desenvolvido previamente, na grande maioria das vezes acaba sendo desastrosa para o consumidor. Sendo assim há um insucesso para todos os envolvidos, tanto para a equipe que despendeu tempo e dinheiro para o desenvolvimento, quanto para o cliente que não teve suas necessidades atendidas.

Atualmente as empresas operam em um ambiente global que muda constantemente. Aplicativos e softwares geralmente fazem parte de todas as operações de negócios, logo os novos softwares são feitos rapidamente para competir

com o mercado e aproveitar as novas oportunidades. O desenvolvimento e a entrega rápida de um projeto, são os requisitos mais críticos para o desenvolvimento.

Grande parte dos softwares e websites não atendem as necessidades dos clientes, o que acarreta em uma alta taxa de rejeição. Segundo Unger e Chandler (2009), estes problemas ocorrem devido a particularidade e singularidade de cada projeto. É imprescindível entender as funcionalidades e características, para entregar um produto de alta qualidade. Para atingir estes objetivos é necessário realizar o levantamento de requisitos junto ao cliente, porém, após o seu entendimento é necessário realizar uma validação prévia para identificar possíveis divergências. O levantamento de requisitos é um padrão de tarefa que é importante para a prática bem-sucedida da engenharia de software, que prevê um processo completo envolvendo prototipação, planejamento e estimativa, como é descrito por (PRESSMAN, 2006).

Segundo Preece (2013, p. 355) "[...] um requisito é uma declaração sobre um produto pretendido que especifica o que ele deveria fazer ou como deveria funcionar." A principal função dos requisitos é esclarecer os principais pontos e identificar características e especificações. Após os requisitos serem estabelecidos, iniciam-se as atividades do desenvolvimento. Para o usuário avaliar e dar um *feedback* eficiente, deve-se criar um protótipo que faz parte do estágio inicial do desenvolvimento do projeto.

Pretende-se desenvolver uma ferramenta utilizando as técnicas e metodologias de gerenciamento de projeto e engenharia de software visando validar requisitos entre cliente/usuário e equipe de desenvolvimento. Esta ferramenta irá auxiliar no processo de desenvolvimento do projeto Web e a demonstrar com mais clareza as necessidades do projeto, assim, evitando problemas com a falta de formalização e divergência com o cliente/usuário.

1.4 ESTRUTURA DO TRABALHO

Este trabalho de pesquisa é constituído por seis capítulos, sendo que o primeiro capítulo aborda uma introdução sobre o tema da pesquisa, e sua estrutura é formada pela introdução, objetivo geral e seus objetivos específicos, e da justificativa para o qual o projeto será desenvolvido.

O segundo capítulo trata sobre a Engenharia de Software e faz uma introdução sobre metodologias e conceitos de software. Também há informações sobre Prototipagem que vai ser um dos objetos utilizados nesse projeto proposto.

No terceiro capítulo foi abordado sobre a engenharia de requisitos e a importância de fazer com cuidado essa parte de descobrir, analisar, documentar, gerenciar e controlar a qualidade dos requisitos.

No capítulo quatro é abordado o desenvolvimento Web. As vantagens e tecnologias utilizadas para o desenvolvimento Web.

O capítulo cinco trata dos trabalhos relacionados a este projeto de pesquisa, que abordam, indiretamente ou diretamente alguns objetivos semelhantes a este projeto. É possível observar que a validação de requisitos é um processo difícil de ser executado, mas fazendo com cuidado o processo pode ter êxito.

O capítulo seis, é mostrado os passos realizados durante a elaboração do trabalho proposto e a metodologia utilizada para o seu desenvolvimento, também será informado os recursos necessários para o progresso do mesmo. Finalizando com o capítulo sete, traz as conclusões obtidas com este trabalho e sugestões para possíveis aprimoramentos para dar continuidade a este estudo.

2 ENGENHARIA DE SOFTWARE

A Engenharia de software fundamenta-se em uma disciplina que compõe de três camadas: processos, métodos e ferramentas. A qualidade no desenvolvimento de software é a base das camadas que são conduzidas durante as etapas de fabricação. Independente da área de aplicação ou complexidade do projeto, no processo de desenvolvimento de software, há três fases genéricas, as quais são: definição, desenvolvimento e manutenção (PRESSMANN, 2010).

Segundo Sommerville (2004), a engenharia de software é uma disciplina que pertence a área da engenharia e é responsável por todos os aspectos do desenvolvimento de software e está presente desde os estágios iniciais de especificação, até no final do ciclo na fase de manutenção.

2.1 SOFTWARE

O conceito de software para grande parte das pessoas é apenas um programa de computador. O software não consiste em ser apenas um programa em si, mas também de toda a documentação e configuração que são necessárias para que atenda às necessidades dos usuários e opere corretamente (SOMMERVILLE, 2004).

Pressmann (2010), define o software como um elemento de sistema lógico e não físico. Para ter uma boa compreensão do que é um software, as características devem ser examinadas, o que irá diferenciar das outras coisas que são construídas pelo ser humano.

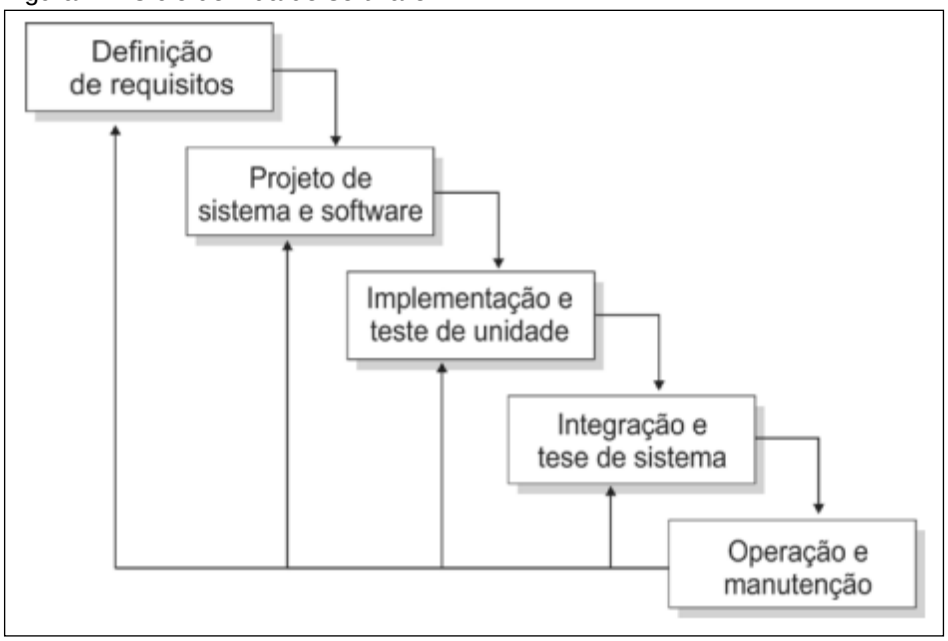
2.2 CICLO DE VIDA DO SOFTWARE

Todo software possui um ciclo de vida, geralmente ele é criado a partir de uma necessidade; desenvolvido e entregue ao cliente e usuário; entra em operação com sujeição a mudanças e manutenções e finalmente retirado de operação no final de sua vida útil (PAULA FILHO, 2000).

O ciclo de vida clássico da engenharia de software é mais conhecido como modelo cascata. As fases seguintes nesse modelo, apenas ocorrem quando a fase

anterior foi finalizada e o ciclo de vida do software é mostrado através deste ciclo (figura 1).

Figura 1 – Ciclo de vida de software



Fonte: Sommerville (2008).

Segundo Pressmann (2010), o software percorre por diversas fases desde o início, até o desenvolvimento e implementação. O ciclo de vida requer uma abordagem sequencial, sistemática ao desenvolvimento e é considerado uma sequência de tarefas e fases que são feitas no decorrer do projeto.

2.3 METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

Há diversos tipos de métodos, o que diferencia cada um é o quanto o cliente irá participar do projeto e a quantidade de documentação produzida. Independente da metodologia escolhida, grande parte das atividades envolvidas no processo de desenvolvimento do software serão comuns entre si.

Na década de 1980, a maneira mais correta de desenvolvimento de software era através de um planejamento cuidadoso, qualidade da segurança formalizada, uso de ferramentas CASE e um processo rigoroso e controlado no desenvolvimento do software. Alguns anos depois, devido a insatisfação dos desenvolvedores de software, começou a surgir os métodos ágeis, onde o foco era o software e não mais a concepção e documentação (SOMMERVILLE, 2008).

Existem inúmeras metodologias ágeis, a mais conhecida provavelmente é a *Extreme Programming* (BECK, 1999; BECK, 2000). Apesar disso há outras opções,

como o Crystal (COCKBURN, 2001; COCKBURN, 2004), *Scrum* (COHN, 2009; SCHWABER, 2004; SCHWABER; BEEDLE, 2001), DSDM (STAPLETON, 1997; STAPLETON, 2003), Desenvolvimento de Software Adaptativo (HIGHSMITH, 2000) e Desenvolvimento Dirigido a Características (PALMER; FELSING, 2002).

Apesar de existirem diversos tipos de métodos, eles compartilham do mesmo objetivo e princípio, a tabela 1 contém os princípios dos métodos ágeis.

Tabela 1 - Princípios dos métodos ágeis.

Princípios	**Descrição**
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.
Entrega incremental	O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar as mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso, projete o sistema de maneira a acomodar essas mudanças.
Manter a simplicidade	Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

Fonte: Sommerville (2008).

Segundo Pressmann (2010), quatro tópicos são essenciais para a filosofia ágil, dentre elas são: comunicação e colaboração entre os membros da equipe e entre os clientes; equipes com controle sobre o trabalho que realizam; destaque na entrega de software que deixe o cliente satisfeito; e o reconhecimento que as oportunidades podem estar nas modificações solicitadas.

2.4 DESIGN RATIONALE

O *design rationale* se baseia nas decisões previamente tomadas durante o processo de desenvolvimento do *design* de um artefato. O *design rationale* inclui a descrição de um artefato, outras opções alternativas e razões que levaram as escolhas das soluções específicas para a produção do artefato (MACLEAN, 1989).

A utilização e registro do *design rationale* traz diversos benefícios para o projeto, a seguir os mais expressivos:

a) contribuir na resolução de problemas provendo informações sobre o raciocínio por trás do *design* (MACLEAN, 1989);

b) prever as consequências de alterações solicitadas ao *design* (MACLEAN, 1989; MACLEAN et al, 1996; FISCHER et al, 1996; GRUBER; RUSSELL, 1996);
c) detectar erros de *design*, provendo explicitamente uma descrição do raciocínio utilizado no processo de design (CONKLIN; YAKEMOVIC, 1996);
d) disponibilizar as decisões rejeitadas e aceitas, entendendo melhor os motivos das decisões e evitando o retrabalho (BURGE; BROWN, 2000);
e) deixar registrado o raciocínio utilizado para evitar uma possível perda, caso ocorra um desligamento do projetista na empresa (BURGE; BROWN, 2000; CONKLIN; YAKEMOVIC, 1996);
f) recuperar algumas decisões previamente tomadas, argumentos e suas justificativas. “Talvez, devido à fragilidade da memória humana, os projetistas necessitem de seus próprios registros de *design rationale* tanto quanto os responsáveis pela manutenção que trabalharão no sistema em um momento posterior.” (CONKLIN; YAKEMOVIC, 1996, p. 393).

As técnicas de *design* rationale evitam as repetições do trabalho feito anteriormente. Quando pessoas diferentes revisam o projeto, é possível identificar quais os motivos as decisões foram tomadas, alternativas consideradas e as rejeitadas (CAMPONOGARA, 2008).

2.5 INTERAÇÃO HOMEM COMPUTADOR (IHC)

O crescimento do número de pessoas que utilizam computadores para realizar diferentes tarefas tem crescido e na mesma proporção o interesse na Interação Humano-Computador cresce também.

A IHC define e estuda métodos para os dispositivos de interação ou projetos de sistema sejam mais eficientes, eficazes, fáceis na utilização e que seja confortável aos indivíduos que irão utilizá-los (AGNER, 2006).

Segundo Preece et al, (2005), a IHC preocupa-se com a avaliação, *design* e implementação de sistemas com foco na interação entre as pessoas, máquinas e com o ambiente ao redor. A principal função da IHC é desenvolver a interface com o

usuário, a interface apresenta estudos para a comunicação e interação entre o usuário, sistema e computador.

A IHC é uma ciência multidisciplinar que engloba diversas área, conforme a figura 2.

Figura 2 – Disciplinas que englobam o IHC

Fonte: Preece (2013).

O desenvolvedor tem a possibilidade de gerar interfaces de sistemas que sejam mais adaptadas aos usuários e de um modo em que eles realizem melhor suas tarefas, utilizando as métricas que são sugeridas pela área de IHC. Uma interface com uma interação difícil, irá resultar em uma desmotivação ao usuário e no final fazer com que ele procure outras alternativas (CYBIS, 2007).

Os métodos ágeis podem agregar valor para a IHC, pois tem como objetivo colaborar com o cliente através de pequenos ciclos de desenvolvimento de forma incremental e iterativa, para conseguir o *feedback* do cliente e assim corrigir o processo do desenvolvimento (ARMITAGE, 2004; BLOMKVIST, 2005).

As atividades de IHC de um modo geral envolvem uma análise atual de como está o projeto, a identificação dos pontos que podem ser melhorados, a avaliação e criação de uma intervenção nesta situação. Essas atividades são organizadas de maneiras diferentes, prescrevem ou sugerem os artefatos utilizados

em cada uma delas, e sugerem como cada uma será executada (BARBOSA; SILVA, 2010).

2.6 PROTOTIPAÇÃO

O protótipo pode ser considerado a primeira versão inicial de um sistema de software. O mesmo é utilizado para demonstrar conceitos, experimentar opções de projeto e encontrar mais sobre possíveis falhas e soluções. É essencial que o desenvolvimento do protótipo seja rápido para que os custos fiquem controlados e os *stakeholders* conseguiam testar no começo do processo.

Nos processos de desenvolvimento de software, os protótipos podem ajudar a antecipar as mudanças requisitas: Um protótipo pode auxiliar na elicitação e validação dos requisitos de sistema no processo da engenharia de requisitos; um protótipo pode ser utilizado nos processos de projeto de sistema, para apoiar o projeto de interface do usuário e estudar soluções específicas do software (SOMMERVILLE, 2011).

Um protótipo segundo Preece (2013), pode ser considerado diversas coisas, desde um *storyboard* feito de papel, até uma tela de software complexo. No nível de *design*, o protótipo permite que os *stakeholders* interajam com ele, possibilitando que o mesmo tenha mais experiência ao utilizar o software.

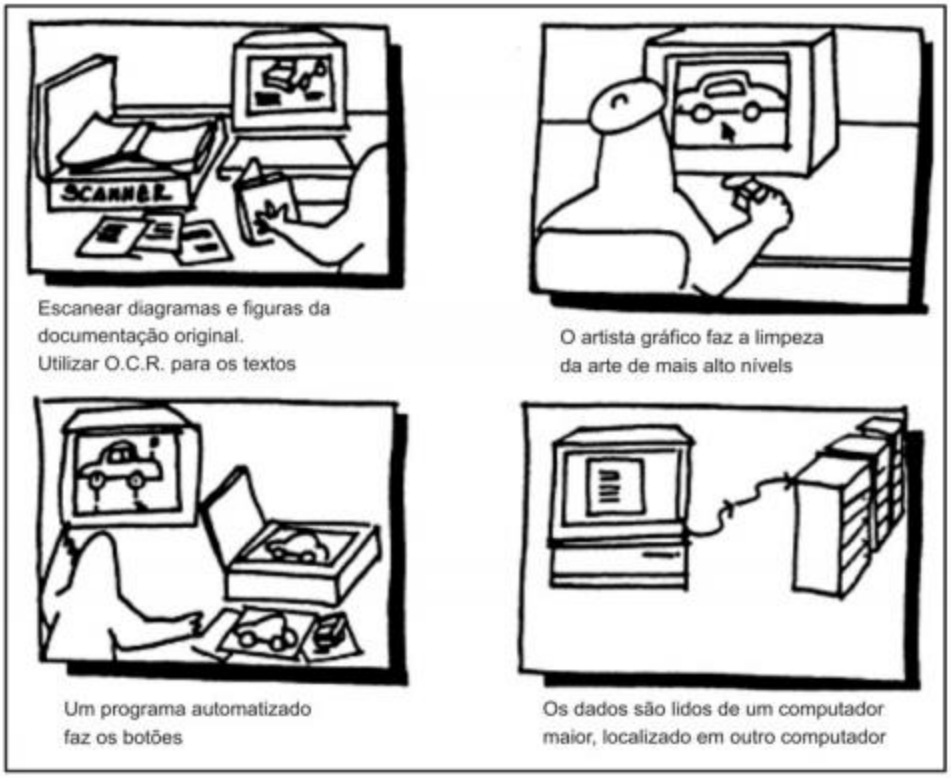
A prototipagem não é um artefato, é um processo iterativo. O resultado final do processo é obter um retorno acionável dos conceitos que podem ser utilizados para aprimorar o projeto (UNGER; CHANDLER, 2009).

2.6.1 Prototipação de Baixa Fidelidade

Os protótipos de baixa fidelidade são bem úteis, pois são baratos, simples e rápidos de se produzir. Além disso também são baratos, simples e rápidos para modifica-los, logo é possível ter mais ideias e alternativas. Os protótipos de baixa finalidade não têm como o objetivo de serem mantidos no produto final, eles não se parecem muito com o produto final, são apenas para a exploração inicial do projeto.

Na figura 3, mostra o *storyboard*, um modelo de baixa finalidade.

Figura 3 – Exemplo de um *storyboard*



Fonte: Winograd et al (1996).

O *storyboard* utiliza esboços em uma ordem sequencial e mostra ao usuário como ele pode progredir em uma determinada tarefa utilizando o produto em desenvolvimento. Na figura 3 mostra o indivíduo utilizando um novo sistema de digitalização de imagens e descreve os passos necessários em que o usuário poderia seguir (PREECE, 2013).

2.6.2 Prototipação de Alta Fidelidade

A prototipação de alta fidelidade, diferente da baixa fidelidade, usa os materiais que se esperam que estejam no produto final e constrói um protótipo mais parecido possível com ele.

A tabela 2 contém as vantagens e desvantagens dos tipos de prototipação (PREECE, 2013).

Tabela 2 - Vantagens e desvantagens dos protótipos de baixa e alta fidelidade.

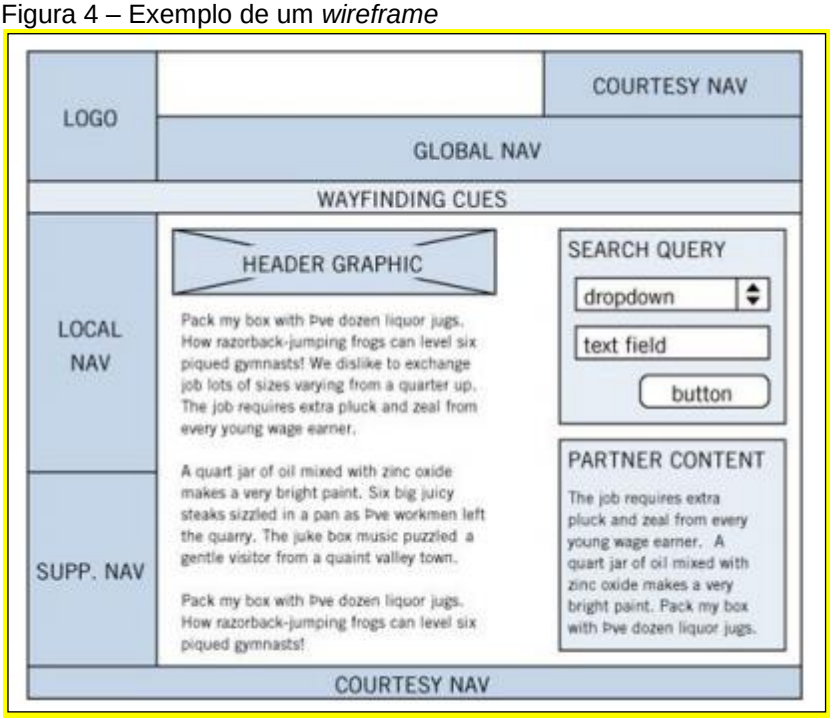
Tipo	Vantagens	Desvantagens
Protótipo de baixa fidelidade	• Custo mais baixo de desenvolvimento. • Avalia múltiplos conceitos de *design*. • Instrumento de comunicação útil. • Aborda questões de layout de tela. • Útil para identificação de requisitos de mercado. • *Proof-of-concept* (demonstrações de que o conceito funciona).	• Verificação limitada de erros. • Especificação pobre em detalhe do código. • Mais facilitado. • Utilidade limitada após estabelecimento dos requisitos. • Utilidade para testes de usabilidade limitada • Limitações de fluxo de navegação
Protótipo de alta fidelidade	• Funcionalidade completa. • Totalmente interativo. • Dirigido aos usuários. • Define claramente o esquema de navegação. • Uso para exploração e teste. • Mesma aparência do produto final. • Serve como uma especificação viva. • Ferramenta de venda e marketing.	• Desenvolvimento mais caro. • Sua criação demanda tempo. • Ineficiente para *designs proof-of-concept* (demonstrações de que o conceito funciona). • Não serve para coleta de requisitos.

Fonte: Preece (2013).

Wireframe é um tipo de prototipagem que pode ser de baixa e de alta fidelidade. O que irá definir a fidelidade do *wireframe* será o nível de complexidade dos elementos inseridos no mesmo. Os *wireframes* geralmente simplificam demonstrando qual conteúdo deverá aparecer em cada tela no produto final (BROWN, 2007).

Segundo a definição de Santos (2009), os *wireframes* (figura 4), apresentam com clareza a organização, hierarquização e interações de uma determinada página. Eles podem ser feitos em rascunho no papel, onde será mais rápido e menos fiel ao sistema final ou feitos digitalmente, onde é mais complexo e fiel, porém menos ágil. Todo um sistema pode ser planejado e validado através deste método, para posteriormente ser desenvolvido.

Figura 4 – Exemplo de um *wireframe*



Fonte: Garrett (2011).

Quando os elementos de *Design* de Interface, *Design* de Navegação e *Design* de Informação se integram em um mesmo documento, o esqueleto que inicia o processo para formalizar o *design* visual de um website é definido pelo *wireframe* (GARRETT, 2011).

Os *wireframes* são fundamentais para o sucesso do desenvolvimento do sistema. O problema é quando os *wireframes* são entregues para os clientes sem um acompanhamento, o cliente aprova sem entender e futuramente irá gerar possíveis problemas, devido a isso é importante que tenha um acompanhamento (YOUNG, 2011).

2.6.3 A importância de utilizar protótipos

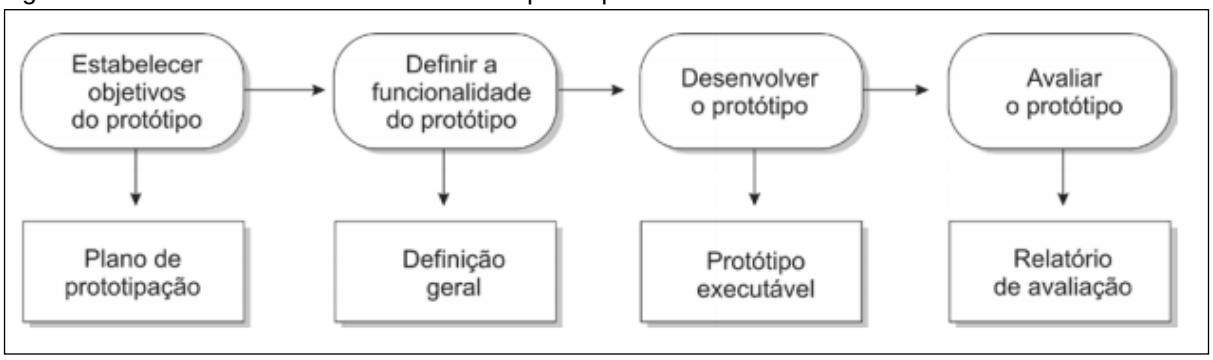
Os protótipos podem ajudar a dar novas ideias para requisitos e encontrar possíveis pontos fracos e fortes do sistema em desenvolvimento. Eles permitem aos utilizadores que vejam se o sistema dá suporte ao seu trabalho. Quando as funções são combinadas com outras, as vezes o usuário percebe que a visão inicial foi incompleta ou incorreta.

Um protótipo do sistema pode ser utilizado durante o desenvolvimento do sistema para fazer experimentos de projeto com o intuito de verificar a viabilidade da proposta. No processo de interface de usuário a prototipação é essencial, pois descrições textuais e diagramas não são suficientes para expressar seus requisitos (SOMMERVILLE, 2011).

Para Preece (2013), os protótipos são dispositivos de comunicação entre os membros responsáveis da equipe e são essenciais na discussão de ideias com os *stakeholders*. Os protótipos têm diversos propósitos, como: esclarecer requisitos vagos, testar a viabilidade de uma ideia, para realizar algum teste e avaliação com usuários ou para verificar se a direção do *design* é compatível com o resto do desenvolvimento do projeto.

Conforme a figura 5, o modelo de processos de desenvolvimento de protótipos é detalhado.

Figura 5 – Processo de desenvolvimento de protótipo



Fonte: Sommerville (2011).

Desde o início do processo os objetivos da prototipação devem ser estabelecidos. Os objetivos podem ser: desenvolvimento de um sistema para validação de requisitos funcionais, desenvolvimento de um sistema para prototipar a interface do usuário ou o desenvolvimento de um sistema para demonstrar a viabilidade da aplicação. Caso os objetivos não sejam declarados, o protótipo não será entendido pelos usuários e pela gerência.

O que deixar de fora e decidir o que colocar no protótipo é a próxima etapa do processo, tudo isso visando reduzir os custos e acelerar o cronograma da entrega.

A avaliação do protótipo é a última etapa. Os objetivos do protótipo devem ser utilizados para proceder um plano de avaliação e as provisões devem ser feitas para o treinamento do usuário. Os usuários precisam de um tempo hábil para se acostumar com o sistema novo, após a familiaridade do novo sistema, eles descobrem possíveis omissões e erros de requisitos (SOMMERVILLE, 2011).

2.6.4 Abordagens de prototipação

A prototipação baseada em software é dividida e categorizada em três áreas distintas (FLOYD, 1984):

a) **abordagem exploratória**: nesta abordagem segundo Floyd (1984), a ênfase está em identificar ou esclarecer os requisitos e as características desejadas do sistema final e ter novas possibilidades para as soluções discutidas. A prototipação rápida descartável e modelo espiral (BOEHM,1988), são dois métodos que utilizam essa abordagem. No protótipo rápido descartável, os usuários provêm o *feedback* aos responsáveis pelo desenvolvimento revisando o protótipo, que é útil para refinar a especificação dos requisitos. Desde que haja algum acordo entre os desenvolvedores e usuários, a implementação do sistema é continuada (DAVIS; BERSOFF; COMER, 1988);

b) **abordagem experimental**: o protótipo desenvolvido pode ser uma simulação funcional parcial que demonstra características do sistema, um "esqueleto" que mostra a estrutura do sistema e uma simulação funcional que demonstra todas as funções do sistema. Nesta abordagem, o uso experimental pode ser examinado pelas soluções propostas (MAYHEW; DEARNLEY, 1987);

c) **abordagem evolucionária**: a abordagem evolucionária visa a adaptação gradual do sistema com as mudanças de requisitos e é subdividas em incremental e evolucionária. (FLOYD, 1984). O método de desenvolvimento incremental consiste em várias versões parciais, onde individualmente cada uma se integra a uma parte do projeto original. A performance e certas funcionalidades são adicionadas gradativamente ao sistema parcial (INCE; HEKMATPOUR, 1987). O desenvolvimento evolucionário segundo Floyd (1984), precisa de uma sequência de ciclos de reprojeto, reimplementação e reavaliação. Tudo isso sem esforço para captar previamente um conjunto completo de requisitos. Novos requisitos são implementados, após um entendimento

maior no momento em que os usuários utilizam o protótipo (DAVIS; BERSOFF; COMER, 1988).

2.6.5 Prototipagem baseado na Web

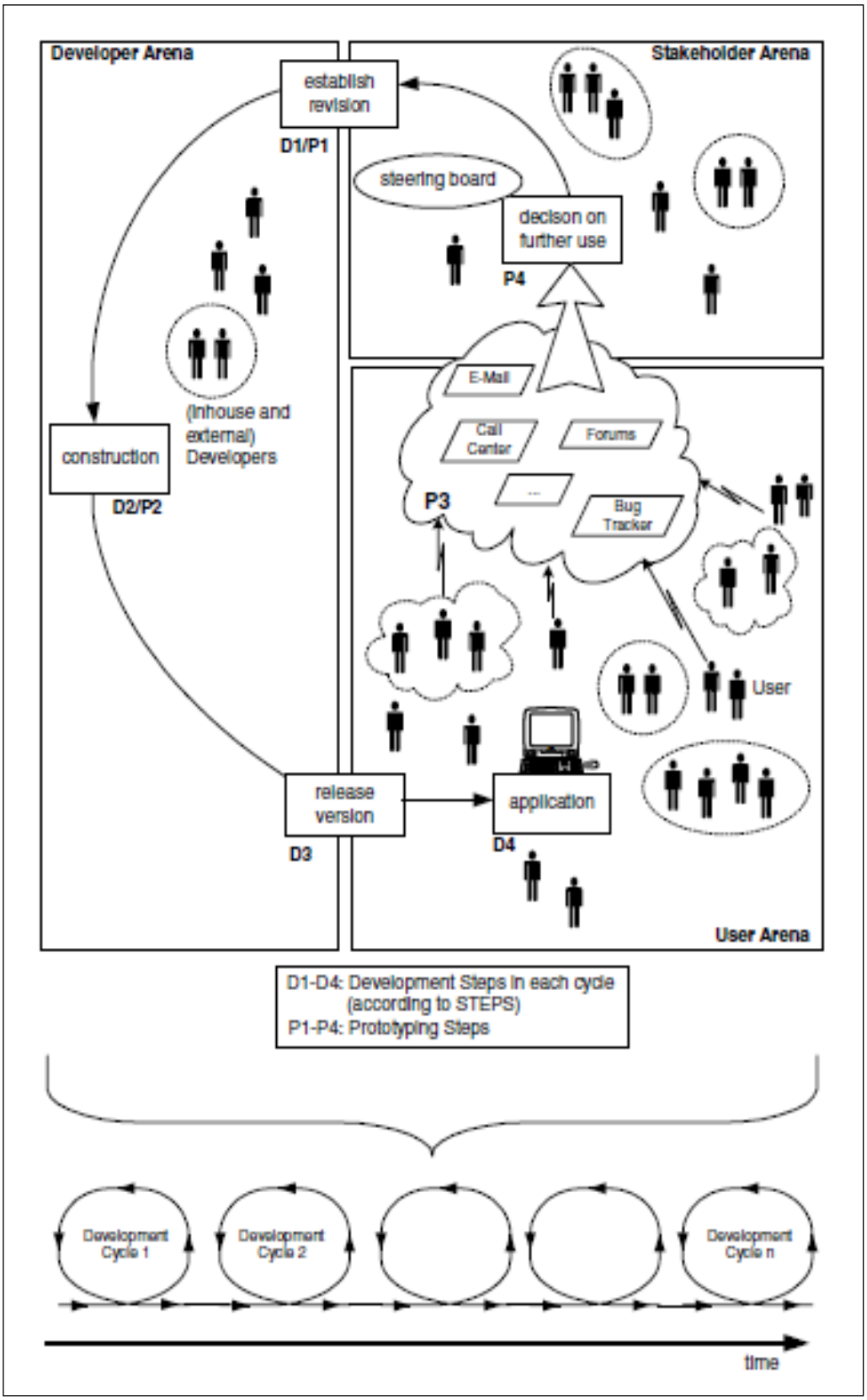
A prototipação baseada na Web, segundo Jeenicke *et al.* (2003), é diferente da tradicional. O maior problema enfrentado pelos usuários, é identificar os requisitos. Eles precisam de um artefato de *design* para auxiliá-los na especificação de suas necessidades. É baseado no desenvolvimento curto em ciclos, e utiliza o *feedback* dos usuários e outros fatores relevantes durante o desenvolvimento.

Dentro do desenvolvimento evolucionário e participativo, as abordagens de ciclo colocam a ênfase na comunicação do desenvolvedor com o usuário. Os passos do modelo propõem ciclos constituindo: revisões, produção, liberação de uma versão do sistema e aplicação desta versão.

Os quatro passos podem ser considerados um ciclo dentro do processo de desenvolvimento evolucionário conduzido pela abordagem da prototipagem. As decisões tomadas após a avaliação, fornecem informações necessárias para o próximo ciclo começando com a seleção funcional e técnica antes da construção da próxima versão. Baseado nas correções essenciais e nas mudanças inovadoras, os requisitos para a nova versão devem ser limitados para que a construção e lançamento da nova versão não leve mais que três meses (FIFTEENTH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 2003).

Baseado nesse tipo de abordagem, a prototipagem é proposta para realizar um desenvolvimento de software evolutivo no âmbito Web (figura 6).

Figura 6 – O processo da e-prototipagem como parte do ciclo de desenvolvimento



Fonte: Fifteenth International Conference on Software Engineering and Knowledge Engineering (2003).

3 ENGENHARIA DE REQUISITOS

A engenharia de requisitos é responsável pelo processo de descoberta, análise, documentação, gerenciamento e controle da qualidade dos requisitos de um sistema e as suas restrições. Os requisitos são fundamentais no processo de software e é considerado um fator determinando para o sucesso ou fracasso de um projeto (SOMMERVILLE, 2004).

Segundo Preece (2013), um dos objetivos na atividade de requisitos é entender o melhor possível sobre os usuários, suas atividades e o contexto das atividades para que o desenvolvimento possibilite dar suporte para que os objetivos sejam alcançados. Além disso, outro objetivo é produzir uma lista de requisitos estáveis para formar uma base sólida para iniciar os trabalhos.

> A engenharia de requisitos ajuda os engenheiros de software a compreender melhor o problema que eles vão trabalhar para resolver. Ela inclui o conjunto de tarefas que levam a um entendimento de qual será o impacto do software sobre o negócio, do que o cliente quer e de como os usuários finais vão interagir com o software (PRESSMANN, 2010, p.116).

A engenharia de requisitos pode ser dividida em três áreas: Levantamento de requisitos; Especificação ou análise de requisitos; Validação. A prática da atividade de levantamento de requisitos é a chave da engenharia de requisitos e deve ser feito com muito cuidado na hora de apurar o que é necessário para o desenvolvimento do sistema, identificar e levantar os requisitos com cuidado, identificar as prioridades e integrar diferentes demandas (CHRISTEL; KANG, 1992).

As técnicas de análise de requisitos auxiliam na produção mais precisas de especificações. As chances de êxito são muito maiores quando é realizado uma documentação de qualidade. O conjunto de levantamento, documentação e análise, dão origem a engenharia de requisitos que ajuda e faz parte da engenharia de software (PAULA FILHO, 2000).

3.1 ESTABELECENDO REQUISITOS

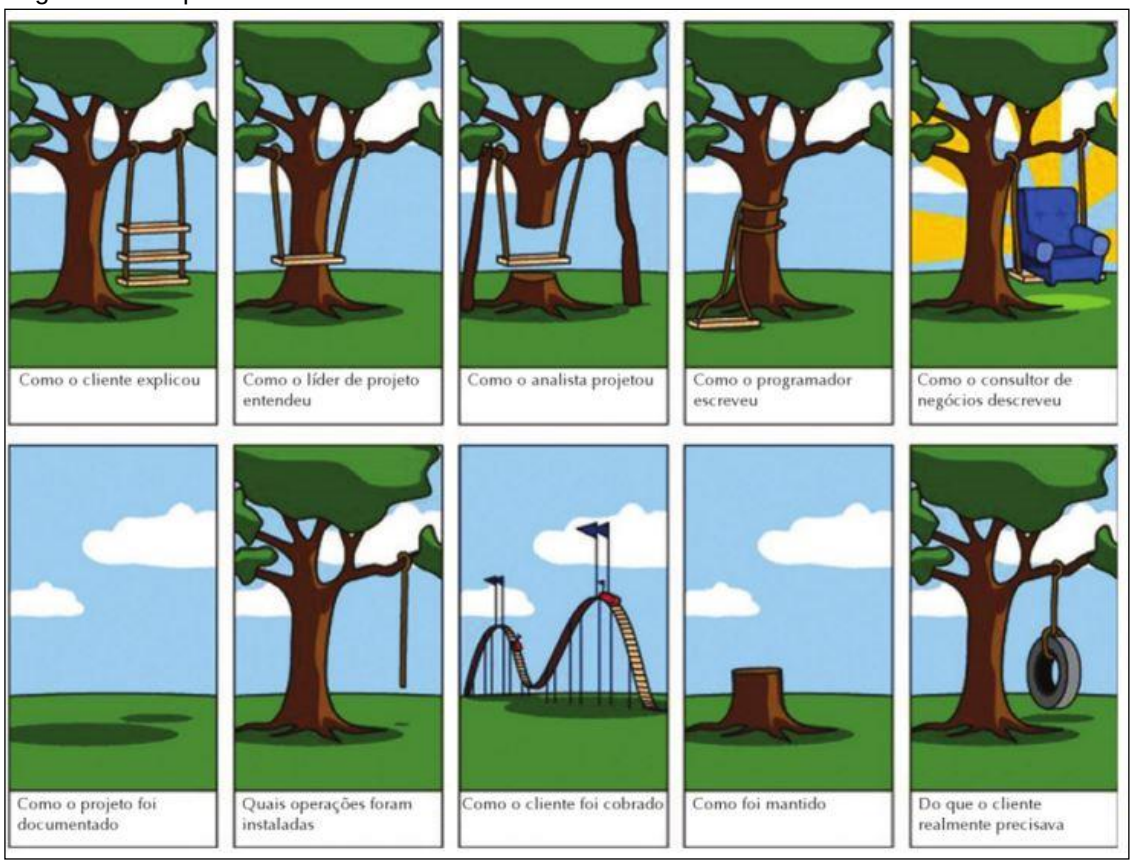
Existem diversos termos para esta atividade de entender o que deveria fazer um produto, os mais conhecidos são: levantamento de requisitos, análise de requisitos, coleta de requisitos, captura de requisitos e engenharia de requisitos. No levantamento os clientes e usuários dizem o que tem de ser feito e eles tem o

conhecimento dos requisitos necessários, mas nem sempre os requisitos são fáceis de serem identificados.

Normalmente o termo análise de requisitos descreve a atividade de analisar e investigar os requisitos coletados inicialmente. É de suma importância que as informações coletadas sejam analisadas para se obter um melhor entendimento do desenvolvimento do projeto. A engenharia de requisitos identifica que o desenvolvimento de um conjunto de requisitos é um processo interativo de negociação e evolução que deve ser gerenciado e controlado com muito cuidado (PREECE, 2013).

A charge demonstrada na figura 7 ilustra quando os requisitos não estão realmente articulados.

Figura 7 – Requisitos não articulados



Fonte: Preece (2013).

A falta de comunicação com o usuário, as especificações de baixa qualidade e a análise insuficiente são os principais responsáveis pela estimativa má e do alto custo gerado durante as atividades de requisitos, por isso que ela deve ser bem realizada (DAVIS, 1995).

3.2 CONCEITO DE REQUISITO

Os requisitos de um sistema possuem especificações as quais o sistema deve prover, propriedades gerais, restrições em como se deve operar e restrições que precisam ser executadas durante o processo de desenvolvimento.

Os requisitos de um sistema são características ou definições que o sistema pode realizar para obter os objetivos (PFLEEGER, 2004).

É necessário conhecer os requisitos e documentar de maneira correta para um desenvolvimento bem-sucedido de um projeto (POHL; RUPP, 2011, tradução nossa).

Segundo Leite (2001) os requisitos demonstram a necessidade dos clientes e mantém a qualidade do software.

Maciaszek (2001) argumenta que um requisito é uma capacidade ou condição que deve ser obtida por um sistema para satisfazer a especificação, padrão, contrato.

Há várias taxonomias para classificar o número variado de requisitos. São identificados como: funcionais ou não funcionais, fixos ou voláteis, de negócio ou de sistema.

3.3 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os requisitos funcionais são declarações de serviços de como o sistema deve reagir a entradas específicas e como deve comportar em determinadas situações. Os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer. A especificação deve ser completa e consistente (SOMMERVILLE, 2007).

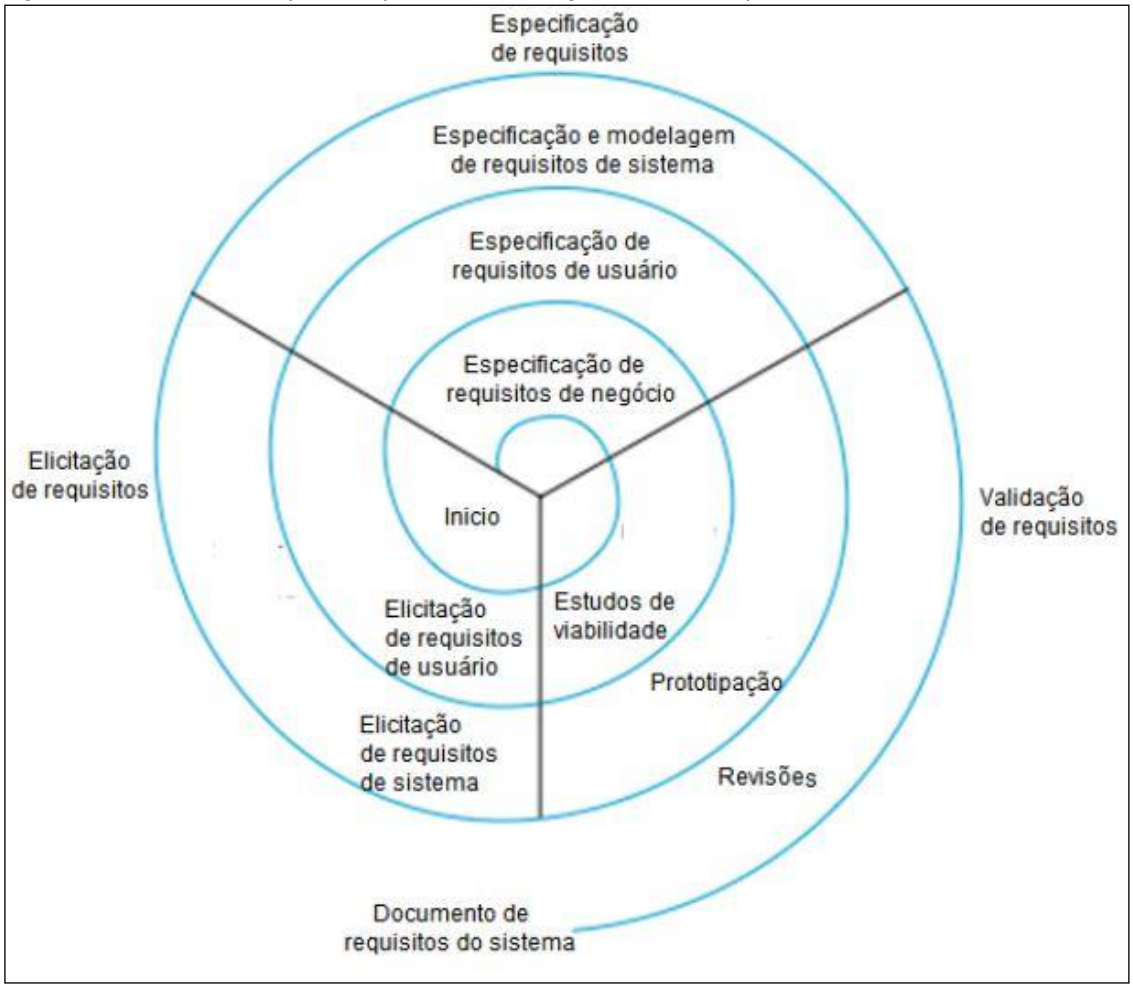
Os requisitos não funcionais são restrições as funções ou serviços do sistema. As restrições de timing, processo de desenvolvimento e outras restrições que são impostas pelas normas (SOMMERVILLE, 2007).

Os requisitos funcionais e não funcionais não são independentes e geralmente restringem ou geram outros requisitos. Para garantir que essas características e serviços sejam entregues corretamente, os requisitos de sistema devem especificar os serviços e características e as funcionalidades necessárias.

3.4 O PROCESSO DE ENGENHARIA DE REQUISITOS

O processo de engenharia de requisitos geralmente inclui quatro atividades de alto nível. O estudo de viabilidade visa avaliar a utilidade do sistema para a empresa; a elicitação e análise para descobrir os requisitos; a especificação, convertendo-os em alguma forma-padrão; a validação, onde verifica se os requisitos definem realmente o que o cliente quer. Na prática o processo é iterativo e as atividades são intercaladas (figura 8).

Figura 8 – Uma visão espiral do processo da engenharia de requisitos



Fonte: Sommerville (2007).

O modelo espiral organiza abordagens que os requisitos são desenvolvidos em diferentes níveis de detalhamento. A espiral pode acabar após a definição de alguns requisitos de usuário. Substituindo a prototipação, o desenvolvimento ágil pode ser utilizado para que a implementação e os requisitos do sistema possam ser desenvolvidos em conjunto.

Geralmente em quase todos os sistemas os requisitos mudam. As pessoas responsáveis envolvidas obtêm uma melhor compreensão do que querem e precisam do software e o cliente também muda.

3.4.1 Levantamento de Requisitos

O levantamento de requisitos também conhecido como elicitação de requisitos é um processo de busca, descoberta, aquisição e elaboração de requisitos para sistemas. A elicitação de requisitos está preocupada com a aprendizagem e entendimento das necessidades dos usuários e de quem financia o projeto, com a finalidade de comunicar os desenvolvedores a respeito destas necessidades (AURUM; WOHLIN, 2005).

Conforme Leite (2001), na tabela 3 há algumas técnicas para o levantamento de requisitos.

Tabela 3 - Técnicas para levantamento de requisitos.

Técnica	**Qualidade**	**Deficiências**
Leitura de documentos	facilidade de acesso às fontes de informação, volume de informação	dispersão das informações, volume de trabalho requerido para identificação de fatos
Observação	baixo custo, pouca complexidade da tarefa	dependência do ator (observador), superficialidade decorrente da pouca exposição ao universo de informações
Entrevistas	contato direto com os atores, possibilidade de validação imediata	conhecimento tático, diferenças culturais
Reuniões	múltiplas opiniões, criação coletiva	dispersão custo
Questionários	padronização de perguntas, tratamento estatístico	limitação das respostas, pouca interação/participação
Etnografia	visão de dentro para fora, contextualização	tempo, pouca sistematização
Participação ativa dos autores	envolvimento de clientes e usuários, validação	treinamentos, falsa impressão de eficácia
Análise de protocolos	fatos não observáveis, melhor compreensão dos fatos	foco na performance, o que se diz não é o que se faz
Engenharia reversa	disponibilidade de informação (código), reutilização	descontinuidade de informações, informação muito detalhada
Reutilização	produtividade, qualidade	nível de abstração (requisitos), possibilidade de reutilização real

Fonte: Leite (2001).

Nesta fase os clientes e usuários finais trabalham em conjunto com os desenvolvedores para abstrair todas as informações necessárias sobre a aplicação (SOMMERVILLE, 2011).

3.4.2 Análise de requisitos

O propósito da análise de requisitos é detalhar os requisitos de usuário, pois quando identificados no levantamento de requisitos, eles são descritos em linguagem natural e com poucos detalhes. Então na análise coloca-se os requisitos em um nível de descrição de requisitos do sistema, com mais detalhes. Os requisitos de usuário são organizados e detalhes pelos analistas o que irá gerar os requisitos de sistema. Para garantir nenhum equívoco seja realizado pelos analistas durante a fase de especificação de requisitos é necessário que a derivação de requisitos de sistema a partir dos requisitos de usuário seja feita (AURUM; WOHLIN, 2005).

3.4.3 Validação de requisitos

O processo de validação de requisito define realmente o que o cliente quer. Esta parte do processo é importante, pois erros nesta etapa pode gerar um custo muito alto de retrabalho durante o desenvolvimento ou após o projeto estiver finalizado. O custo para consertar erros de codificação ou de projeto é muito menor que o custo que se tem para consertar um erro de requisito. Quando ocorre alguma mudança de requisito, o projeto e a implementação do mesmo tem que ser alterado e também retestado.

A validação é essencial, pois não se sabe se os requisitos estão corretos, e o software que será desenvolvido pode também não satisfazer e cumprir com os objetivos dos requisitos (PINHEIRO, 2003).

Segundo Graham (2002), é fundamental a participação de elementos de teste para se obter bons requisitos na validação de requisitos. Incluir elementos de testes nas inspeções e revisões de requisitos; Dar início ao planejamento de testes em conjunto com a análise de requisitos; Usar como exemplos na especificação de requisitos as condições e casos de teste identificados; marcar no documento de requisitos os casos específicos detectados; utilizar casos de uso e cenários de negócios para demonstrar e exemplificar como o sistema deveria funcionar.

Durante o processo de validação, conforme Sommerville (2011), no documento de requisitos deve ser feito diferentes tipos de verificações, como:

a) verificação de validade: é necessário fazer mais análises e estudos para identificar funções diferentes ou adicionais que são exigidas. Os sistemas possuem uma base variada de usuários com costumes e necessidades distintas, qualquer conjunto de requisitos é uma solução que conciliaria os usuários;
b) verificação de consistência: não pode haver descrições diferentes para a mesma função de um sistema e nem restrições incoerentes.
c) verificação de completeza: os requisitos que definem todas as funções e restrições que são exigidas no sistema pelo usuário, devem ser incluídos no documento de requisitos;
d) verificação de realismo: levando em conta a tecnologia utilizada, os requisitos são verificados para assegurar que é realmente possível de implementa-los. O orçamento e os prazos também são levados em conta;
e) facilidade de verificação: os requisitos do sistema devem ser escritos de maneira que possam ser verificados, para reduzir possíveis divergências entre cliente e fornecedor. Uma soma de verificações pode ser projetada para mostrar se o sistema entregue cumpre com esses requisitos.

Além disso, é possível utilizar em conjunto ou isoladamente diversas técnicas de validação de requisitos como as revisões de requisitos; geração de casos de teste; análise automatizada da consistência; e prototipação, a qual foi citada no Capítulo 02.

3.5 GERENCIAMENTO DE REQUISITOS

As mudanças nos requisitos às vezes são inevitáveis. Por mais perfeita que seja feito o levantamento, podem ocorrer mudanças durante o processo por causa de fatores externos e estará sempre sujeito a limitação do ser humano. A gestão e o gerenciamento de requisitos propõem manter sob controle o conjunto de requisitos (PAULA FILHO, 2000).

A satisfação dos clientes é o principal objetivo da gestão de requisitos. Quando os prazos e os orçamentos são cumprimos, consequentemente os clientes ficam satisfeitos e as expectativas são alcançadas. A gestão de requisitos envolve os

requisitos técnicos e os não técnicos e estabelece a base para planejar, estimar, executar e acompanhar as atividades de um projeto, melhorando o desenvolvimento do sistema (MIRANDA, 2003).

Segundo Pressmann (2010), o gerenciamento de requisitos é um compilado de atividades que dão suporte a equipe de desenvolvimento a identificar, controlar e rastrear alterações em requisitos. O primeiro passo é identificar os requisitos, após este passo, os requisitos são incluídos em tabelas de rastreamento que os relacionam a aspectos do sistema como relação com interfaces do sistema, características dos requisitos, dependência com os outros requisitos, fonte de obtenção do requisito e associação com subsistemas.

3.6 PROCESSO DE DESENVOLVIMENTO WEB

As características de desenvolvimento de aplicações Web, são diferentes do processo convencional de desenvolvimento de software. Algumas atividades específicas buscaram diferenciar ambos, como a construção de diagramas de navegação e hipertexto (CERI *et al.*, 2000; CONALLEN 1999; ISAKOWITZ *et al*., 1995; PRESSMAN, 2006; SCHWABE *et al*., 1999). Design de interfaces com layouts e elementos utilizando HTML (CERI *et al.*, 2000; CONALLEN 1999; ISAKOWITZ *et al*., 1995; PRESSMAN 2006; SCHWABE *et al.*, 1999). Construção de um modelo de Arquitetura da Informação (CERI *et al*., 2000; PRESSMAN 2006; SCHWABE *et al.*, 1999).

Conforme Ceri *et al*. (2000), para representar uma aplicação Web, quatro perspectivas ortogonais são definidas: modelo estrutural; modelo de hipertexto, envolvendo o modelo de composição e navegação; modelo de apresentação; modelo de personalização.

Segundo Pressmann (2010), os modelos de processos para WebApp adotam a filosofia de desenvolvimento ágil, porém não descarta a utilização de um processo incremental para aplicações que exigem um tempo maior de desenvolvimento. A seguir, fases de um processo incremental no desenvolvimento:

a) comunicação com o cliente: abrange a análise de negócio que define o contexto do negócio e a formulação de negócio que utiliza a coleta de requisitos envolvendo todas as partes;

b) planejamento: representa uma definição de um cronograma e de uma tarefa, projetado para o desenvolvimento do incremento da aplicação.
c) modelagem: construir modelos de análise e projetos ágeis que definem requisitos e representam uma WebApp;
d) construção: tecnologias e ferramentas são aplicadas para a modelagem da aplicação Web;
e) implantação: o WebApp é entregue para os usuários finais e configurado para o ambiente operacional, após a entrega, o período de avaliação é iniciado.

Conforme descrito no capítulo 02, os processos ágeis focam no produto e não na documentação. O desenvolvimento é iterativo e incremental, e algumas ideias podem ser retiradas do Agile Manifesto, onde seres humanos e interações são mais relevantes do que processos e ferramentas; software em funcionamento é mais importante do que a documentação; colaboração do cliente é mais importante do que negociação por contrato; responder a mudanças é mais importante do que seguir um plano pré-definido.

No levantamento de requisitos para Web é preciso que seja integrado os aspectos relacionados ao conteúdo visual. É importante adicionar visões de diferentes disciplinas como de Usabilidade e Design Gráfico (OVERMYER, 2000).

Segundo Gonçalves (2010), dentre as abordagens analisadas, os projetos de aplicações Web envolvem: Projeto de informação, onde define o conteúdo que será apresentado; projeto de navegação, descreve como a informação é acessada; projeto de apresentação, demonstra como a informação é apresentada ao usuário final, com o *layout* de interface e protótipos; projeto funcional, estabelece como é o processamento da informação na aplicação.

3.6.1 XWebProcess

O *XWebProcess* é um processo ágil utilizado para o desenvolvimento Web e é baseado no *Extreme Programming* (XP). Ele adaptou os elementos do XP para o domínio Web, atendendo melhor as interfaces do usuário, navegação, requisitos não funcionais, testes e suporte de infraestrutura (SAMPAIO *et al*., 2004).

O início do processo é pela disciplina exploratória que envolve protótipos. O objetivo é avaliar junto ao cliente a viabilidade e definir os requisitos iniciais.

Após isto, é feito a definição e revisão dos requisitos. A equipe define o esforço e o cliente define a prioridade de cada historieta. Nesta disciplina foi incluída a atividade de design de arquitetura, para ser mais flexível.

Os envolvidos no projeto planejam o próximo release, escolhendo as historietas que devem ser implementadas. Várias iterações ocorrem dentro de cada release e para cada iteração é utilizada as seguintes disciplinas: planejamento de iteração, design, escrita de teste de unidade, codificação, teste e integração.

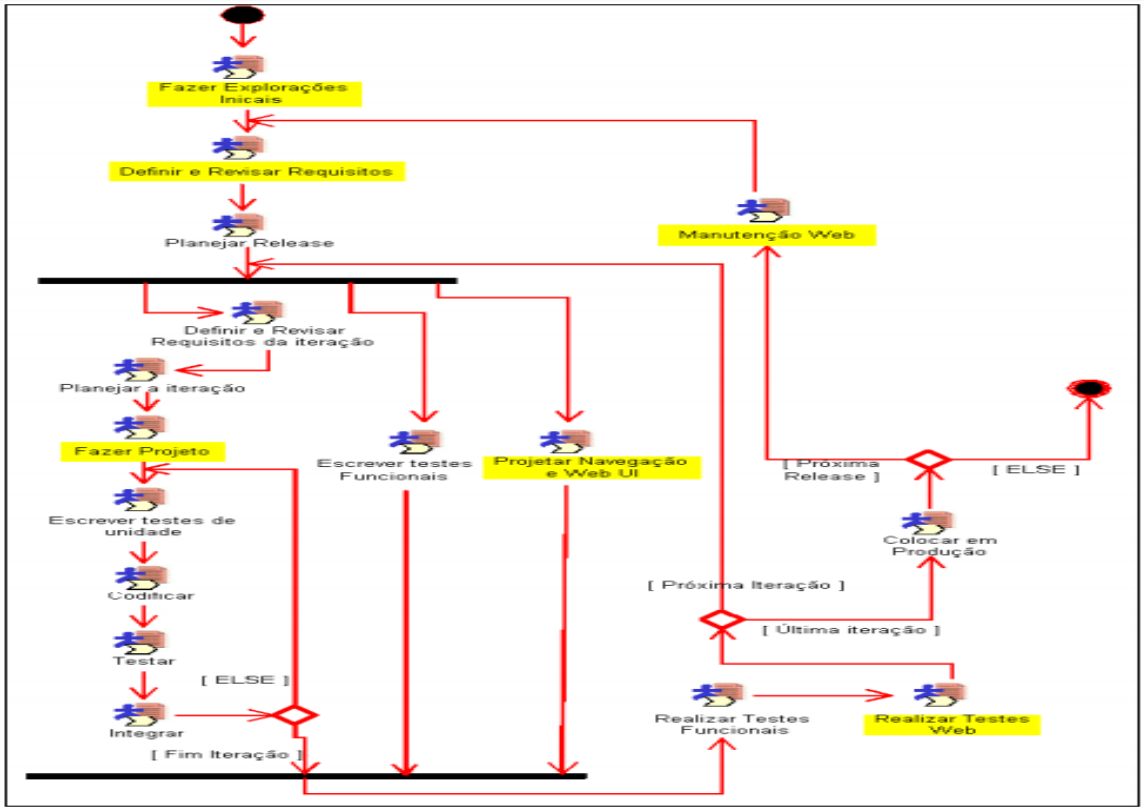
Na disciplina de design foi incluído a atividade de design da camada de dados. As estimativas anteriores devem ser reavaliado e pode haver mudanças nos requisitos durante a iteração. Devido a importante a uma aplicação Web o design de navegação e apresentação foram inseridas e são feitos em paralelo com as atividades anteriores.

É preciso verificar com os testes funcionais após o fim da iteração se está em concordância com o que foi proposto. Para verificar se os requisitos não funcionais foram atendidos, foi acrescentado a disciplina de testes Web.

A figura 9 ilustra uma visão dinâmica do *XWebProcess* e as disciplinas inseridas ou modificadas, estão destacadas.

Na última iteração do *release*, o sistema entra em produção. O processo é finalizado quando todas as historietas forem implementadas e postas em produção.

Figura 9 – XWebProcess sob ponto de vista dinâmico



Fonte: Sampaio et al (2004).

3.6.2 Modelo WebML

A WebML permite que os designers expressem as funcionalidades em um nível alto, sem que comprometa os detalhes. Os conceitos são associados com uma representação gráfica que pode ser facilmente entendida por membros da equipe. A WebML também suporta a sintaxe XML e consiste em quatro perspectivas ortogonais (CERI; FRATERNALI; BONGIO, 2000).

a) modelo estrutural: descreve o conteúdo de dados que contém o website. É baseado no modelo de entidade e relacionamento, porém também é compatível com diagramas de classes *Unified Modeling Language* (UML);

b) modelo de hipertexto: descreve um ou mais hipertextos que podem ser publicados no site. Cada hipertexto define uma visão do site, que pode ser subdividade em:

- modelo de composição: especifica quais são as páginas que constituirão o site. Há seis tipos de unidades de conteúdo (*content units*) que são utilizados para demonstrar os componentes que serão retirados do modelo estrutural e inseridos nas páginas,

- modelo de navegação: expressa como as páginas e as unidades de conteúdo são interligados para formar o hipertexto;

c) modelo de apresentação: representa o layout e a aparência gráfica das páginas, independente do dispositivo de saída ou da linguagem utilizada;

d) modelo de personalização: os usuários e grupos de usuários são explicitamente modelados no esquema estrutural. Os recursos dessas entidades pode ser utilizado para armazenar conteúdo específico individual ou de grupo, como sugestões de compras, lista de favoritos, entre outros.

O desenvolvimento de aplicações Web utilizando o WebML consiste em distintas fases que podem ser feitas de maneira incremental e interativa. O processo do WebML possui vários ciclos e em cada ciclo é produzido um protótipo da aplicação que possibilita os testes e a progressão do desenvolvimento (DZENDZIK, 2004).

4 TECNOLOGIAS PARA DESENVOLVIMENTO WEB

Uma aplicação web precisa ser instalada em um servidor web que esteja disponível e que possa ser acessada através do protocolo *http*. O usuário precisará utilizar algum navegador para acessar a aplicação, como Internet Explorer e Google Chrome (CONALLEN, 1999; KAPPEL *et al*., 2006).

As aplicações web podem ser confundidas com websites dinâmicos, o que difere eles, são os objetivos a qual cada um tem. Um website dinâmico tem como objetivo informar, vender, entreter e uma infinidade de outras funções que não se encaixam nos negócios. As aplicações web são desenvolvidas conforme a lógica de negócio proposto, as informações são alteráveis nas bases de dados e o usuário pode interagir. Segundo a lógica do negócio, as informações são processadas e geram resultados que tendem alterar o estado atual do objeto do sistema (CONALLEN, 1999).

4.1 VANTAGENS DAS APLICAÇÕES WEB

Com o advento da internet, grande parte das empresas viu a necessidade de migrar suas aplicações convencionais para *Web*. A possibilidade de acessar a aplicação apenas com um *browser* (navegador), sem a necessidade de instalar nada no computador.

As atualizações são feitas através do servidor e são atualizadas instantaneamente para o cliente, sem a necessidade de distribuir pacotes de atualização ou instalação individual para cada máquina.

Muitos fatores técnicos tem trabalhado ao lado de incentivos comerciais para levar para a revolução atual na forma em que usamos a internet. A seguir alguns aspectos que contribuíram como vantagens para esse aumento proeminente das aplicações Web:

a) o protocolo de comunicação HTTP que é utilizado para acessar a *World Wide Web* é leve e sem perda de conexão. Isto proporciona a resiliência nos eventos de erros de comunicação e evita a necessidade do servidor em manter aberta uma conexão de rede para cada usuário;

b) todo usuário que é adepto a Web já tem um navegador instalado em seu computador ou dispositivo móvel. As aplicações Web implementam sua interface gráfica de usuário dinamicamente no navegador, assim evitando a necessidade de distribuir e implementar separadamente o software para cada cliente. As interfaces precisam ser implementadas apenas uma vez no servidor e o efeito é imediato;
c) os navegadores atuais são altamente funcionais, eles permitem a construção de uma interface de usuário bem rica e satisfatória. As interfaces Web utilizam *input controls* (controles de entrada) e navegação padrão que são familiares para os usuários, evitando a necessidade de aprender como cada elemento individual da aplicação funciona;
d) as principais tecnologias e linguagens utilizadas para desenvolver aplicações Web são relativamente simples. Uma ampla gama de plataformas e ferramentas para desenvolvimento estão disponíveis para facilitar a construção de poderosas aplicações. A quantidade de código-fonte aberto e outros recursos disponíveis para incorporar em aplicativos personalizados, auxiliam bastante quem está iniciando (STUTTARD; PINTO, 2011).

4.2 HYPERTEXT MARKUP LANGUAGE (HTML)

O *World Wide Web Consortium* W3C, é a entidade responsável por regular a definição dos standards da Web. Segundo o W3C, o HTML não é considerado uma linguagem de programação, mas sim uma linguagem de marcação (*markup*). Utiliza um conjunto de tags de marcação (*markup tags*) com o objetivo de descrever o conteúdo de uma página Web. O conteúdo é posteriormente interpretado pelo navegador e transforma as respectivas *tags* em conteúdo visual (CASTELEYN *et al.*, 2009).

Com um aumento das aplicações Web, o uso da linguagem de programação Javascript no desenvolvimento Web, fez com que o HTML 4 se tornasse limitado. A nova especificação do HTML5 trouxe novas funcionalidades que tornaram o desenvolvimento mais viável, como as novas funções: *drag and drop*, desenhos, eventos ordenados a partir do servidor, entre outros. Além disso a necessidade de se ter um standard (padrão) aberto para ser usado e implementado livremente, sem ter

que utilizar padrões proprietários como o Adobe Flash e Microsoft Silverlight (LAWSON; SHARP, 2012).

A nova versão do HTML5 tem como principal objetivo facilitar a manipulação do elemento possibilitando que o desenvolvedor modifique as características dos objetos de um modo que é transparente para o usuário final e não intrusiva. O HTML5 auxilia e fornece ferramentas para o Javascript e o CSS trabalharem da melhor maneira possível juntos. Através de suas APIs, o HTML5 permite a manipulação das características destes elementos, de uma forma que a aplicação fique leve e funcional (EIS; FERREIRA, 2012).

4.3 CASCADING STYLE SHEETS (CSS)

O CSS é utilizado para descrever a apresentação de páginas Web, incluindo cores, fontes e layouts. O código HTML é separado do CSS, o que facilita na hora de manter e gerenciar os websites. Também é possível adaptar sua aplicação para diferentes tipos de dispositivos com tamanhos de telas variáveis (MIRANDA, 2012).

> As CSS têm por finalidade devolver à marcação HTML/XML o propósito inicial da linguagem. A HTML foi criada para ser uma linguagem exclusivamente de marcação e estruturação de conteúdo. Isso significa que, segundo seus idealizadores, não cabe à HTML fornecer informações ao agente do usuário sobre a apresentação dos elementos. Por exemplo: cores de fontes, tamanhos de textos, posicionamentos e todo o aspecto visual de um documento não devem ser funções da HTML. Cabem às CSS todas as funções de apresentação de um documento, e essa é sua finalidade maior. Daí a já consagrada frase que resume a dobradinha CSS + HTML: “HTML para estruturar e CSS para apresentar.” (SILVA, 2012, p.25).

CSS3 foi iniciado em 1998, mas ainda é considerado novo e é uma especificação ainda em fluxo. Algumas partes destas especificações ainda são experimentais, mas grande parte dos navegadores modernos já aceita e considera a tecnologia estável. O CSS3 trouxe um visual muito elegante, o que era possível fazer com diversos códigos não otimizados e lentos, agora é possível utilizar ele para criar efeitos de rotação, dimensionamento, animação em duas e três dimensões, sombras, cantos arredondados, textos dinâmicos e efeitos de gradiente (GASSTON, 2011, tradução nossa).

4.4 PHP

O *Hypertext Preprocessor* (PHP) é uma das linguagens de programação mais utilizadas na Web. Esta linguagem está presente em milhões de sites do mundo inteiro e a principal diferença desta linguagem comparada com as outras, é a capacidade que o PHP tem de interagir com a Web e trazer dinamismo aos websites estáticos (NIEDERAUER, 2011).

Segundo Casteleyn et al, (2009), o PHP é a linguagem *server-side* de programação mais utilizadas atualmente. Esta popularidade se deve ao fato dela ser gratuita, *open source* com uma comunidade enorme de usuários que dão suporte, disponíveis em todas as plataformas e é suportado pelos maiores e mais populares *Web Servers*. Em sua essência é uma linguagem dinâmica tipada e também é orientada a objetos.

4.5 JAVASCRIPT EM APLICAÇÕES WEB

JavaScript é uma linguagem de programação Web, a maioria dos websites mais modernos utilizam esta linguagem. Todos os navegadores modernos de desktops, vídeo games, televisões, dispositivos móveis, interpretam o JavaScript fazendo-a uma das linguagens mais onipresentes na história. O JavaScript é uma das três tecnologias que todo desenvolvedor Web deve utilizar: HTML para especificar o conteúdo das páginas web, CSS para especificar a apresentação de páginas da web e JavaScript para especificar o comportamento de páginas da web (FLANAGAN, 2011).

Uma das utilidades do JavaScript é deixar a aplicação ou website mais dinâmico. Com o Javascript é possível manipular conteúdo e apresentação, por exemplo, inserir conteúdo diferenciado de acordo com o navegador utilizado pelo usuário; manipular o navegador, é possível controlar o comportamento do navegador apresentando mensagens aos usuários, pop-ups etc; interagir com formulários, permite a realização de cálculos, validação de dados e fornecendo dicas de preenchimentos em formulários HTML; interagir com outras linguagens dinâmicas,

costuma-se ser utilizada em conjunto com outras linguagens para realizar tarefas complementares relacionadas ao fluxo da programação (SILVA, 2010).

5 TRABALHOS CORRELATOS

Para a realização deste trabalho foram pesquisados alguns trabalhos semelhantes, adquirindo assim um maior embasamento do que já foi realizado. Foram encontrados diversos trabalhos, nacionais e internacionais sobre os assuntos de engenharia de software, engenharia de requisitos, prototipação e desenvolvimento web.

5.1 METODOLOGIA PARA EQUIPES DE DESENVOLVIMENTO DE REQUISITOS DE SISTEMAS DE INFORMAÇÃO

Desenvolvido em 2008 por Pedro Rodrigo Caetano Strecht Ribeiro como relatório de Dissertação Mestrado Integrado em Engenharia Informática e Computação para a Faculdade de Engenharia da Universidade do Porto em Portugal, o objetivo através deste trabalho é definir uma metodologia de desenvolvimento de requisitos adequada ao tipo de projeto desenvolvido e a aplicação dessa metodologia na especificação de um sistema em particular.

O principal resultado que se pretende alcançar com este trabalho é a definição de um processo de desenvolvimento de requisitos, que possa ser realizado de forma sistemática e com efeitos facilmente previsíveis, contribuindo dessa forma para o aumento da maturidade da organização. Como resultado adicional, pretende-se mostrar exemplos de artefatos produzidos durante a especificação de um sistema real para ilustrar a aplicação (RIBEIRO, 2008).

5.2 UMA ABORDAGEM SISTÊMICA PARA O PROCESSO DE PRODUÇÃO EM ENGENHARIA WEB, NA FASE DE CONCEPÇÃO

Este trabalho foi confeccionado por Rodrigo Franco Gonçalves como tese de Doutorado para Escola Politécnica da Universidade de São Paulo em 2010, o trabalho tem como objetivo apresentar diretrizes para o processo de desenvolvimento de aplicações Web, onde diferentes disciplinas, papéis e atividades são correlacionadas na fase de concepção do projeto.

Em função de diferentes padrões e categorias de aplicações Web, foi apresentado qual a melhor abordagem técnica do processo de desenvolvimento. No contexto apresentado deste trabalho, a abordagem correta a ser utilizada é aquela que no final do projeto o nível de qualidade é obtido mais rapidamente. Portanto, pode-se utilizar diferentes estruturas de processos de software, como: cascata, espiral, incremental, etc (GONÇALVES, 2010).

5.3 AS COMPETÊNCIAS DA EQUIPE DE PROJETO NO PROCESSO DE DESENVOLVIMENTO DE APLICAÇÕES WEB

Este trabalho foi desenvolvido por Daniela Szabluk como trabalho de conclusão de curso de Pós-Graduação em Design da Universidade federal do Rio Grande do Sul no ano de 2011 e teve como objetivo descobrir o alinhamento entre a metodologia adotada no processo de desenvolvimento de aplicações web e as competências da equipe de projeto.

A solução proposta para este trabalho é demonstrar a importância da utilização de uma metodologia no processo de desenvolvimento de uma aplicação web. Devido a metodologia utilizada, foi possível otimizar o tempo no desenvolvimento e contribuiu para a qualidade do produto final. Também foi possível comprovar a relação entre as etapas de metodologia de desenvolvimento e as competências dos profissionais. A partir do mapeamento realizado, é possível construir novas metodologias focando no desenvolvimento web e as áreas ao Design e às TICs (SZABLUK, 2011).

5.4 ELABORAÇÃO DE UMA PLATAFORMA PARA METANÁLISE DIAGNÓSTICA UTILIZANDO CONCEITOS E ANÁLISE DE REQUISITOS DA ENGENHARIA DE SOFTWARE

O trabalho foi realizado pelo acadêmico Jadson Frassetto como trabalho de conclusão de curso da Universidade do extremo sul catarinense - UNESC no ano de 2015 tendo como objetivo geral aplicar a engenharia de software para estruturar uma plataforma de metanálise diagnóstica através da elicitação e modelo de requisitos para auxiliar os envolvidos.

Foi possível avaliar o estudo com os conceitos e uso das práticas da engenharia de software. A metodologia utilizada baseou-se em estudar e pesquisar os conceitos da engenharia de requisitos, engenharia de software e da arquitetura de software.

Por meio de um estudo de viabilidade em três software distintos para realizar a metanálise, verificou-se a necessidade da elaboração de uma nova ferramenta. Foi possível validar junto às partes interessadas com o desenvolvimento do protótipo e verificar os requisitos. Ficou explicito a necessidade e a importância do ambiente WEB na colaboração de metanálises (FRASSETTO, 2015).

6 FERRAMENTA PARA VALIDAÇÃO DE REQUISITOS NO MODELO DE PROTOTIPAÇÃO EM PROJETOS WEB

Por meio do conhecimento adquirido durante o tempo do levantamento bibliográfico, foi possível criar um protótipo de aplicação de validação de requisitos baseado na estrutura Web, utilizando a tecnologia PHP para programar e fazer a interação entre cliente e servidor. A ferramenta faz parte do início do processo do *XwebProcess*, como visto no capitulo 5, onde envolve a parte de protótipos e da avaliação conjunta com o cliente. É possível fazer na ferramenta o cadastro de clientes, que será identificado como o agente responsável por fazer as anotações na imagem como forma de realizar o *feedback* e o cadastro de projeto, onde será possível atrelar um projeto a determinado cliente. Tendo em vista que a validação de requisitos é muito importante, o intuito deste projeto é criar algo para facilitar a comunicação entre cliente e desenvolvedor, evitando assim o tempo perdido e os custos gerados pela má comunicação na prototipação.

6.1 METODOLOGIA

O início deste projeto de pesquisa foi constituído pelo levantamento bibliográfico com o intuito de obter referências bibliográficas e conteúdo teórico de qualidade. Foram também realizadas pesquisas referentes à engenharia de software. Posteriormente os estudos foram voltados para as tecnologias de desenvolvimento Web, e para a idealização da construção do protótipo da ferramenta proposta.

A fundamentação teórica foi dividida em partes com a finalidade de dar embasamento teórico e estruturar a implementação. Esse estudo iniciou com a pesquisa sobre a engenharia de software e seus conceitos.

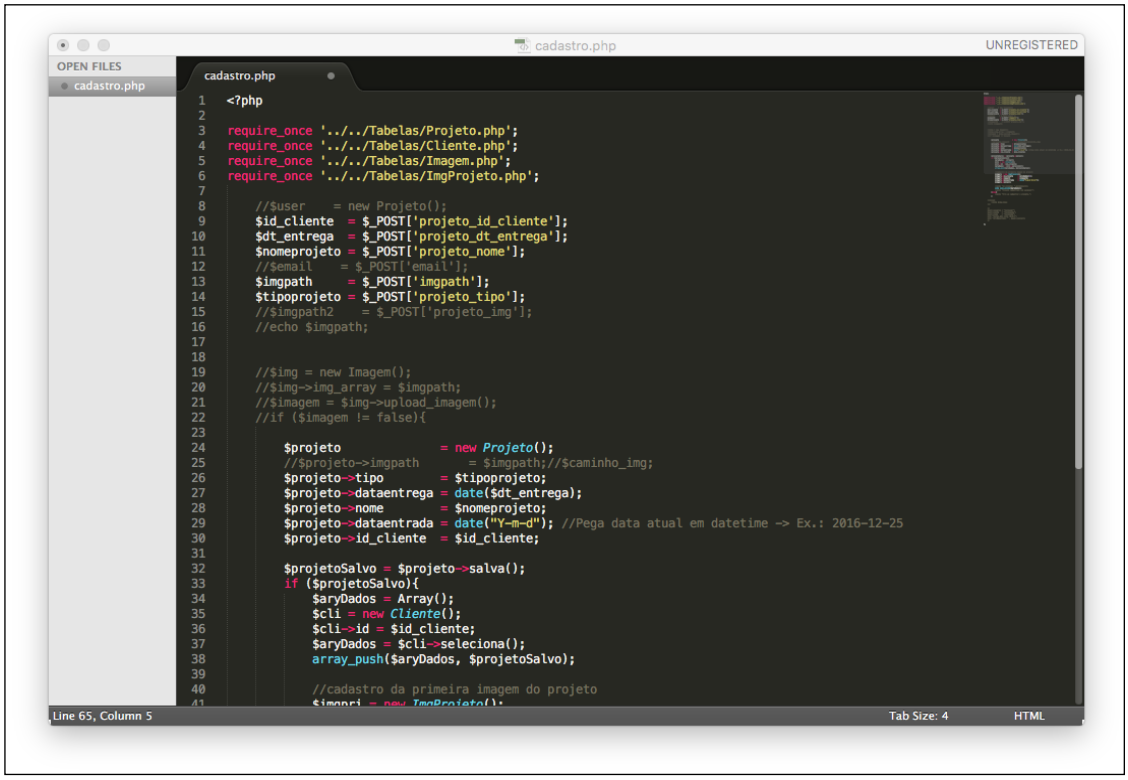
Em seguida foi abordado sobre a engenharia de requisitos, dando um entendimento maior sobre os aspectos gerais dos requisitos e uma visão geral sobre levantamento, análise, validação e gerenciamento de requisitos.

Para concluir, o estudo realizado foi sobre os projetos e desenvolvimento Web. Foi feito uma contextualização do cenário dos projetos Web e também sobre as tecnologias escolhidas para o desenvolvimento do projeto proposto.

6.2 FERRAMENTAS UTILIZADAS

Para começar o desenvolvimento do protótipo de ferramenta, foi utilizado o *Sublime Text 2* que é um editor de código fonte muito leve e customizável (figura 10). O arquivo pode ser baixado gratuitamente no endereço: https://sublimetext.com/2.

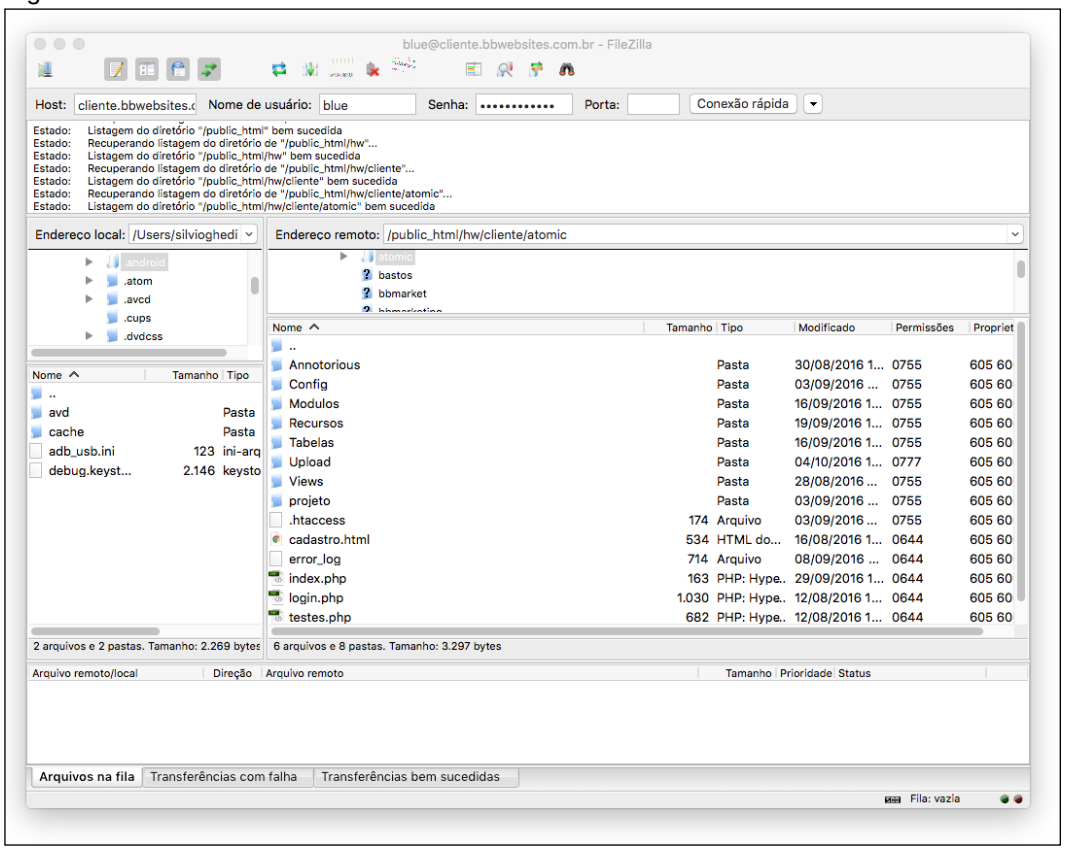
Figura 10 – Sublime Text 2



Fonte: Do autor.

Para enviar os arquivos para o servidor através do protocolo FTP (*File Transfer Protocol*), foi utilizado a ferramenta de código aberto multiplataforma FileZilla. Esta ferramenta tem uma interface gráfica intuitiva que auxilia no envio de arquivos para o servidor com uma conexão segura (figura 11).

Figura 11 – Filezilla



Fonte: Do autor.

Foi utilizado o *Annotorious* que é um kit de ferramentas de imagem para anotação *Open Source* escrito em JavaScript. O *plugin* pode ser baixado gratuitamente no endereço: http://annotorious.github.io/.

O servidor responsável onde foi publicado a aplicação, utilizou o Apache que é um servidor HTTP livre e o mais utilizado na Web, juntamente com PHP, que é amplamente usado por aplicações Web, a versão do PHP utilizado foi a: 5.4.45.

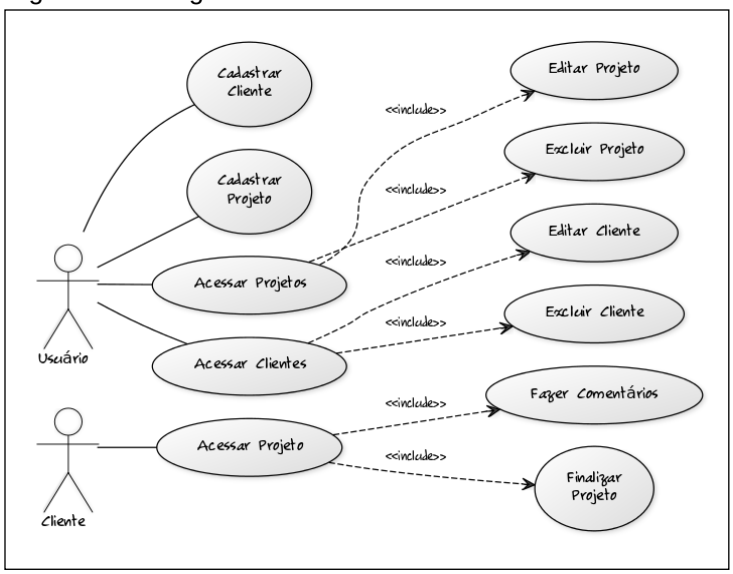
Para o desenvolvimento do banco de dados, foi utilizado o MySQL na versão 5.5.52, pois além de ser gratuito, seguro e com uma boa performance, ele é o mais utilizado para Web.

6.3 MODELAGEM DE DADOS

Todas as funções necessárias para a validação de requisitos estão presentes no projeto, começando pelo cadastro do cliente, cadastro de projeto, comentários na imagem. Esses dados são salvos no banco de dados no servidor web.

O diagrama de caso de uso na figura 12, foi criado para dar um melhor entendimento de como a ferramenta funciona.

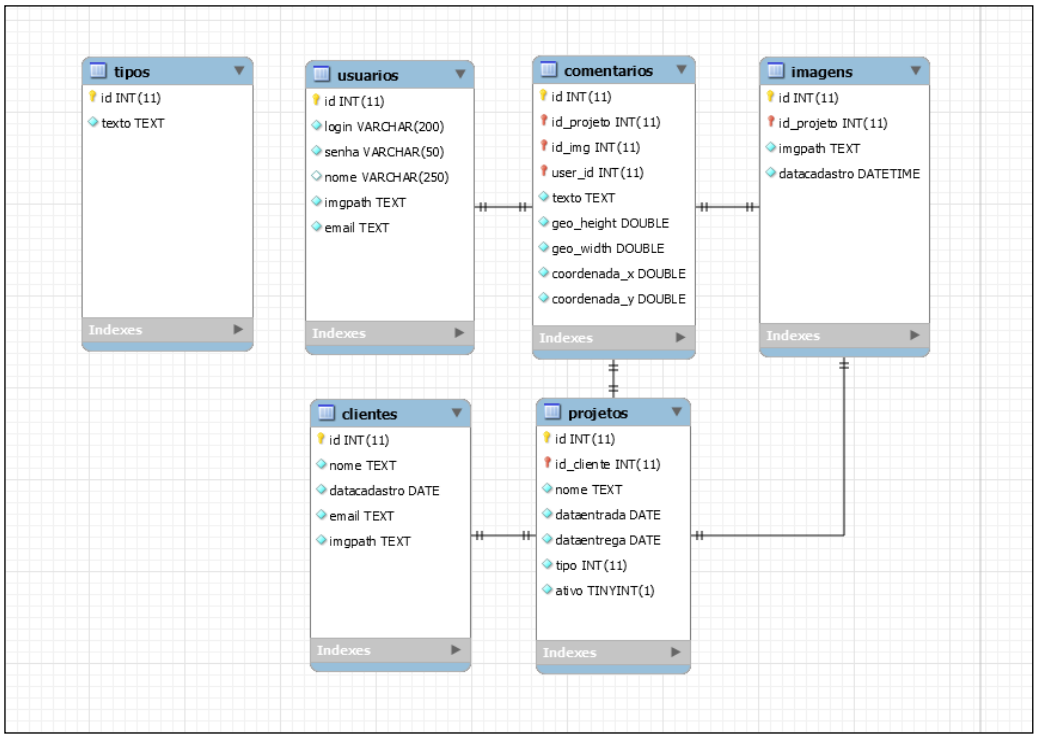
Figura 12 – Diagrama de casos de uso



Fonte: Do autor.

Para facilitar o entendimento do projeto e diminuir possíveis erros na hora do desenvolvimento foi realizado uma modelagem do banco de dados. Foi utilizado o programa MySQL Workbench, que é um software livre disponível nas plataformas Windows, Mac OS X e Linux. O MySQL Workbench é uma ferramenta visual que oferece modelagem de dados, desenvolvimento SQL e diversas outras opções (figura 13).

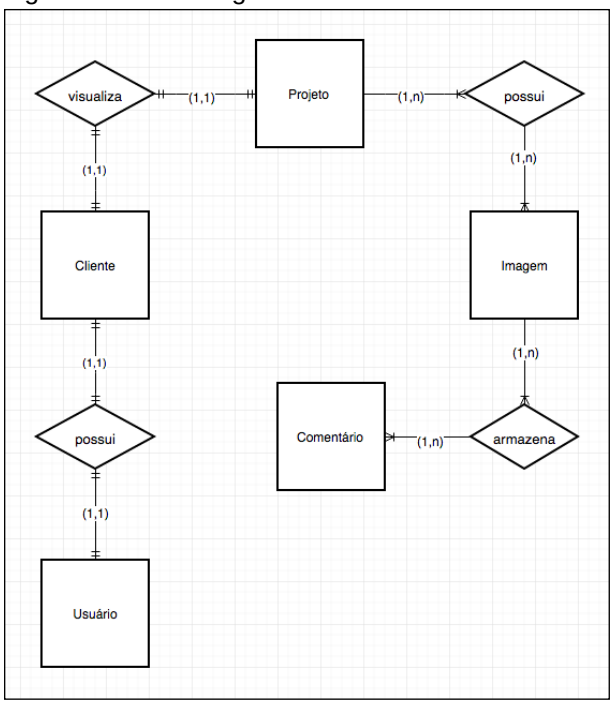
Figura 13 – Modelagem do banco



Fonte: Do autor.

Além disso, foi criado uma modelagem conceitual pois ajudaria no entendimento do processo e ela é a própria descrição do sistema proposto (figura 14).

Figura 14 – Modelagem conceitual



Fonte: Do autor.

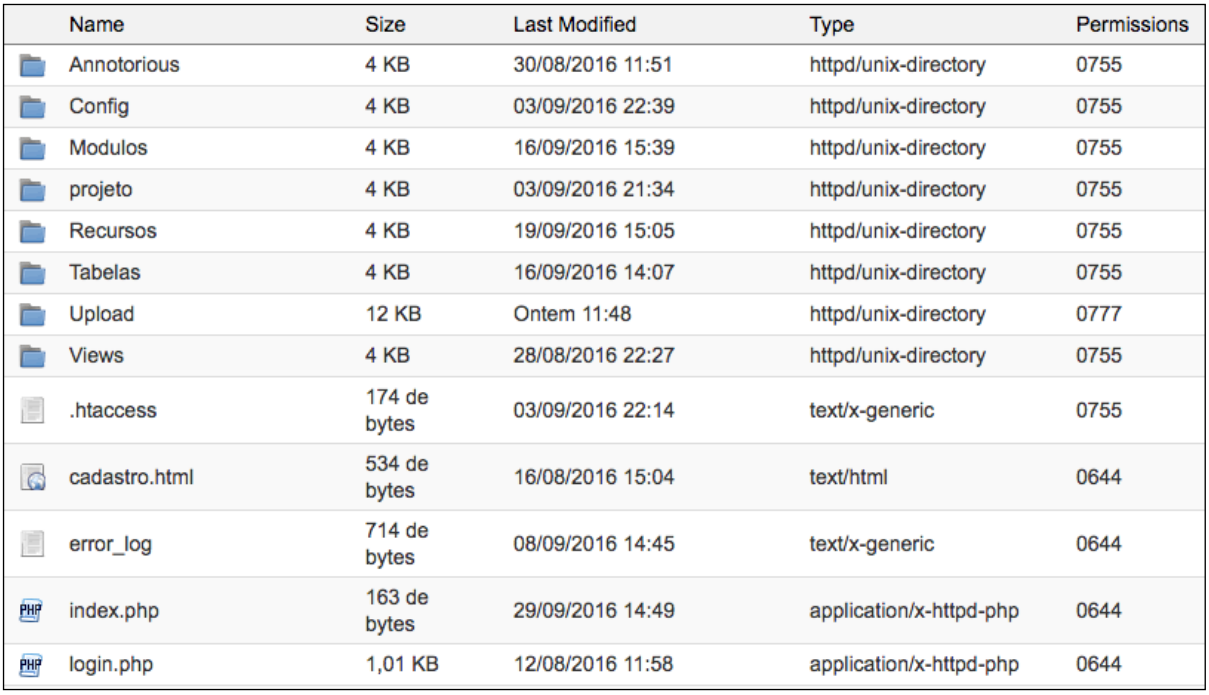
6.4 DESENVOLVIMENTO DA FERRAMENTA

Depois do planejamento inicial estruturado e das ferramentas devidamente instaladas e configuradas, deu-se início ao desenvolvimento da ferramenta. Primeiramente foi feito uma estrutura em HTML simples sem estilos para se ter uma base do projeto e um melhor entendimento do mesmo.

6.4.1 Estrutura do servidor web

A criação do projeto foi realizada localmente com o Sublime e enviada para o servidor através do Filezilla. Na figura 15 é possível visualizar a forma em que o projeto foi organizado e dividido.

Figura 15 – Estrutura do servidor web



Name	Size	Last Modified	Type	Permissions
Annotorious	4 KB	30/08/2016 11:51	httpd/unix-directory	0755
Config	4 KB	03/09/2016 22:39	httpd/unix-directory	0755
Modulos	4 KB	16/09/2016 15:39	httpd/unix-directory	0755
projeto	4 KB	03/09/2016 21:34	httpd/unix-directory	0755
Recursos	4 KB	19/09/2016 15:05	httpd/unix-directory	0755
Tabelas	4 KB	16/09/2016 14:07	httpd/unix-directory	0755
Upload	12 KB	Ontem 11:48	httpd/unix-directory	0777
Views	4 KB	28/08/2016 22:27	httpd/unix-directory	0755
.htaccess	174 de bytes	03/09/2016 22:14	text/x-generic	0755
cadastro.html	534 de bytes	16/08/2016 15:04	text/html	0644
error_log	714 de bytes	08/09/2016 14:45	text/x-generic	0644
index.php	163 de bytes	29/09/2016 14:49	application/x-httpd-php	0644
login.php	1,01 KB	12/08/2016 11:58	application/x-httpd-php	0644

Fonte: Do autor.

Para realizar a conexão com o banco de dados, dentro da pasta *Config* foi criado dois arquivos PHPs. O arquivo *config.php* contém uma classe responsável por obter as informações e configurações do banco. O arquivo *db.php* contém os códigos responsáveis necessários para conectar o banco com a aplicação, como é descrito na figura 16.

Figura 16 – Comunicação entre servidor e banco de dados

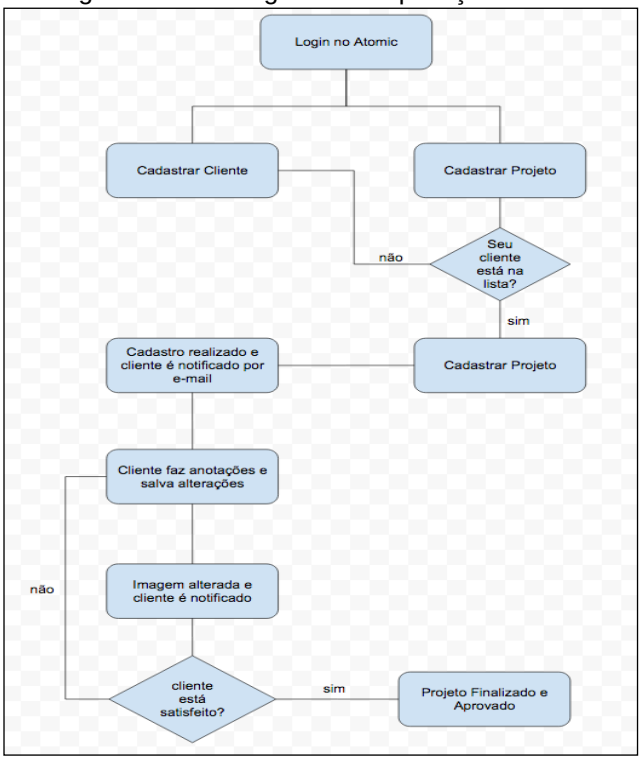
```php
<?php

require_once 'config.php';

class Database {



    //atributos
    public $conexao;

    public $tabela;
    public $valores;
    public $condicao;
    public $query;

    function Database(){
        $this->inicializa_database();
    }

    public function inicializa_database(){
        $cfg = new Config();

        $this->conexao = new mysqli($cfg->db_host, $cfg->db_usuario, $cfg->db_senha, $cfg->db_nome);

        if ($this->conexao->connect_error) {
            die("Erro na conexão com o banco de dados: " . $this->conexao->connect_error);
        }
        echo "INICIALIZOU DB !!!!<br>";
    }

    public function insere(){

        if (isset($this->tabela)){
            if (isset($this->tabela->valores)){
                echo "insere um<br>";
                //var_dump($this->tabela->valores);
                $campos   = implode(",", $this->tabela->campos);
                $valores  = implode(",", $this->tabela->valores);
                $tabela   = $this->tabela->nome_tabela;
                echo "Campos: $campos <br>";
                echo "Valores: $valores <br>";
                //$valores  = implode(",", $valores);
```

Fonte: Do autor.

6.4.2 Aplicação web

Para dar início ao desenvolvimento da aplicação Web, foi utilizado o fluxograma criado conforme a figura 17.

Figura 17 – Fluxograma da aplicação

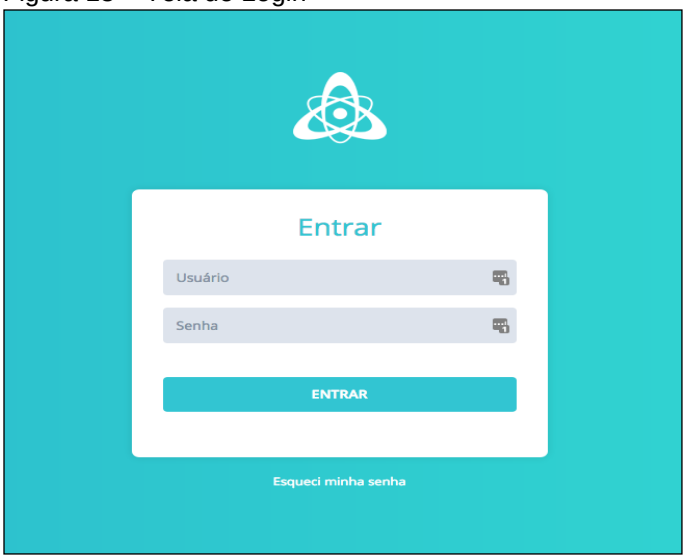


Fonte: Do autor.

6.4.2.1 Tela de Login

O arquivo inicial *index.php* que se encontra no diretório raiz, irá redirecionar para o endereço */Views/Usuario/login.php* onde se encontra a tela inicial de *login* (figura 18).

Figura 18 – Tela de Login



Fonte: Do autor.

A tela de *login* possui os campos de usuário e senha, quando o botão “ENTRAR” é clicado, conforme a figura 19, uma requisição ajax é disparado em */Recursos/js/atomic.js*), que chama o */Modulos/Usuario/login.php*.

Figura 19 – Ajax chamando a tela de login

```
372         $.ajax({
373           type: "POST",
374           url: "/atomic/Modulos/Usuario/login.php",
375           data: dados,        ──► na variavel dados contém o usuario e a senha
376           success: function( data )
377           {
378             if (data){        ──► sucesso
379               location.href = "/atomic/Views/Geral/dashboard.php";
380             }else{            ──► falha (não retornou nada)
381               $('.alert.alert-danger.display-hide span').html("Usuário ou senha incorretos.");
382               $('.alert.alert-danger.display-hide').show();
383             }
384
385           }
386         });
```

Fonte: Do autor.

São enviados como parâmetros o e-mail e a senha como descrito na figura 20, para que seja realizado um *select* com esse e-mail e senha, caso retorne registro significa que estava correta a combinação de usuário e senha, então ele cria uma sessão para o usuário ficar *logado* e então retorna o registro para o ajax que foi disparado na tela de *login*, logo se retornou algo ele redireciona o usuário para a tela da *dashboard*.

Figura 20 – Parâmetros do login

```
1  <?php
2      include '../../Tabelas/Usuario.php';
3
4      $usr          = new Usuario();
5      $usr->usuario = $_POST['email'];
6      $usr->senha   = $_POST['senha'];   ──► parametros
7      $logou        = $usr->seleciona();
8      if ($logou){
9          session_start();
10         $_SESSION['nome_usuario']  = $usr->usuario;
11         $_SESSION['senha_usuario'] = $usr->senha;
12         $idusr                     = $logou[0]["id"];
13         setcookie("idusr", $idusr, ((3600000) * 2), '/'); //3600000 = 1 hora
14     }
15     echo $idusr;   ──► retorno
```

Fonte: Do autor.

6.4.2.2 Cadastro de Projeto

No modal (janela secundária, também é conhecida como janela *pop-up*) de criação de novo projeto, é necessário preencher os campos, exceto os campos de selecionar, pois são previamente populados com as informações do banco de dados, conforme a figura 21.

Figura 21 – Campos de selecionar no modal de criação de novo projeto

Fonte: Do autor.

Quando o botão "CONTINUAR" é acionado, primeiramente um ajax é disparado conforme a figura 22, para o arquivo */Modulos/Projeto/cadastro_img.php* que recebe a imagem por parâmetro e da upload nela para o servidor e retorna sucesso ou falha para o ajax.

Figura 22 – Ajax de cadastro de imagem

(Recursos/js/atomic.js:408)

```
        var dados     = $(form).serialize();
            $form     = form,
            formData  = new FormData(),
            params    = $($form).serializeArray(),
            files     = $($form).find('[name="projeto_img"]')[0].files; //input type=file da imagem

        $.each(files, function(i, file) {
            formData.append('IMG'+i, file);
        });
    /* ajax pra dar upload na img */
        $.ajax({
          type: "POST",
          url: "/atomic/Modulos/Projeto/cadastro_img.php",
          dataType: "json",
          data: formData,
          processData: false,
          contentType: false
        }).done(function(imgname){
              /* ajax com retorno pra enviar os parametros do cadastro de projeto e retornar os dados necessários para o envio do email*/
              $.ajax({
                type: "POST",
                url: "/atomic/Modulos/Projeto/cadastro.php",
                dataType: "json",
                data: dados+"&imgpath="+imgname //dados = formulario | imgname = nome da imagem retornado pelo ajax do envio da imagem
              }).done(function(data){ //data é o retorno do ajax, que no caso seria um array com o cliente do projeto cadastrado
                                      // e o id do projeto novo que foi gerado no cadastro.
                  var obj_cliente = data[0];
                  var id_projeto  = data[1];
                  var email       = obj_cliente.email; //email do cliente do projeto

                  $('#projeto_email').val(email); //preenche o campo email do cliente da modal de envio do email
                  var url_projeto = $('#url_projeto').val(); //resgata a url do projeto padrão que já está carregada no html
                  $('#projeto_link').val(url_projeto + id_projeto); //preenche o campo link do projeto com a url padrao + o id do projeto
                  $('#NovoProjeto .close').click(); //fecha o modal

                  //aumenta o numero de projetos que já esta carregado no dashboard
                  var atual = parseInt($('#nro_projetos').attr('data-value'));
                  atual++;
                  $('#nro_projetos').attr('data-value', atual);
                  $('#nro_projetos').html(atual);

                  //abre o modal do envio do email
                  $('#ModalProjetoEmail').click();
                });
        });
```

(Modulos/Projeto/cadastro_img.php:5)

```
5   $imgpath = $_FILES['IMG0']; //imagem é recebida como parâmetro do ajax
6
7   $img = new Imagem();
8   $img->img_array = $imgpath;
9   $imagem = $img->upload_imagem(); //faz upload da imagem pro servidor e retorna sucesso ou falha
10
11  echo json_encode($imagem); //retorno pro ajax
```

Fonte: Do autor.

Assim que o retorno é fornecido para o ajax, ele mesmo dispara outro ajax para o arquivo */Modulos/Projeto/cadastro.php* contendo todos os dados preenchidos como parâmetros, ele recebe os parâmetros e armazena-os em variáveis e depois cria um objeto do tipo Projeto e seta todas as informações do projeto para as que foram enviadas como parâmetro e manda salvar o projeto no banco de dados com essas informações e retorna as informações necessárias para que seja enviado um e-mail para o cliente com o link do projeto e preenche o modal de envio de e-mail com estas informações, conforme a figura 23.

Figura 23 – Cadastro de projeto e modal de envio de e-mail

Fonte: Do autor.

Ao clicar no botão “ENVIAR” um ajax é disparado para */Modulos/Projeto/enviar_email.php* contendo os parâmetros de entrada necessários para o envio do e-mail que são o e-mail do cliente (destino), o link do projeto, e o tipo de evento (1 = inserção de novo projeto / 2 = o layout foi atualizado / 3 = o layout foi finalizado/aprovado) conforme definido na figura 24.

Figura 24 – Envio de e-mail para o cliente

Fonte: Do autor.

6.4.2.3 Comentários na imagem

Os comentários são escritos e devem ser salvos após clicar o botão "Salvar", isso fará com que fique salvo apenas no *localStorage* do navegador, será salvo apenas no *client-side*. Para salvar diretamente no banco de dados, quando o botão "SALVAR ALTERAÇÕES" for acionado, onde um ajax será disparado para o arquivo */Modulos/Projeto/atomic_salva_bd.php* e e este arquivo recebe o objeto com todos os comentários da imagem como parâmetro para que eles sejam salvos no banco de dados conforme a figura 25.

Figura 25 – Salvar comentários no banco de dados

```
function salvar_comentarios(){

var aryComentarios = anno.getAnnotations(); //pega a instância(objeto) dos comentários criado em tempo de execução
var aryDados = [];

for(var i=0; i < aryComentarios.length; i++){
  var obj = aryComentarios[i];
    var img = obj.src; //imgpath nome da imagem (caminho)
    var txt = obj.text; //comentário escrito em texto
    var geo = obj.shapes[0].geometry; //dados de posicionamento e geometria
    var hei = geo.height; //altura do comentario
    var wid = geo.width; //largura do comentario
    var coX = geo.x; //coordenada X do comentario
    var coY = geo.y; //coordenada Y do comentario
  aryDados.push({'id_projeto': PROJETO_ID
                 ,'texto': txt
                 ,'geo_height': hei
                 ,'geo_width': wid
                 ,'coord_x': coX
                 ,'coord_y': coY
                });
}
  $('#salvar_btn').attr('value', 'SALVANDO...');
  $.ajax({
    method: "POST",
    url: "atomic_salva_bd.php",
    data: {
           dados: aryDados //todos os comentarios
          ,id_projeto: PROJETO_ID
          ,id_img: IMAGEM_ID
        }
  }).done(function( msg ) {
    //sucesso
    $('#salvar_btn').attr('value', 'SALVAR ALTERAÇÕES');
    alert("Alterações salvas com sucesso.");
  });
}
```

Fonte: Do autor.

No momento em que a página do projeto é carregada, ou seja, no evento *onLoad* do documento, um ajax é disparado para o arquivo

/Modulos/Projeto/atomic_carrega_bd.php (figura 26), onde é enviado por parâmetro o id do projeto, o id do cliente e o id da imagem que está sendo carregada. Essas informações são enviadas para posterior recebimento e uso dos parâmetros para selecionar os comentários que condizem com tais parâmetros no banco de dados (figura 27) e retornar para o ajax e adicionar os comentários retornados na instância global do objeto *annotation* para que o mesmo apareça na imagem.

Figura 26 – Carregamento dos comentários

```
function carregar_comentarios(){

  $.ajax({
    method: "GET",
    url: "atomic_carrega_bd.php",
    data: { id_projeto: PROJETO_ID, id_cliente: CLIENTE_ID, id_img: IMAGEM_ID }
  }).done(function( data ) {
    var imgsrc = $(".annotatable").attr('src'); //source da imagem para identificar de que imagem faz parte o comentário
    var data = JSON.parse(data); //comentários retornados do banco de dados
    for(var i=0; i < data.length; i++){

      var Annotation = {
         src     : imgsrc
        ,context : document.location.href //url atual (contexto)
        ,text    : data[i].texto //texto do comentario
        ,shapes : [{
           type : 'rect' //padrão retangulo
          ,style : { }
          ,geometry : { x      : parseFloat(data[i].coordenada_x) //dados de geometria/posicionamento na imagem
                      , y      : parseFloat(data[i].coordenada_y)
                      , width  : parseFloat(data[i].geo_width)
                      , height : parseFloat(data[i].geo_height)
                    }
        }]

      };
      anno.addAnnotation(Annotation); //adiciona o comentário ao objeto global de comentários da tela e consequentemente na imagem também.
    }
  });
}
```

Fonte: Do autor.

Figura 27 – Carregamento dos comentários 02

```
    include '../../Tabelas/Comentario.php';

    $id_projeto         = (is_numeric($_GET['id_projeto'])) ? $_GET['id_projeto'] : 0; //verifica se o id do projeto é numeral.
    $id_img             = $_GET['id_img']; //id da imagem que foi carregada
    $cmtdel             = new Comentario();
    $cmtdel->id_projeto = $id_projeto; //id do projeto que foi carregado
    $cmtdel->id_img     = $id_img;
    $aryComentarios     = $cmtdel->seleciona(); //carrega os comentarios

    echo json_encode($aryComentarios); //retorna os comentarios pro resultado do ajax
```

Fonte: Do autor.

6.4.2.3 Cadastro de Usuário

O cadastro de usuário é realizado no botão "Novo usuário" e após os dados serem preenchidos corretamente e o botão "Cadastrar" for acionado, um ajax será disparado para o arquivo */atomic/Modulos/Projeto/cadastro_img.php* com o intuito de dar upload na imagem da foto do cliente para o banco de dados. O *upload* da imagem funciona exatamente da mesma forma que o upload da imagem de cadastro de projeto, conforme a figura 28.

Figura 28 – Upload no cadastro de usuário

```
18          var dados    = $(form).serialize();
19              $form    = form,
20              formData = new FormData(),
21              params   = $($form).serializeArray(),
22              files    = $($form).find('[name="usuario_img"]')[0].files; //campo de upload da imagem
23
24            $.each(files, function(i, file) {
25                formData.append('IMG'+i, file); //coloca a imagem dentro do formData
26            });
27            $.ajax({
28                type: "POST",
29                url: "/atomic/Modulos/Projeto/cadastro_img.php",
30                dataType: "json",
31                data: formData, //parametros a enviar
32                processData: false,
33                contentType: false
34            }).done(function(imgname){
35                $.ajax({
36                    type: "POST",
37                    url: "/atomic/Modulos/Usuario/cadastro.php",
38                    dataType: "json",
39                    data: dados+"&usuario_img="+imgname
40                    }).done(function(data){
41                        alert("Usuário cadastrado com sucesso!");
42                    });
43            });
```

Fonte: Do autor.

Caso o upload retorne com sucesso, outro ajax é disparado para */atomic/Modulos/Usuario/cadastro.php*, enviando todos estes dados do formulário e o nome da imagem que retornou no ajax do upload da imagem como parâmetros, que posteriormente são recebidos e tratados para que seja criada uma nova instância de usuário e que a mesma receba estes parâmetros e seja encaminhada para ser salva no banco de dados, como demonstrado na figura 29.

Figura 29 – Cadastro de usuário

```
5       $login_usuario = $_POST['usuario_login']; //login(email)
6       $senha_usuario = $_POST['usuario_senha']; //senha
7       $email_usuario = $_POST['usuario_email']; //e-mail
8       $nome_usuario  = $_POST['usuario_nome'];  //nome do usuario
9       $foto_usuario  = $_POST['usuario_img'];   //nome da imagem do cliente retornada pelo ajax anteriormente
10
11      //Criada uma nova instância de usuario
12      $usuario            = new Usuario();
13
14      //Setando parâmetros para a nova instância.
15      $usuario->login     = $login_usuario;
16      $usuario->senha     = $senha_usuario;
17      $usuario->email     = $email_usuario;
18      $usuario->nome      = $nome_usuario;
19      $usuario->imgpath   = $foto_usuario;
20
21      //Mandando a instância ser salva no banco de dados.
22      $resultado = $usuario->salva();
23
24      //Resultado do processo.
25      echo $resultado;
```

Fonte: Do autor.

6.4.2.3 Cadastro de Cliente

No cadastro de cliente, após os dados serem preenchidos e o botão "Cadastrar" for clicado, primeiramente um ajax é disparado para a tela */atomic/Projetos/cadastro_img.php* para dar upload na imagem. Caso o upload tenha tido sucesso, ele retorna o nome da imagem e com ele juntamente com os parâmetros do formulário dispara outro ajax para o arquivo */atomic/Modulos/Cliente/cadastro.php* contendo todos estes parâmetros e cria uma nova instância de Cliente e seta os parâmetros recebidos nessa nova instância e manda ela para ser salva no banco de dados, como na figura 30.

Figura 30 – Cadastro de cliente

```
14              var dados     = $(form).serialize();
15                  $form     = form,
16                  formData = new FormData(),
17                  params    = $($form).serializeArray(),
18                  files     = $($form).find('[name="cliente_foto"]')[0].files;
19
20                $.each(files, function(i, file) {
21                    formData.append('IMG'+i, file);
22                });
23            $.ajax({
24                  type: "POST",
25                  url: "/atomic/Modulos/Projeto/cadastro_img.php",
26                  dataType: "json",
27                  data: formData,
28                  processData: false,
29                  contentType: false
30                }).done(function(imgname){
31            $.ajax({
32              type: "POST",
33              url: "/atomic/Modulos/Cliente/cadastro.php",
34              dataType: "json",
35              data: dados+"&cliente_foto="+imgname
36              }).done(function(data){
37                alert("Cliente Cadastrado com sucesso!");
38
39                //atualiza o numero de clientes do dashboard sem atualizar a pagina
40                var atual = parseInt($('#nro_clientes').attr('data-value'));
41                atual++;
42                $('#nro_clientes').attr('data-value', atual);
43                $('#nro_clientes').html(atual);
44
45              });
46          });
```

Fonte: Do autor.

6.4.2.4 Carregamento e endereçamento das imagens

Na tela de edição do projeto, a imagem da versão do projeto atual é carregada. O arquivo que se encontra em */Modulos/Projeto/atomic.php* que contém esses parametros, carrega as informações do banco de dados e define o caminho da imagem através da pasta de upload que foi definido no arquivo *config* e também o nome da imagem que provém do campo *imgpath* da tabela imagens direto do banco de dados associada ao projeto e a versão em questão (figura 31).

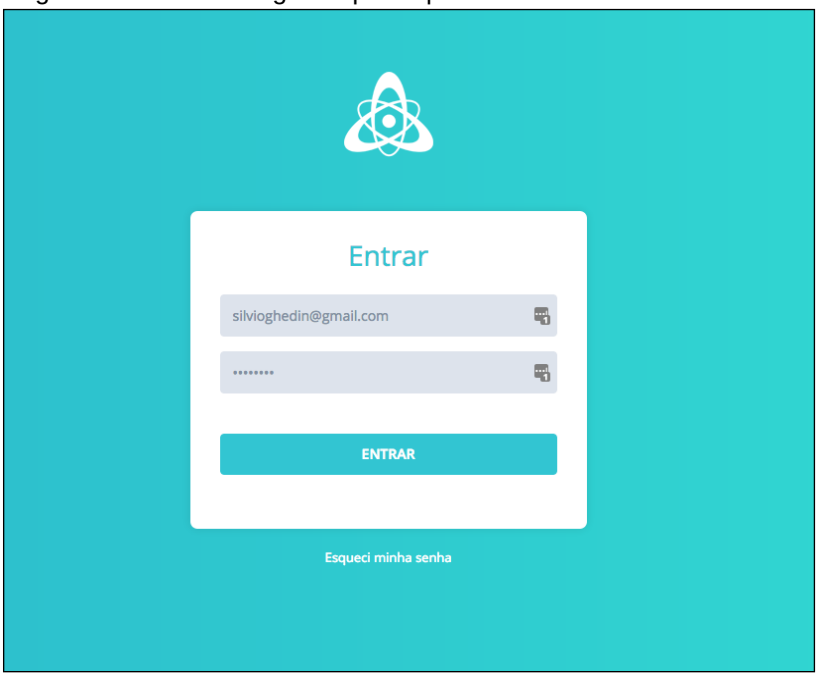
Figura 31 – Endereçamento das imagens

Fonte: Do autor.

6.4.3 Funcionamento do sistema

O início do fluxo do protótipo inicia-se na tela de *login*, conforme a figura 32. Nessa mesma tela é possível solicitar que a senha seja enviada novamente para o usuário em caso de esquecimento. O *login* é efetuado atráves do preenchimento dos dados de e-mail, senha e do botão "ENTRAR".

Figura 32 – Tela de login do protótipo

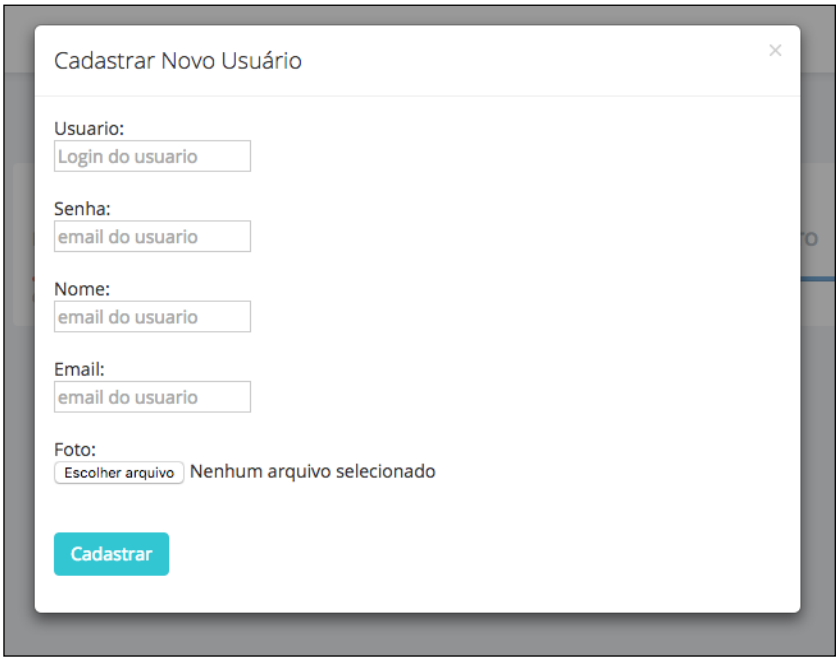


Fonte: Do autor.

Após realizar o *login* com sucesso, é possível realizar três tipos de cadastro, o cadastro de usuário (figura 33), cliente e projeto.

Os usuários serão responsáveis por cadastrar posteriormente clientes e novos projetos, esses usuários serão as pessoas que irão fazer a interação direta com o cliente, sem nenhuma interrupção de terceiros.

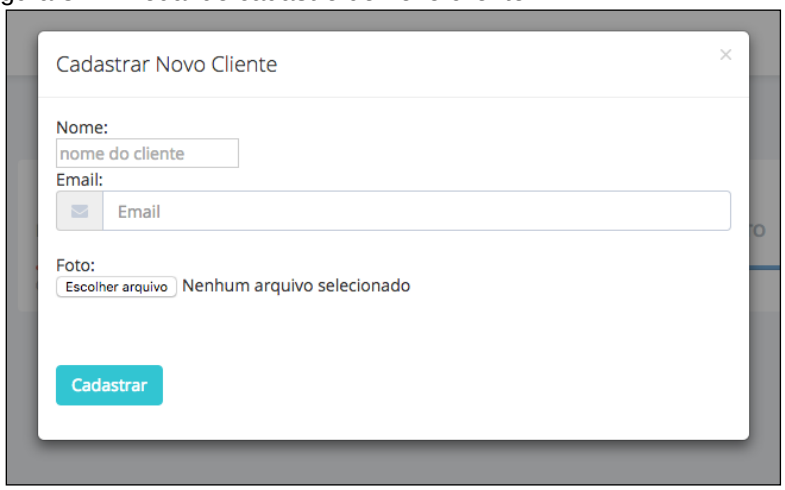
Figura 33 – Modal de cadastro de novo usuário



Fonte: Do autor.

Ao clicar no botão "Novo Cliente" o usuário será levado a um modal onde é solicitado apenas três informações, nome do cliente, e-mail e uma foto para identificação (figura 34).

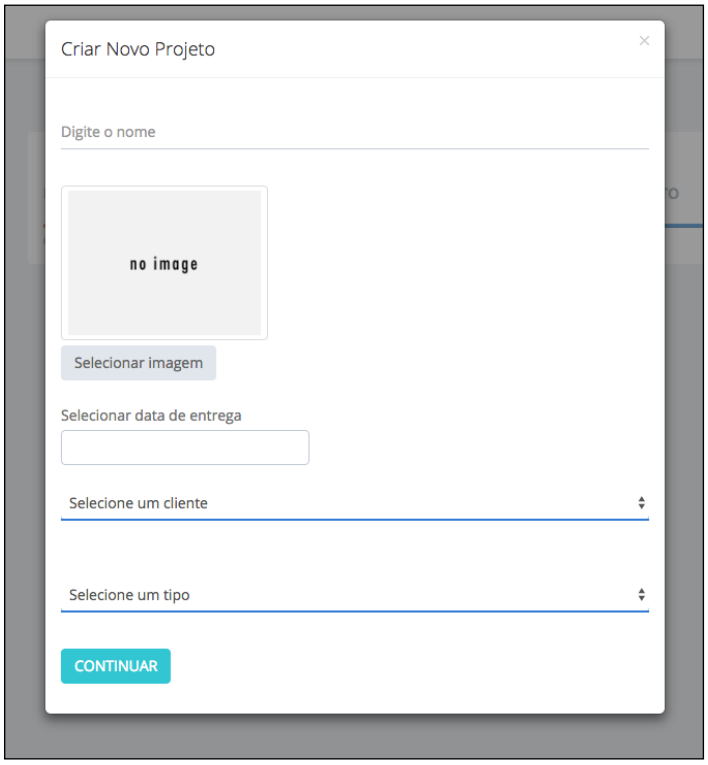
Figura 34 – Modal de cadastro de novo cliente



Fonte: Do autor.

Para cadastrar um novo projeto é importante que o cliente que será atrelado a este projeto tenha sido previamente cadastrado. Após clicar em "Novo Projeto", irá aparecer na tela um *modal* com o cadastro, conforme a figura 35.

Figura 35 – Modal de cadastro de novo projeto



Fonte: Do autor.

Após clicar no botão "CONTINUAR", o projeto é criado e automaticamente um e-mail é enviado para o cliente para notifica-lo com o link do projeto e com as instruções iniciais. O cliente irá clicar no link e será redirecionado para a tela de projeto, onde ele irá visualizar o projeto dele de forma mais realista, diretamente no seu navegador. Isso fará com que ele não apenas baixe uma imagem, mas fará com que ele tenha uma noção real do protótipo do seu projeto e um melhor entendimento e uma experiência mais realista (figura 36).

Figura 36 – Tela de projeto

Fonte: Do autor.

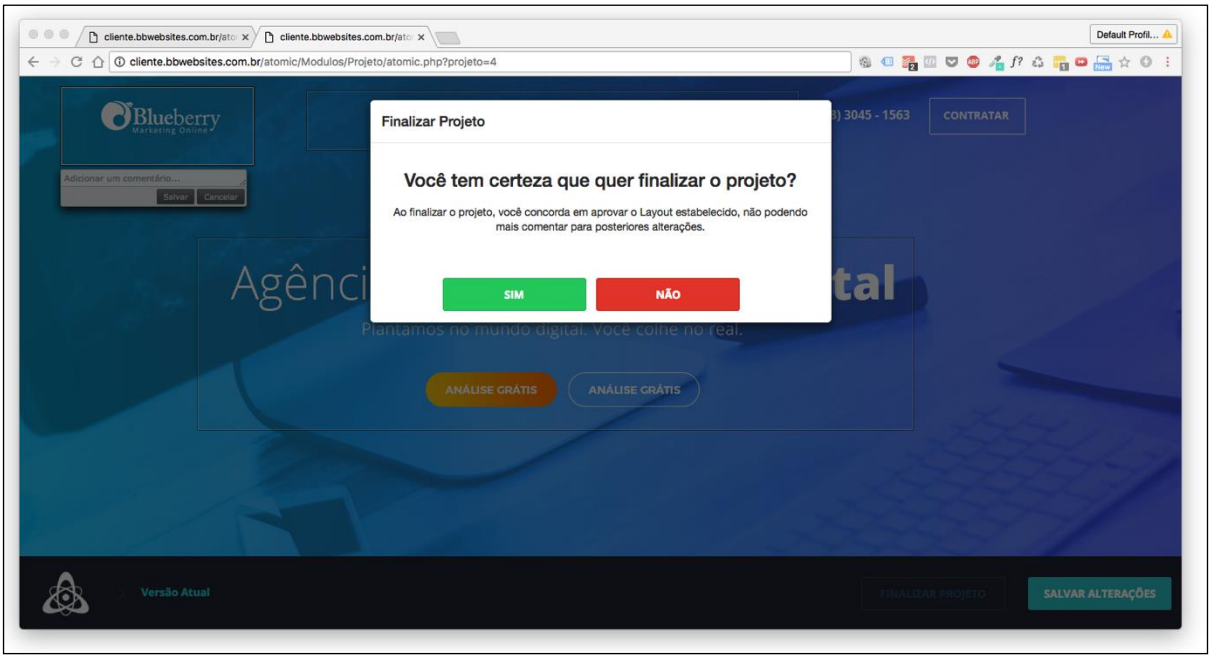
O cliente poderá fazer inúmeros comentários, é necessário utilizar o mouse e selecionar a área desejada, um campo para acrescentar o comentário irá aparecer e após isto é possível clicar em “Salvar” ou “Cancelar” (figura 37).

Figura 37 – Fazer comentário

Fonte: Do autor.

Após realizar os comentários, o cliente tem duas opções, clicar no botão "SALVAR ALTERAÇÕES" e o usuário responsável por esse projeto será notificado e posteriormente irá fazer outra versão com as alterações solicitadas pelo cliente, ou clicar em "FINALIZAR PROJETO", onde o cliente aceita que ao finalizar o projeto, ele concorda em aprovar o projeto estabelecido, não podendo mais comentar para posteriores alterações (figura 38).

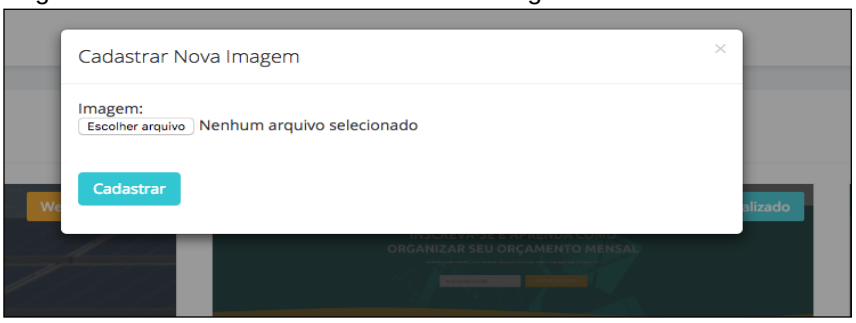
Figura 38 – Finalizar projeto



Fonte: Do autor.

O desenvolvedor é notificado via e-mail quando o cliente realiza comentários na imagem ou quando finaliza o projeto, para continuar o fluxo e criar outra versão é necessário realizar o upload de uma nova imagem clicando em "Atualizar Imagem", onde irá surgir um modal com a opção de cadastrar uma nova imagem. Esse passo é feito até o cliente decidir finalizar projeto (figura 38).

Figura 39 – Modal de cadastro de novo imagem



Fonte: Do autor.

6.5 RESULTADOS OBTIDOS

A partir dos requisitos iniciais levantados neste trabalho, foi possível compilar um material de qualidade sobre os temas abordados na implementação do sistema proposto, sendo esse o principal objetivo. Foram destacados no projeto os principais motivos pelo insucesso e comprometimento dos projetos realizados.

Também foram estudados todos os métodos mais modernos e utilizados para o desenvolvimento da ferramenta protótipo.

Não foi encontrado trabalhos iguais ou similares utilizando o *plugin* JavaScript para anotação de imagem, deixando na fase inicial do trabalho um certo receio de como seria seu real funcionamento, porém durante o decorrer do trabalho ficou mais claro o seu funcionamento.

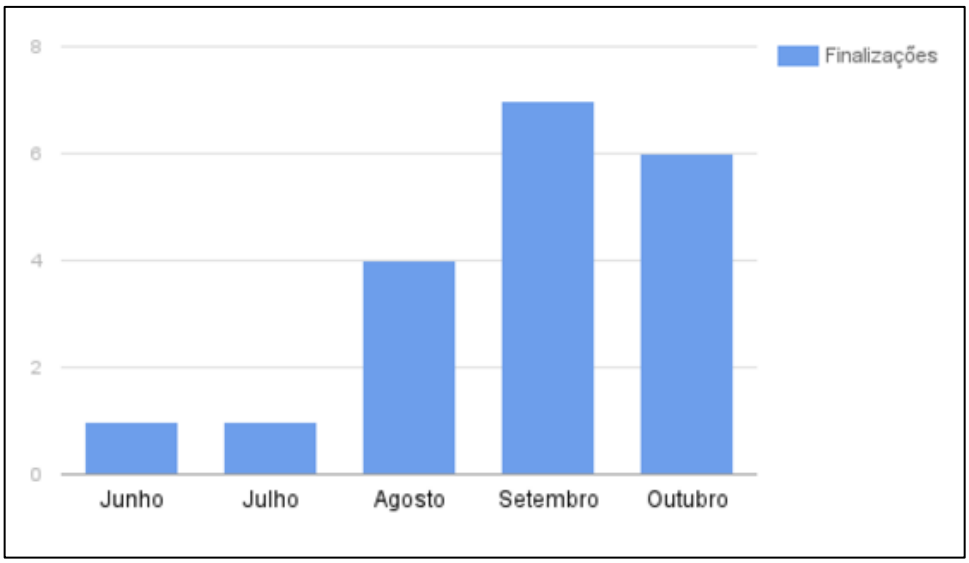
A homologação foi realizada na empresa de marketing online Blueberry e foi feita com dez clientes reais. Com a utilização dos clientes, ficou evidente a eficácia da ferramenta desenvolvida e no auxílio da comunicação entre o cliente e o desenvolvedor.

Um dos problemas recorrentes que acontecia antes da implementação da ferramenta, era a falta de entendimento com o protótipo do projeto. O cliente recebia uma imagem por e-mail, devido ao seu tamanho ser grande, quando o ele abria com seu programa padrão de imagens, ela se ajustava proporcionalmente, o que deixava os elementos menores e uma visão errônea do protótipo. Com a possibilidade de visualizar o protótipo em seu navegador, o índice de problemas reduziu completamente.

O tempo da etapa de validação de requisitos foi diminuída drasticamente, a média de finalização de projeto na ferramenta é de 15 dias, antigamente sem a ferramenta a média era de 25 dias. O processo ficou mais ágil e foi constatado que o retrabalho diminuiu consideravelmente, pois como antes os clientes não conseguiam visualizar de forma clara o projeto, os requisitos solicitados não podiam ser validados como é feito com a ferramenta.

Outro benefício com a ferramenta utilizada foi o número de projetos finalizados. Os testes se iniciaram no final de julho e conforme a figura 39, houve um aumento nos projetos finalizados.

Figura 40 – Modal de cadastro de novo imagem



Fonte: Do autor.

Os resultados obtidos foram satisfatórios com relação aos objetivos apresentados. O objetivo principal deste trabalho de conclusão de curso, se trata da construção de uma ferramenta capaz de validar requisitos em projetos Web, conforme descrito na etapa de homologação, ambos foram bem-sucedidos.

7 CONCLUSÃO

Criar um projeto web requer uma grande variedade de ferramentas como observado durante o desenvolvimento do trabalho. Por meio dos assuntos estudados como engenharia de software, desenvolvimento web e engenharia de requisitos, os objetivos propostos no início do projeto foram alcançados.

As ferramentas utilizadas durante a execução do projeto e o planejamento detalhado, auxiliaram e ajudaram a evitar possíveis erros, evitando o retrabalho de algumas etapas. Afim de obter um resultado satisfatório no projeto, foi imprescindível a realização de um estudo bibliográfico para compreender quais conhecimentos eram necessários para atingir os objetivos propostos.

Com o estudo bibliográfico feito aliado com a metodologia do trabalho, foi possível alcançar o objetivo geral e os objetivos específico, resultando em uma ferramenta protótipo que valida requisitos que atende as necessidades do cliente e do desenvolvedor.

O projeto torna-se importante, pois permite solucionar um problema vigente e contribui com o envolvimento do usuário com o projeto, e também, sua pesquisa contribuiu de grande fonte de conhecimento adquirido.

Durante a fase de testes ocorreu alguns problemas e dificuldades durante o desenvolvimento da aplicação, como o salvamento dos comentários na imagem, sendo assim necessário um esforço a mais e a correção da requisição ajax que estava com problemas.

Outros problemas foram sendo encontrados durante o desenvolvimento da aplicação, mas foram se resolvendo a medida que os testes se aprofundavam.

A ferramenta protótipo foi homologada dentro da empresa Blueberry Marketing Online, onde foi utilizada em dez projetos com clientes reais. Mesmo após a fase de testes, enquanto a ferramenta foi utilizada nos clientes, foi possível fazer ajustes e aprimorando ela para ter um resultado melhor.

O feedback com os clientes que utilizaram a ferramenta foi satisfatório. Eles puderam visualizar o projeto deles de uma forma mais real e diretamente do navegador deles, sendo assim, os requisitos levantados previamente foram vistos de forma mais real e os próprios clientes visualizaram o que eles haviam solicitado.

Foi dado a possibilidade de fazer os comentários no projeto, e a interação dos clientes com a ferramenta foi boa. As solicitações foram mais coesas, pois os clientes conseguiam visualizar exatamente o que eles solicitaram e tinham a possibilidade de comparar com a versão anterior.

Durante o processo de execução do projeto, alguns pontos surgiram como forma de aprimoramento, possibilitando a continuação do projeto para novas pesquisas nesta área. Alguns destes pontos são:

a) inserir comentários na própria aplicação, sem que seja feito *upload* de uma imagem;
b) implementar a opção de visualização da aplicação para dispositivos móveis;
c) criar um canal na própria ferramenta de chat para dúvidas e para ter um controle maior dos requisitos solicitados.

Contudo, podemos concluir que o trabalho obteve o resultado esperado, atingindo todos os itens propostos com a conclusão do desenvolvimento da ferramenta protótipo para validação de requisitos em projetos Web.

REFERÊNCIAS

AGNER, Luiz. **Ergodesign e arquitetura da informação**: trabalhando com o usuário. Rio de Janeiro: Quartet, 2006.

APPLETON, B. **Patterns and Software:** essential concepts and terminology. Disponível em: <www.enteract.com/~bradapp/docs/>. Acesso em: 05 mar 2016.

ARMITAGE, J. **Are agile methods good for design?** Interactions, New York, 2004.

AURUM, A., WOHLIN, C., **Engineering and Managing Software Requirements**, Springer-Verlag, 2005.

BARBOSA, Simone Diniz Junqueira; SILVA, Bruno Santana da. **Interação humano-computador**. 10. ed. Rio de Janeiro: Campus, 2010. 384 p.

BÄUMER, D. et al, **User interface prototyping**: concepts, tools and experience. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 18., 1996, Berlin. Proceedings...Los Alamitos, CA: IEEE Computer Society Press, 1996. p.532 541.

BECK, K. E**mbracing Change with Extreme Programming.** IEE Computer, v. 32, n. 10, 1999, p. 70-78. **Extreme Programming Explained.** Reading, Mass,: Addison-Wesley, 2000.

BERGHEL, H. **New wave prototyping**: the use and abuse of vacuous prototypes interactions, New York, v.1, n.2, p.49-54, Apr.1994. Disponível em: <http://www.acm.org/~hlb/publications/new-wave/new-wave.html> Acesso em: 26 mar. 2016.

BLOMKVIST, S. **Human-Centered Software Engineering**: Integrating Usability in the Software Development Lifecycle. Springer, Sweden, 2005.

BROWN, Daniel M. **Communicating design**: developing web site documentation for design and planning. Berkeley: New Riders. 2007.

BURGE, J., BROWN, D. **Reasoning with design rationale**. In: Proceedings of 6 th International Conference on Artificial Intelligence in Design, 2000.

CASTELEYN, Sven et al, **Engineering Web Applications.** Berlin Heidelberg: Springer, 2009. 349 p.

CATANI, M.B.; BIERS, D.W. **Usability evaluation and prototype fidelity**: users and usability professionals. In: ANNUAL MEETING OF THE HUMAN FACTORS AND ERGONOMICS SOCIETY, 42., 1998. Proceedings: Disponível em: <http://www.humanfactors.com/download/dec98.asp> Acesso: 26 mar. 2016.

CERI, S.; FRATERNALI, P.; BONGIO, A., **Web modeling language (WebML):** a Modeling Language for Designing Web Sites. Computer Networks-The International Journal of Computer and Telecommunications Networking, v.33, n.1-6, p. 137-157, Jun. 2000. Disponível em: Acesso em: 11 mai. 2016

CHRISTEL, M.; KANG, K. **Issues in Requirements Elicitation (CMU/SEI-92-TR-012, ADA258932)**. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992. Disponível em: < http://www.sei.cmu.edu/reports/92tr012.pdf >, Acesso em: 30 mar. 2016.

COCKBURN. A. **Agile Software Development.** Reading, Mass.:Addison-Wesley, 2001. **Crystal Clear:** A human-Powered Methodology for Small Teams. Boston: Addison-Wesley, 2004.

COHN, M. **Succeeding with Agile:** Software Development Using Scrum. Boston: Addison-Wesley, 2009.

CONALLEN, J. **Modeling web application architectures with UML**. Communications of the ACM, v.42, n.10, 1999.

CONKLIN, E., BURGESS-YAKEMOVIC, KC. **A Process-Oriented Approach to Design Rationale**. Design Rationale: Concepts, Techniques, and Use. By John M. Carroll, Thomas P. Moran. Lawrence Erlbaum Associates, p. 393-427, 1996.

COOPER, Alan; REIMANN, Robert; CRONIN, Davis. **About Face 3**: The Essentials of Interaction Design. Indianopolis: Wiley Publishing, 2007. 610 p.
CYBIS, W.; BETIOL, A. H.; FAUST, R. **Ergonomia e Usabilidade: Conhecimentos, Métodos e Aplicações**. São Paulo: Novatec, 2007

DAVIS, A. M. **201 Principles of Software Development**. McGraw-Hill, New York. 1995.

DAVIS, A.; BERSOFF, E.; COMER, E. **A Strategy for Comparing Alternative Software Development Life Cycle Models**. IEEE Transactions on Software Engineering, New York, v.14, n.10, p.1453–1461, 1988.

DIAS, Claudia. **Usabilidade na Web:** Criando portais mais acessíveis. 2. ed. Rio de Janeiro: Alta Books Ltda, 2007. 296 p.

DIX, Alan; FINLAY, Janet; ABOWD, Gregory D.; BEALE, Russel. **Human-Computer Interaction**. New York, Prentice Hall: 1998.

DZENDZIK, Isolete Teresinha. **Processo de Desenvolvimento de Web Sites com Recursos da UML.** 2004. 182 f. Dissertação (Mestrado) - Curso de Computação Aplicada, Inpe, São José dos Campos, 2005. Cap. 5.

EIS, Diego; FERREIRA, Elcio. **HTML5 e CSS3 com Farinha e Pimenta**. São Paulo: Lulu.com, 2012.

FIFTEENTH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 15., 2003, p. 5-6, San Francisco. **Revealing Web User Requirements through e-Prototyping. Germany**: University Of Hamburg Vogt-kolln-str, 2003. 8 p.

FISCHER, G., LEMKE, A., McCALL, R., MORCH, A. **Making Argumentating Serve Design**. Design Rationale: Concepts, Techniques, and Use. By John M. Carroll, Thomas P. Moran. Lawrence Erlbaum Associates, p. 267-293, 1996.

FLANAGAN, David. **JavaScript:** The Definitive Guide. 6. ed. Sebastopol: O'reilly, 2011. 1078 p.

FLOYD, C. **A Systematic Look at Prototyping**. In: APPROACHES TO PROTOTYPING, 1984, Berlin. Proceedings... [S.l.]: Springer-Verlag, 1984. p.1–18.

FOURTH INTERNATIONAL CONFERENCE ON WEB ENGINEERING, 4., 2004, Glasgow. **Evaluation of Commercial Web Engineering Processes.** Glasgow: Icwe, 2004. 170 p.

FOWLER, M. **UML Distilled:** applying the standard object modeling language. Reading: Addison-Wesley, 1997.

GAMMA, E. **Design-Patterns:** elements of reusable object-oriented software. Reading, MA: Addison-Wesley, 1995.

GARRETT, Jesse James; RIDERS, New. **The Elements of User Experience**: User-Centered Design for the Web. Berkeley: New Riders, 2011. 172 p.

GASSTON, Peter. **The Book of CSS3:** A Developer's Guide to the Future of Web Design. San Francisco: no Starch Press, 2011. 304 p.

GONÇALVES, Rodrigo Franco. **Uma abordagem sistêmica para o processo de produção em engenharia web, na fase de concepção.** 2010. 140 f. Tese (Doutorado) - Curso de Engenharia de Produção, Escola Politécnica da Universidade de São Paulo, São Paulo, 2010.

GRAHAM, DOROTHY. **Requirements and Testing**: Seven Missing-Link Myths; IEEE Software, Vol. 19, No5, pp. 15-17. 2002.

GRUBER, T., RUSSEL, M. **Generative Design Rationale: Beyond the Record and Replay Paradigm**. Design Rationale: Concepts, Techniques, and Use. By John M. Carroll, Thomas P. Moran. Lawrence Erlbaum Associates, p. 323-349, 1996.

HIGHSMITH, J. A. **Adaptative Software Development**: A Collaborative Approach to Managing Complex Systems. Nova York: Dorset House, 2000.

INCE, D.; HEKMATPOUR, S. **Software Prototyping: progress and prospects**. Information and Software Technology, [S.l.], v.29, n.1, p.9–14, 1987.

INOGRAD, Terry et al, **Bringing Design to Software.** New York: Acm Press, 1996. 352 p.

ISAKOWITZ, T., STOHR E., BALASUBRAMANIAN P. RMM: **A Methodology for Strutured Hypermidia Design**. Communications of ACM, v. 38, no. 8, ago./1995. Jeenicke, M., Bleek, W-G., and Kliscewski, R. (2003) "**Revealing Web User Requirements through e-Prototyping**", Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering: SEKE 03. São Francisco, EUA.

KRUG, Steve. **Don't Make Me Think**: A Common Sense Approach to Web Usability. 2. ed. Berkeley: New Riders, 2005. 216 p.

LAWSON, Bruce; SHARP, Remy. **Introducing HTML5**. 2. ed. Berkeley: New Riders, 2012. 295 p.

LERDORF, Rasmus; TATROE, Kevin. **Programming PHP.** 3. ed. Sebastopol: O'reilly, 2013. 600 p.

LOWDERMILK, Travis. **Design Centrado no Usuário**. São Paulo: Novatec, 2013. 182 p.

LOWE, David; EKLUND John. (2002). **Client Needs and the Design Process in Web Projects**. Web Engineering Track of the WWW2002 Conference.

MACLEAN, A., YOUNG, R., BELLOTI, V., MORAN, T. **Questions, Options, and Criteria: Elements of Design Space Analysis**. Design Rationale: Concepts, Techniques, and Use. By John M. Carroll, Thomas P. Moran. Lawrence Erlbaum Associates, p. 53-105, 1996.

MACLEAN, A., YOUNG, R., MORAN, T. **Design rationale: the argument behind the artifact.** Proceedings of the SIGCHI conference on Human factors in computing systems: Wings for the mind, p. 247-252. Março, 1989.

MAYHEW, P.; DEARNLEY, P. **An Alternative Prototyping Classification**. The Computer Journal, [S.l.], v.40, n.6, p.481–484, 1987.

MENEZES, Luis C. de M. **Gestão de Projetos**, São Paulo: Atlas, 2001.

MIRANDA, Ana Paula. **Website com Dreamweaver, CSS3 e HTML5**. Florianópolis: Visual Books, 2012. 224 p.

MIRANDA, LEONEL. **Gestão de Requisitos**: Engenharia de Software, No4, pp.2, http://www.engenharia-software.com. 2003.

MORVILLE, Peter; ROSENFELD, Louis. **Information Architecture for the World Wide Web**: Designing Large-Scale Web Sites. 3. ed. Sebastopol: O'reilly, 2006. 504 p.

NIEDERAUER, Juliano. **Desenvolvendo Websites com PHP:** Aprenda a criar Websites dinâmicos e interativos com PHP e bancos de dados. 2. ed. São Paulo: Novatec, 2011. 304 p.

NIELSEN, Jakob. **Projetando Websites**: Designing Web Usability. Rio de Janeiro: Campus, 2000. 416 p.

NIELSEN, Jakob; TAHIR, Marie. **Homepage Usabilidade**: 50 Web Sites Desconstruídos. Rio de Janeiro: Campus, 2002. 336 p.

NORMAN, Don. **The Design of Everyday Things**. Philadelphia: Basic Books, 2013. 347 p.

OVERMYER, S. P. **What is Different about Requirements Engineering for Web Sites?** Requirement Engineering Journal, v. 5, no 1, pp 62-65, 2000.

PALMER, S. R.; FELSING, J. M. **A Practical Guide to Feature-Driven Development**. Englewood Cliffs, NJ: Prentice Hall, 2002.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software:** fundamentos, métodos e padrões. São Paulo: Ltc, 2000. 260 p.

PFLEEGER, S.L., **Engenharia de Software**: Teoria e Prática, São Paulo: Prentice Hall, 2a edição, 2004.

PINHEIRO, F. **Requirements Honesty**; Requirements Eng, Springer-Verlag; Vol. 8, 2003.

PMI (PROJECT MANAGEMENT INSTITUTE). **Um Guia do Conhecimento em Gerenciamento de Projetos** (Guia PMBOK®) – Quinta Edição. Newtown Square: Project Management Institute, 2013.

POHL, Klaus; RUPP, Chris. **Requirements Engineering Fundamentals:** A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant. Sebastopol: Rocky Nook Computing, 2011. 178 p.

PREECE, J.; ROGERS, Y.; SHARP, H.; BENYON, D.; HOLLAND, S.; CAREY, T. **Human-Computer Interaction**. EUA: Addison Wesley, 1994.

PRESSMAN, R. **Engenharia de Software**: Uma Abordagem Profissional. 7ª Ed., São Paulo: McGraw-Hill, 2011.

REZENDE, Denis Alcides ; Denis Alcides Rezende , **Engenharia de software e sistemas de informação**. 3. Ed. rev. Ampl -- Rio de Janeiro: Brasport. 2005.

ROGERS, Yvonne; SHARP, Helen; PREECE, Jennifer. **Design de Interação**: além da interação humano-computador. 3. ed. Porto Alegre: Bookman, 2013. 585 p. Tradução Isabela Gasparini.

SAMPAIO, Américo; VASCONCELOS, Alexandre; SAMPAIO, Pedro R. Falcone. **Design and Empirical Evaluation of na Agile Web Engineering Process**. In 18th Simpósio Brasileiro de Engenharia de Software, SBES 2004.

SANTOS, S. S.; **Tratamento da informação em ambientes digitais**: capacitação do bibliotecário para atuar como arquiteto de informação para a Web. Porto Alegre, 2009. 127f. Monografia (Graduação em Biblioteconomia). Universidade Federal do Rio Grande do Sul – UFRS. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/37533/000717875.pdf?sequence=1>; Acesso em 22 mar. 2016.

SCHWABE, D.; PONTES, R.; MOURA, I. OOHDM-Web: **An Environment for Implementation of Hypermedia Applications in the WWW**. ACM SigWEB Newsletter, v. 8, no 2, pp. 18-34, jun./1999.

SCHWABER, K. **Agile Project Management with Scrum.** Seattle: Microsoft Press, 2004.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum.** Englewood Cliffs, NJ: Prentice Hall, 2001.

SHNEIDERMAN, Ben; PLAISANT, Catherine. **Designing the User Interface:** Strategies for Effective Human-Computer Interaction. 5. ed. Maryland: Addison Wesley, 2009. 624 p.

SILVA, Maurício Samy. **JavaScript:** Guia do Programador. São Paulo: Novatec, 2010. 608 p.

_____. **CSS3:** Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das css3. São Paulo: Novatec, 2012. 496 p.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011. 529 p.

STANDISH, Group International. Disponível em:<https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. Acesso em: 05 mar. 2016.

STAPLETON,J. **DSDM Dynamic-Systems Development Method.** Harlow, Reino Unido: Addison-Wesley, 1997. DSDM: Business Focused Development, 2 ed.Harlow, Reino Unido: Pearson Education, 2003.

STUTTARD, Dafydd; PINTO, Marcus. **The Web Application Hacker's Handbook:** Finding and Exploiting Security Flaws. 2. ed. Indianapolis: Wiley Publishing, 2011. 878 p.

TIDWELL, Jenifer. **Designing Interfaces**: Patterns for Effective Interaction Design. 2. ed. Canada: O'reilly, 2011. 547 p.

UNGER, Russ; CHANDLER, Carolyn. **O Guia para Projetar UX**. Rio de Janeiro: Alta Books, 2009. 268 p.

YOUNG, Ash. **Wireframes are not enough**. Blog “The Web Design ThinkTank”. 14/06/2011. Disponível em http://www.evoluted.net/thinktank/web-design/wireframes-are-not-enough. Acesso em: 21 de mar. 2016.

APÊNDICE

APÊNDICE A – ARTIGO CIENTÍFICO

Validação de requisitos no modelo de prototipação em projetos web: construção de uma ferramenta

Silvio Polli Ghedin[1], Gilberto Vieira da Silva[2]

[1]Acadêmico do Curso de Ciência da Computação – Universidade Acadêmica de Ciências, Engenharia e Tecnologias (UnaCet) – Universidade do Extremo Sul Catarinense (UNESC) Av. Universitária, 1105 – Bairro Universitário – Criciúma – SC- Brasil

[2] Professor do Curso de Ciência da Computação – Universidade Acadêmica de Ciências, Engenharia e Tecnologias (UnaCet) – Universidade do Extremo Sul Catarinense (UNESC) Av. Universitária, 1105 – Bairro Universitário – Criciúma – SC- Brasil

silvioghedin@gmail.com, gilbertovieirasilva@hotmail.com

Abstract. *With the polarization of Internet and the growth of users using it, big companies and softwares saw both the opportunity and the need to migrate to the Web. The main factors of failure and commitment in these projects are: lack of user involvement, changes in the requisites or problems in definition. A tool was created to meet the needs of these current problems. During the prototyping, this new tool is used and aims at validating requisites and provide a better interaction between the cliente and the developer. The tool was used in a real environment at Blueberry – Agência de Marketing Digital, allowing such requisites to be validated with real clientes and improving communication with them. The results gathered in the development were satisfactory and there was an improvement of up to 40% in the completion of the projects developed.*

Resumo. *Com a popularização da internet e crescimento dos usuários, grandes empresas e softwares viram a necessidade de migrarem para a Web. Os principais motivos do insucesso dos projetos são: falta de envolvimento do usuário, mudanças nos requisitos ou problemas na definição. Diante desses problemas, foi criada uma ferramenta. Durante a fase de prototipação, a ferramenta é utilizada na empresa Blueberry - Agência de Marketing Digital, permitindo a validação dos requisitos com clientes reais e melhorando a comunicação com eles. Os resultados obtidos foram satisfatórios com melhoria de até 40% na finalização dos projetos desenvolvidos.*

1. Introdução

A crise do software, termo utilizado desde o final dos anos 1960, expressa as dificuldades do desenvolvimento de software frente à complexidade dos problemas a serem resolvidos, crescente demanda de sistemas computacionais e o uso de técnicas e métodos adequados na sua implementação (SOMMERVILLE, 2008). Para superar essa

"crise do software", a Engenharia de Software (ES) tem proposto caminhos de solução que envolvem ferramentas, metodologias, padrões, artefatos, entre outros recursos para dar apoio ao processo de desenvolvimento, avaliação, implantação e manutenção de softwares. Apesar

dos esforços da ES, muitos projetos de software apresentam as situações citadas ainda na atualidade (PRESSMAN, 2011).

A confirmação dessa afirmação vem dos dados do estudo intitulado de Chaos Research (STANDISH, 2015), realizado desde 1994 pela Standish Group International, que relata o sucesso e o fracasso de projetos na área da Tecnologia da Informação (TI). No relatório Chaos Research os projetos são enquadrados em três categorias distintas: mal sucedido (o projeto é cancelado em algum momento do desenvolvimento por uma ou mais razões); bem-sucedido (o projeto é concluído dentro do prazo previsto e do orçamento estimado); e comprometido (o Projeto é concluído, porém entregue com atraso, com orçamento além do estimado, e em alguns casos, o software não possui todas as funcionalidades especificadas).

A partir do estudo será desenvolvida uma ferramenta para gerenciar o processo de comunicação entre cliente/usuário e equipe de desenvolvimento a partir do registro formalizado dos requisitos de um projeto Web, suas alterações e respectivas aprovações e reprovações do cliente/usuário. Com os registros visa-se a geração de um histórico que poderá ser utilizado para acompanhar a evolução, as alterações, as aprovações e reprovações, além de possibilitar informações para identificação dos impactos desses registros no sucesso, comprometimento ou insucesso do projeto.

O foco da proposição da ferramenta é o gerenciamento do processo de comunicação dos requisitos e escopo do projeto Web entre cliente/usuário e equipe de desenvolvimento. Nesse trabalho este processo é identificado como feedback, pois visa exprimir os registros de alimentação de retorno nas comunicações. Para validação da ferramenta será realizado um estudo de caso abrangendo um projeto de aplicativo Web e uma equipe de desenvolvimento.

Como resultado de aplicação da ferramenta, busca-se na perspectiva do cliente/usuário o atendimento de suas necessidades e expectativas contribuindo para a rápida identificação e resolução de questões e descontentamentos de forma a potenciar o seu envolvimento positivo no decurso da realização do projeto. Já na perspectiva do desenvolvedor, o propósito da ferramenta é dar suporte ao processo de comunicação com o cliente/usuário a partir do registro e acompanhamento da etapa de levantamento de requisitos e escopo com vistas ao sucesso do projeto.

2. Problemática

O avanço e o constante desenvolvimento da Internet e dos projetos que são criados especificamente para Web, tem modificado a forma de interação com as pessoas e os meios digitais. Nosso mundo mudou e com isso nossa forma de trabalhar também mudou. Há diversas formas de encantar e mudar a forma com que as pessoas interagem com o mundo e consigo (LOWDERMILK, 2013).

Conforme Norman (2013), mecanismos e formas para coletar feedback do usuário raramente existem. A busca incansável na tentativa de desenvolver algo melhor ao que foi desenvolvido previamente, na grande maioria das vezes acaba sendo desastrosa para o consumidor. Sendo assim há um insucesso para todos os envolvidos, tanto para a equipe que despendeu tempo e dinheiro para o desenvolvimento, quanto para o cliente que não teve suas necessidades atendidas.

Grande parte dos softwares e websites não atendem as necessidades dos clientes, o que acarreta em uma alta taxa de rejeição. Segundo Unger e Chandler (2009), estes problemas ocorrem devido a particularidade e singularidade de cada projeto. É imprescindível entender as funcionalidades e características, para entregar um produto de alta qualidade. Para atingir estes objetivos é necessário realizar o levantamento de requisitos junto ao cliente, porém, após o seu entendimento é necessário realizar uma validação prévia para identificar possíveis divergências.

O levantamento de requisitos é um padrão de tarefa que é importante para a prática bem-sucedida da engenharia de software, que prevê um processo completo envolvendo prototipação, planejamento e estimativa, como é descrito por (PRESSMAN, 2011).

Pretende-se desenvolver uma ferramenta utilizando as técnicas e metodologias de gerenciamento de projeto e engenharia de software visando validar requisitos entre cliente/usuário e equipe de desenvolvimento. Esta ferramenta irá auxiliar no processo de desenvolvimento do projeto Web e a demonstrar com mais clareza as necessidades do projeto, assim, evitando problemas com a falta de formalização e divergência com o cliente/usuário.

3. Ferramenta para validação de requisitos no modelo de prototipação em projetos web

Por meio do conhecimento adquirido durante o tempo do levantamento bibliográfico, foi possível criar um protótipo de aplicação de validação de requisitos baseado na estrutura Web, utilizando a tecnologia PHP para programar e fazer a interação entre cliente e servidor. A ferramenta faz parte do início do processo do XwebProcess, onde envolve a parte de protótipos e da avaliação conjunta com o cliente. É possível fazer na ferramenta o cadastro de clientes, que será identificado como o agente responsável por fazer as anotações na imagem como forma de realizar o feedback e o cadastro de projeto, onde será possível atrelar um projeto a determinado cliente. Tendo em vista que a validação de requisitos é muito importante, o intuito deste projeto é criar algo para facilitar a comunicação entre cliente e desenvolvedor, evitando assim o tempo perdido e os custos gerados pela má comunicação na prototipação.

3.1. Metodologia

Para começar o desenvolvimento do protótipo de ferramenta, foi utilizado o Sublime Text 2 que é um editor de código fonte muito leve e customizável. O arquivo pode ser baixado gratuitamente no endereço: https://sublimetext.com/2.

Para enviar os arquivos para o servidor através do protocolo FTP (File Transfer Protocol), foi utilizado a ferramenta de código aberto multiplataforma FileZilla. Esta ferramenta tem uma interface gráfica intuitiva que auxilia no envio de arquivos para o servidor com uma conexão segura

Foi utilizado o Annotorious que é um kit de ferramentas de imagem para anotação Open Source escrito em JavaScript. O plugin pode ser baixado gratuitamente no endereço: http://annotorious.github.io/.

O servidor responsável onde foi publicado a aplicação, utilizou o Apache que é um servidor HTTP livre e o mais utilizado na Web, juntamente com PHP, que é amplamente usado por aplicações Web, a versão do PHP utilizado foi a: 5.4.45.

Para o desenvolvimento do banco de dados, foi utilizado o MySQL na versão 5.5.52, pois além de ser gratuito, seguro e com uma boa performance, ele é o mais utilizado para Web.

3.2. Modelagem de dados

Todas as funções necessárias para a validação de requisitos estão presentes no projeto, começando pelo cadastro do cliente, cadastro de projeto, comentários na imagem. Esses dados são salvos no banco de dados no servidor web.

Para facilitar o entendimento do projeto e diminuir possíveis erros na hora do desenvolvimento foi realizado uma modelagem do banco de dados. Foi utilizado o programa MySQL Workbench, que é um software livre disponível nas plataformas Windows, Mac OS X e Linux. O MySQL Workbench é uma ferramenta visual que oferece modelagem de dados, desenvolvimento SQL e diversas outras opções.

Além disso, foi criado uma modelagem conceitual, pois ajudaria no entendimento do processo e ela é a própria descrição do sistema proposto.

3.3. Modelagem de dados

Depois do planejamento inicial estruturado e das ferramentas devidamente instaladas e configuradas, deu-se início ao desenvolvimento da ferramenta. Primeiramente foi feito uma estrutura em HTML simples sem estilos para se ter uma base do projeto e um melhor entendimento do mesmo.

3.3.1. Estrutura do servidor web

A criação do projeto foi realizada localmente com o Sublime e enviada para o servidor através do Filezilla. Na figura 1 é possível visualizar a forma em que o projeto foi organizado e dividido.

Name	Size	Last Modified	Type	Permissions
Annotorious	4 KB	30/08/2016 11:51	httpd/unix-directory	0755
Config	4 KB	03/09/2016 22:39	httpd/unix-directory	0755
Modulos	4 KB	16/09/2016 15:39	httpd/unix-directory	0755
projeto	4 KB	03/09/2016 21:34	httpd/unix-directory	0755
Recursos	4 KB	19/09/2016 15:05	httpd/unix-directory	0755
Tabelas	4 KB	16/09/2016 14:07	httpd/unix-directory	0755
Upload	12 KB	Ontem 11:48	httpd/unix-directory	0777
Views	4 KB	28/08/2016 22:27	httpd/unix-directory	0755
.htaccess	174 de bytes	03/09/2016 22:14	text/x-generic	0755
cadastro.html	534 de bytes	16/08/2016 15:04	text/html	0644
error_log	714 de bytes	08/09/2016 14:45	text/x-generic	0644
index.php	163 de bytes	29/09/2016 14:49	application/x-httpd-php	0644
login.php	1,01 KB	12/08/2016 11:58	application/x-httpd-php	0644

Figura 1. Estrutura do servidor web

Fonte: Do autor, 2016

Para realizar a conexão com o banco de dados, dentro da pasta Config foi criado dois arquivos PHPs. O arquivo config.php contém uma classe responsável por obter as informações e configurações do banco.

3.3.2. Aplicação web

Para dar início ao desenvolvimento da aplicação Web, foi utilizado o fluxograma criado conforme a figura 2.

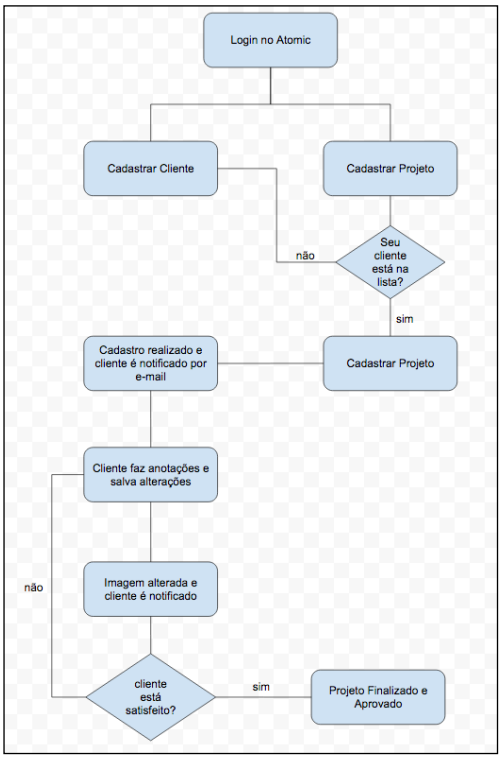


Figura 2. Fluxograma da aplicação

Fonte: Do autor, 2016

3.3.2.1. Tela de login

O arquivo inicial index.php que se encontra no diretório raiz, irá redirecionar para o endereço /Views/Usuario/login.php onde se encontra a tela inicial de login.

A tela de login possui os campos de usuário e senha, quando o botão “ENTRAR” é clicado, conforme a figura 9, uma requisição ajax é disparado em /Recursos/js/atomic.js), que chama o /Modulos/Usuario/login.php.

São enviados como parâmetros o e-mail e a senha como descrito na figura 20, para que seja realizado um select com esse e-mail e senha, caso retorne registro significa que estava correta a combinação de usuário e senha, então ele cria uma sessão para o usuário ficar logado e então retorna o registro para o ajax que foi disparado na tela de login, logo se retornou algo ele redireciona o usuário para a tela da dashboard.

3.3.2.2. Cadastro de projeto

No modal (janela secundária, também é conhecida como janela pop-up) de criação de novo projeto, é necessário preencher os campos, exceto os campos de selecionar, pois são previamente populados com as informações do banco de dados.

Quando o botão "CONTINUAR" é acionado, primeiramente um ajax é disparado conforme a figura 12, para o arquivo /Modulos/Projeto/cadastro_img.php que recebe a imagem por parâmetro e da upload nela para o servidor e retorna sucesso ou falha para o ajax.

Assim que o retorno é fornecido para o ajax, ele mesmo dispara outro ajax para o arquivo /Modulos/Projeto/cadastro.php contendo todos os dados preenchidos como parâmetros, ele recebe os parâmetros e armazena-os em variáveis e depois cria um objeto do tipo Projeto e seta todas as informações do projeto para as que foram enviadas como parâmetro e manda salvar o projeto no banco de dados com essas informações e retorna as informações necessárias para

que seja enviado um e-mail para o cliente com o link do projeto e preenche o modal de envio de e-mail com estas informações.

Ao clicar no botão “ENVIAR” um ajax é disparado para /Modulos/Projeto/enviar_email.php contendo os parâmetros de entrada necessários para o envio do e-mail que são o e-mail do cliente (destino), o link do projeto, e o tipo de evento (1 = inserção de novo projeto / 2 = o layout foi atualizado / 3 = o layout foi finalizado/aprovado).

3.3.2.3. Comentários na imagem

Os comentários são escritos e devem ser salvos após clicar o botão "Salvar", isso fará com que fique salvo apenas no localStorage do navegador, será salvo apenas no client-side. Para salvar diretamente no banco de dados, quando o botão "SALVAR ALTERAÇÕES" for acionado, onde um ajax será disparado para o arquivo /Modulos/Projeto/atomic_salva_bd.php e e este arquivo recebe o objeto com todos os comentários da imagem como parâmetro para que eles sejam salvos no banco de dados.

```
function salvar_comentarios(){

var aryComentarios = anno.getAnnotations(); //pega a instância(objeto) dos comentários criado em tempo de execução
var aryDados = [];

for(var i=0; i < aryComentarios.length; i++){
  var obj = aryComentarios[i];
    var img = obj.src; //imgpath nome da imagem (caminho)
    var txt = obj.text; //comentário escrito em texto
    var geo = obj.shapes[0].geometry; //dados de posicionamento e geometria
    var hei = geo.height; //altura do comentario
    var wid = geo.width; //largura do comentario
    var coX = geo.x; //coordenada X do comentario
    var coY = geo.y; //coordenada Y do comentario
  aryDados.push({'id_projeto': PROJETO_ID
                ,'texto': txt
                ,'geo_height': hei
                ,'geo_width': wid
                ,'coord_x': coX
                ,'coord_y': coY
               });
}
  $('#salvar_btn').attr('value', 'SALVANDO...');
  $.ajax({
    method: "POST",
    url: "atomic_salva_bd.php",
    data: {
           dados: aryDados //todos os comentarios
          ,id_projeto: PROJETO_ID
          ,id_img: IMAGEM_ID
        }
  }).done(function( msg ) {
    //sucesso
    $('#salvar_btn').attr('value', 'SALVAR ALTERAÇÕES');
    alert("Alterações salvas com sucesso.");
  });
}
```

Figura 3. Salvar comentários no banco de dados

Fonte: Do autor, 2016

No momento em que a página do projeto é carregada, ou seja, no evento onLoad do documento, um ajax é disparado para o arquivo /Modulos/Projeto/atomic_carrega_bd.php, onde é enviado por parâmetro o id do projeto, o id do cliente e o id da imagem que está sendo carregada. Essas informações são enviadas para posterior recebimento e uso dos parâmetros para selecionar os comentários que condizem com tais parâmetros no banco de dados e retornar para o ajax e adicionar os comentários retornados na instância global do objeto annotation para que o mesmo apareça na imagem.

3.3.2.4. Cadastro de cliente

No cadastro de cliente, após os dados serem preenchidos e o botão "Cadastrar" for clicado, primeiramente um ajax é disparado para a tela /atomic/Projetos/cadastro_img.php para dar upload na imagem. Caso o upload tenha tido sucesso, ele retorna o nome da imagem e com ele juntamente com os parâmetros do formulário dispara outro ajax para o arquivo /atomic/Modulos/Cliente/cadastro.php contendo todos estes parâmetros e cria uma nova

instância de Cliente e seta os parâmetros recebidos nessa nova instância e manda ela para ser salva no banco de dados.

3.3.2.5. Carregamento e endereçamento das imagens

Na tela de edição do projeto, a imagem da versão do projeto atual é carregada. O arquivo que se encontra em /Modulos/Projeto/atomic.php que contém esses parâmetros, carrega as informações do banco de dados e define o caminho da imagem através da pasta de upload que foi definido no arquivo config e também o nome da imagem que provém do campo imgpath da tabela imagens direto do banco de dados associada ao projeto e a versão em questão.

3.4. Funcionamento do sistema

O início do fluxo do protótipo inicia-se na tela de login. Nessa mesma tela é possível solicitar que a senha seja enviada novamente para o usuário em caso de esquecimento. O login é efetuado através do preenchimento dos dados de e-mail, senha e do botão "ENTRAR".

Após realizar o login com sucesso, é possível realizar três tipos de cadastro, o cadastro de usuário, cliente e projeto.

Os usuários serão responsáveis por cadastrar posteriormente clientes e novos projetos, esses usuários serão as pessoas que irão fazer a interação direta com o cliente, sem nenhuma interrupção de terceiros.

Ao clicar no botão "Novo Cliente" o usuário será levado a um modal onde é solicitado apenas três informações, nome do cliente, e-mail e uma foto para identificação.

Para cadastrar um novo projeto é importante que o cliente que será atrelado a este projeto tenha sido previamente cadastrado. Após clicar em "Novo Projeto", irá aparecer na tela um modal com o cadastro.

Após clicar no botão "CONTINUAR", o projeto é criado e automaticamente um e-mail é enviado para o cliente para notifica-lo com o link do projeto e com as instruções iniciais. O cliente irá clicar no link e será redirecionado para a tela de projeto, onde ele irá visualizar o projeto dele de forma mais realista, diretamente no seu navegador. Isso fará com que ele não apenas baixe uma imagem, mas fará com que ele tenha uma noção real do protótipo do seu projeto e um melhor entendimento e uma experiência mais realista.

O cliente poderá fazer inúmeros comentários, é necessário utilizar o mouse e selecionar a área desejada, um campo para acrescentar o comentário irá aparecer e após isto é possível clicar em “Salvar” ou “Cancelar”.

Após realizar os comentários, o cliente tem duas opções, clicar no botão "SALVAR ALTERAÇÕES" e o usuário responsável por esse projeto será notificado e posteriormente irá fazer outra versão com as alterações solicitadas pelo cliente, ou clicar em "FINALIZAR PROJETO", onde o cliente aceita que ao finalizar o projeto, ele concorda em aprovar o projeto estabelecido, não podendo mais comentar para posteriores alterações.

O desenvolvedor é notificado via e-mail quando o cliente realiza comentários na imagem ou quando finaliza o projeto, para continuar o fluxo e criar outra versão é necessário realizar o upload de uma nova imagem clicando em "Atualizar Imagem", onde irá surgir um modal com a opção de cadastrar uma nova imagem. Esse passo é feito até o cliente decidir finalizar projeto.

4. Resultados obtidos

A partir dos requisitos iniciais levantados neste trabalho, foi possível compilar um material de qualidade sobre os temas abordados na implementação do sistema proposto, sendo esse o principal objetivo. Foram destacados no projeto os principais motivos pelo insucesso e

comprometimento dos projetos realizados. Também foram estudados todos os métodos mais modernos e utilizados para o desenvolvimento da ferramenta protótipo.

Não foi encontrado trabalhos iguais ou similares utilizando o plugin JavaScript para anotação de imagem, deixando na fase inicial do trabalho um certo receio de como seria seu real funcionamento, porém durante o decorrer do trabalho ficou mais claro o seu funcionamento.

A homologação foi realizada na empresa de marketing online Blueberry e foi feita com dez clientes reais. Com a utilização dos clientes, ficou evidente a eficácia da ferramenta desenvolvida e no auxílio da comunicação entre o cliente e o desenvolvedor.

Um dos problemas recorrentes que acontecia antes da implementação da ferramenta, era a falta de entendimento com o protótipo do projeto. O cliente recebia uma imagem por e-mail, devido ao seu tamanho ser grande, quando o ele abria com seu programa padrão de imagens, ela se ajustava proporcionalmente, o que deixava os elementos menores e uma visão errônea do protótipo. Com a possibilidade de visualizar o protótipo em seu navegador, o índice de problemas reduziu completamente.

O tempo da etapa de validação de requisitos foi diminuída drasticamente, a média de finalização de projeto na ferramenta é de 15 dias, antigamente sem a ferramenta a média era de 25 dias. O processo ficou mais ágil e foi constatado que o retrabalho diminuiu consideravelmente, pois como antes os clientes não conseguiam visualizar de forma clara o projeto, os requisitos solicitados não podiam ser validados como é feito com a ferramenta.

Outro benefício com a ferramenta utilizada foi o número de projetos finalizados. Em linhas gerais os resultados obtidos foram satisfatórios com relação aos objetivos apresentados.

5. Considerações finais

Criar um projeto web requer uma grande variedade de ferramentas como observado durante o desenvolvimento do trabalho. As ferramentas utilizadas durante a execução do projeto e o planejamento detalhado, auxiliaram e ajudaram a evitar possíveis erros, evitando o retrabalho de algumas etapas.

Desse modo, o projeto torna-se importante, pois permite solucionar um problema vigente e contribui com o envolvimento do usuário com o projeto, e também, sua pesquisa contribuiu de grande fonte de conhecimento adquirido.

Durante a fase de testes ocorreu alguns problemas e dificuldades durante o desenvolvimento da aplicação, como o salvamento dos comentários na imagem, sendo assim necessário um esforço a mais e a correção da requisição ajax que estava com problemas. Outros problemas foram sendo encontrados durante o desenvolvimento da aplicação, mas foram se resolvendo a medida que os testes se aprofundavam.

A ferramenta protótipo foi homologada dentro da empresa Blueberry Marketing Online, onde foi utilizada em dez projetos com clientes reais. Mesmo após a fase de testes, enquanto a ferramenta foi utilizada nos clientes, foi possível fazer ajustes e aprimoramentos em busca de melhores resultados.

O feedback com os clientes que utilizaram a ferramenta foi satisfatório. Eles puderam visualizar o projeto de uma forma mais real e diretamente em seus próprios navegadores, sendo assim, os requisitos levantados previamente foram vistos de forma mais real e os próprios clientes visualizaram o que eles haviam solicitado.

Foi dado a possibilidade de fazer comentários no projeto, e a interação dos clientes com a ferramenta foi boa. As solicitações foram mais coesas, pois os clientes conseguiam visualizar exatamente o que eles solicitaram e tinham a possibilidade de comparar com a versão anterior.

Durante o processo de execução, alguns pontos surgiram como forma de aprimoramento, possibilitando a continuação do projeto para novas pesquisas nesta área. Alguns destes pontos são: inserir comentários na própria aplicação, sem que seja feito upload de uma imagem, implementar a opção de visualização da aplicação para dispositivos móveis, criar um canal na própria ferramenta de chat para dúvidas e para ter um controle maior dos requisitos solicitados.

Contudo, podemos concluir que o trabalho obteve o resultado esperado, atingindo todos os itens propostos com a conclusão do desenvolvimento da ferramenta protótipo para validação de requisitos em projetos Web.

Referências

LOWDERMILK, T. Design Centrado no Usuário. São Paulo: Novatec, 2013. 182 p.

NORMAN, Don. The Design of Everyday Things. Philadelphia: Basic Books, 2013. 347 p.

PRESSMAN, R. Engenharia de Software: Uma Abordagem Profissional. 7ª Ed., São Paulo: McGraw-Hill 2011.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011. 529 p.

STANDISH, Group International. Disponível em:<https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. Acesso em: 05 mar. 2016.

UNGER, Russ; CHANDLER, Carolyn. O Guia para Projetar UX. Rio de Janeiro: Alta Books, 2009. 268 p.