CENTRO UNIVERSITÁRIO UNIVATES

CURSO DE SISTEMAS DE INFORMAÇÃO

FERRAMENTA PARA PROTOTIPAÇÃO DE INTERFACES E APOIO AO MAPEAMENTO DE REQUISITOS DE SISTEMA

Moisés Heberle

Lajeado, Novembro de 2016.

FERRAMENTA PARA PROTOTIPAÇÃO DE INTERFACES E APOIO AO MAPEAMENTO DE REQUISITOS DE SISTEMA

Trabalho de Conclusão de Curso II apresentado ao curso de Sistemas de Informação do Centro Universitário Univates, para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Fabrício Pretto

Lajeado, Novembro de 2016.

AGRADECIMENTOS

Agradeço primeiramente à minha família, especialmente minha irmã, que sempre esteve presente e deu o apoio e incentivo para a conclusão deste curso sempre que foi necessário.

Aos amigos e colegas de trabalho, que deram conselhos durante toda a evolução deste trabalho, e me motivaram para que eu o concluísse com êxito.

Ao orientador Fabrício Pretto, que sempre me respondeu prontamente e deu o auxílio necessário em todas as etapas.

As demais pessoas que auxiliaram de alguma forma, direta ou indiretamente, respondendo questionários, realizando testes e contribuindo com opiniões para melhorias do trabalho.

RESUMO

Cada vez mais as empresas de software recebem novas demandas de requisitos de sistemas, sendo que há uma demanda cada vez maior, exigindo que o fornecedor atenda com qualidade, agilidade e fidelidade o que o cliente solicita. O setor de desenvolvimento frequentemente sofre com o problema de retrabalho na codificação, pelo fato dos requisitos não terem sido bem elaborados ou não representarem exatamente o que o cliente necessitava. Para reduzir esses problemas e apoiar o processo de concepção de um software, é apresentado neste trabalho a proposta de uma ferramenta visual de prototipação rápida para telas de sistemas na plataforma web. Essa ferramenta, ainda em caráter de protótipo, possibilita que analistas possam fazer esboços iniciais de telas na fase de elicitação de requisitos, permitindo o acesso por parte do cliente, que poderá visualizar, interagir e comentar pontos incorretos da concepção. Essa solução contribui para que a fase de requisitos transcorra de forma mais eficiente, contendo uma melhor documentação, e fazendo com que ambas as partes tenham uma maior participação no processo. Além disto, a ferramenta ainda permite que seja gerado código *frontend* inicial para os desenvolvedores do projeto, bem como pode ser utilizada para o propósito de testes de usabilidade e interface pelos testadores. A solução implementada foi validada por profissionais ligados diretamente a área de TI, que forneceram um *feedback* a respeito da utilidade e importância da ferramenta.

Palavras-chave: *Prototyping tool*, prototipação, requisitos, concepção.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

AJAX:	Asynchronous Javascript and XML
CMMI:	Cap’ability Maturity Model
CSS:	Cascading Style Sheets
HTML:	HyperText Markup Language
JSON:	JavaScript Object Notation
MVC:	Model View Controller
PC:	Personal Computer
PHP:	Hypertext Preprocessor
SQL:	Structured Query Language
TI:	Tecnologia da Informação
URL:	Uniform Resource Locator
XML:	Extensible Markup Language
YAML:	Yet Another Markup Language

SUMÁRIO

1 INTRODUÇÃO

No mercado atual, as equipes de desenvolvimento de software acabam, muitas vezes, ignorando algumas etapas importantes que devem ser seguidas durante a concepção de um sistema. Um dos maiores problemas existentes é o da equipe realizar o desenvolvimento, consumindo tempo e recursos e, ao final disso, o cliente perceber que o resultado não está de acordo com o que ele queria. Isto é um sinal de que não foi feito um esforço suficiente para a fase de concepção e requisitos.

Com a crescente competitividade e cobrança por entregas rápidas nos projetos de software, faz-se necessário cada vez mais o uso de ferramentas auxiliares que reduzam os custos com desenvolvimento, testes e elaboração de requisitos, a fim de reduzir custos com retrabalhos e falhas de entendimento por parte do cliente e dos projetistas.

De acordo com Koscianski (2007), quando a tarefa é executada por uma ferramenta, existem menos chances do resultado ser diferente, pelo fato de que diversas pessoas já a utilizaram. Além disso, existem várias tarefas que podem ser automatizadas, fazendo com que diminua-se a carga de trabalho das pessoas e garantindo maior uniformidade.

Com essas ferramentas automatizadas sendo implantadas junto aos processos de construção de software, reduzem-se as chances de erro provocadas pelo fator humano e tarefas repetitivas, e aumenta-se a qualidade final do produto.

Segundo Koscianski (2007), as ferramentas de quarta geração permitiram que o custo de desenvolvimento de interfaces fosse reduzido significativamente, e tornaram possível a rápida construção de protótipos, incluindo o usuário final no processo,

avaliando imediatamente o resultado obtido.

Pensando nisso, este trabalho propõe a implementação de uma ferramenta que ajude, tanto o cliente como os projetistas, a elucidar melhor as solicitações de novos requisitos, principalmente no que diz respeito a interfaces de usuário (IU). Essa ferramenta classifica-se como um software de prototipação interativa, baseado em *web* e tecnologias modernas.

De acordo com Sommerville (2012), é de vital importância conhecer e gerenciar corretamente a especificação e os requisitos de software, é preciso entender qual a demanda real dos usuários e fazer o gerenciamento de suas expectativas para que o sistema entregue seja útil e esteja dentro do orçamento e cronograma.

Ainda segundo o autor, quando construímos esses sistemas, pensamos em como podemos montá-los a partir de componentes e sistemas de software preexistentes (SOMMERVILLE, 2012). Nesse sentido, a engenharia de software só conseguiu evoluir graças ao reúso de componentes, bibliotecas e metodologias, fazendo com que seja evitado ao máximo o retrabalho.

Assim, caso uma empresa insista em utilizar as mesmas ferramentas e metodologias ao longo do tempo, sem aprender com erros de projetos passados e buscar a excelência constantemente, ela corre o risco de ficar para trás em relação a seus concorrentes.

1.1 Objetivos

O presente trabalho tem como objetivo principal construir uma ferramenta de apoio à prototipação de software, utilizando tecnologias modernas, que permita expressar os requisitos por meio de interfaces gráficas, aproximando, assim, o cliente da fase de concepção.

O analista poderá modelar de forma ágil o protótipo das telas, e o cliente terá um *feedback* rápido de como irá ficar o sistema, podendo com isto criticar, questionar e fazer com que tudo seja revalidado quantas vezes forem necessárias. Pensando dessa forma, o desenvolvedor só receberia o projeto a ser desenvolvido previamente validado pelo cliente, diminuindo a preocupação de estar fazendo algo incorreto ou ainda imaturo.

A ferramenta proposta pretende contribuir para a diminuição do retrabalho e erros na concepção de um novo projeto, permitindo fazer prototipações rápidas e apresentáveis aos usuários, que podem ter uma noção mais clara de como ficará o produto final. As telas geradas servirão também como documentação do projeto, podendo servir para

consulta posterior por qualquer um dos envolvidos no projeto em qualquer etapa pós-requisitos.

Segundo Sommerville (2012), “especificação de software ou engenharia de requisitos é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema. A engenharia de requisitos é um estágio particularmente crítico do processo de software, pois erros nessa fase inevitavelmente geram problemas no projeto e na implementação do sistema.”

1.2 Justificativa e relevância

A motivação da construção da ferramenta proposta foi de que, atualmente, existe uma certa carência de alguma que auxilie os diversos tipos de papéis na construção de um *software*, englobado em apenas um sistema. Por exemplo, a maioria das que se propõe servir para construção de protótipos, não possui um sistema integrado de *feedbacks* com o cliente, ou exportação de código que pode ser utilizado por desenvolvedores ou testadores.

É muito importante que a equipe possua uma ferramenta de prototipagem que se adéque as suas necessidades e auxilie satisfatoriamente nas fases de concepção do software. Desta forma, há uma tendência de redução considerável de custos posteriores com codificação e testes.

Segundo Sommerville (2011), “aproximadamente 60% dos custos de software são de desenvolvimento, 40% são custos de testes. Para o software customizado, os custos de evolução frequentemente superam os custos de desenvolvimento”.

1.3 Delimitação

O presente trabalho visa a implementação de uma ferramenta para auxiliar a fase de concepção e requisitos de software. A ferramenta proposta se concentra em auxiliar na análise de requisitos, projeto e teste. Também ajudará parcialmente no início da implementação, provendo um código inicial para que os desenvolvedores já possam começar com alguma base para a camada *frontend*.

Não constitui o foco deste trabalho tratar de questões como metodologias e uso de tecnologias avançadas no desenvolvimento e o aprimoramento de qualidade e performance em testes de software, tampouco ser utilizado como substituto para a programação customizada do software final, nem permitir a criação de interações e

validações muito complexas.

Serão geradas interfaces gráficas reais, visando ter uma reprodução altamente fiel, que serão executadas inteira e unicamente via navegador web, porém, não existindo a ocorrência de cálculos ou quaisquer outras funções de sistema, sendo que o limite de funcionalidades dinâmicas serão transições de uma tela para outra. Não existirá conexão com banco de dados, partindo da premissa de que os artefatos trabalhados serão protótipos iniciais, onde ainda não existirá a modelagem de dados.

Esse projeto também não tem como propósito substituir ou ser melhor do que sistemas atuais de prototipação de mercado, mas sim permitir a utilização de diferentes tipos de tecnologias e metodologias.

1.4 Organização do trabalho

O presente trabalho foi dividido nos seguintes capítulos:

- Introdução: descreve um *overview* sobre os objetivos principais, justificativa da proposta do trabalho e suas delimitações;
- Referencial teórico: descreve as pesquisas feitas com base teórica, citações relevantes de autores e materiais de estudo utilizados;
- Metodologia: apresenta o tipo de pesquisa que o trabalho utilizou, tecnologias utilizadas, e quais foram os resultados preliminares obtidos antes da ferramenta ser de fato implementada;
- Ferramentas relacionadas: esse capítulo é dedicado a realizar um comparativo das ferramentas existentes de mercado, que possuem propósitos semelhantes ao do presente trabalho, bem como uma visão geral sobre suas vantagens e desvantagens;
- Implementação da solução: apresenta os requisitos funcionais e não funcionais da ferramenta implementada, imagens e descritivos de seu funcionamento, e sua estrutura de diretórios;
- Validações: descreve, do início ao fim, de como foi o processo de validação da ferramenta com os usuários, bem como alguns gráficos resultantes de algumas questões e respostas obtidas por eles;
- Considerações finais: capítulo final, que apresenta um parecer geral deste trabalho, analisando se o objetivo foi atingido e quais possíveis melhorias futuras podem ser feitas.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados artefatos sobre o que a Engenharia de Software menciona em relação a temas que estão relacionadas intrinsecamente com a ferramenta proposta. Também cita alguns processos de requisitos, qualidade de software, testes e vantagens e desvantagens da prototipação, que é o tema principal abordado neste trabalho.

2.1 Processos de software

Para que as fábricas de software tornem-se competitivas no desenvolvimento de projetos, é necessário que sejam implantados processos bem definidos em todas as fases de execução. Isso faz com que os custos sejam reduzidos e a empresa tenha um padrão, podendo melhorar suas estimativas, prazos e produtividade.

De acordo com Koscianski (2007), a Engenharia de software se dedica, além dos processos técnicos de desenvolvimento de software, atividades de gerenciamento de projetos, desenvolvimento de ferramentas, métodos e teorias que dêem ênfase ao aspecto de produção de software.

No início da execução de um projeto, devem ser bem elicitados os requisitos do que o usuário quer. Para ajudar a clarear as ideias em relação ao que deve ser feito, tanto pelos analistas quanto pelo cliente, existem ferramentas de apoio que ajudam a entender qual será o propósito do software a ser desenvolvido.

De acordo com Engholm (2010), a elicitação de requisitos é a fase inicial de

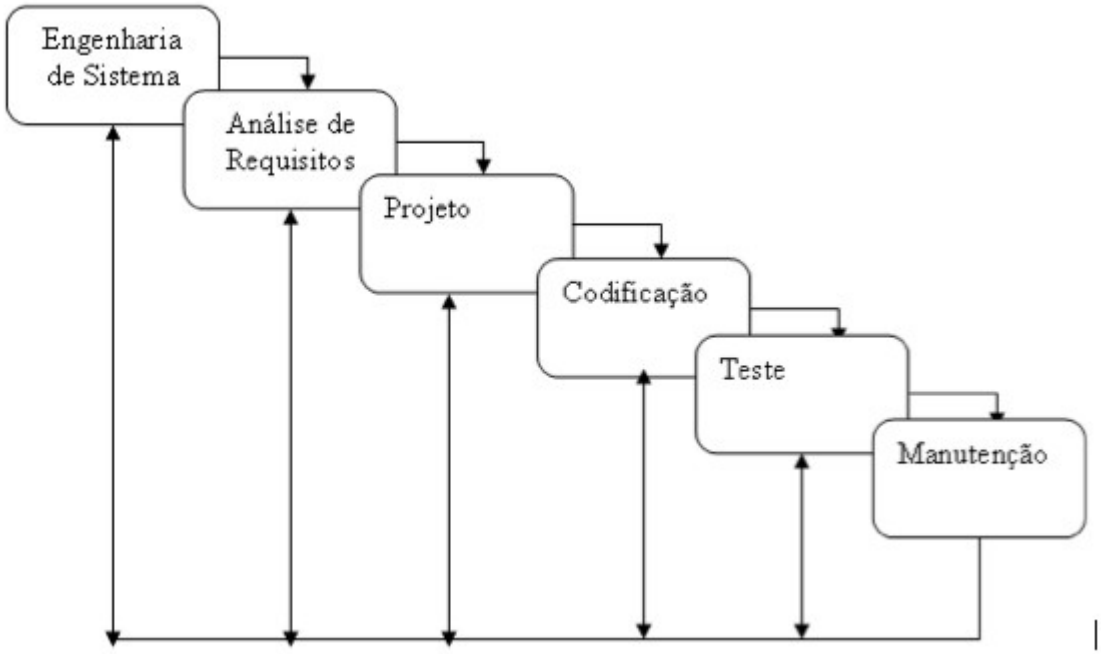
qualquer projeto, e ela é extremamente importante para o seu sucesso. Ela serve não apenas para o planejamento correto, análise de viabilidade e estimativa de custos, mas também para que seja entregue ao cliente um sistema que atenda suas expectativas.

Nesta fase, é recomendável o uso de protótipos visuais e interativos como apoio ao entendimento dos requisitos, o que ajuda a garantir um maior alinhamento entre a equipe e o cliente.

Segundo Koscianski (2007), as dificuldades em informática já começam durante as fases iniciais de um projeto, e delimitar o escopo e requisitos de um sistema não é uma tarefa trivial. A forma com que os requisitos podem ser alterados a todo momento, em consequência das mudanças de necessidades do usuário, é uma das maiores causas de insucesso de projetos de software e pode repercutir negativamente em vários aspectos estruturais do programa.

A Figura 1 representa a disciplina do desenvolvimento de um software tradicional**.**

Figura 1 – Ciclo de vida clássico de desenvolvimento de software



Fonte: PRESSMAN, 1993

2.2 Engenharia de requisitos

Pressman (2006) define requisitos de sistema como o conjunto de descrições que auxiliam os engenheiros de software a entenderem o que o sistema deverá fazer, e como

os usuários irão interagir com o sistema.

Os requisitos são o ponto de partida para toda a definição do sistema e, consequentemente, são fatores decisivos no desenvolvimento do produto final (MACHADO, 2014).

A gerência de requisitos é sempre uma atividade desafiadora no desenvolvimento de software. Isso ocorre porque os requisitos de um sistema são extremamente dinâmicos, com diversos fatores internos e externos ao projeto, contribuindo para sua alteração constante (MACHADO, 2014).

A qualidade dos requisitos feitos pelos analistas em uma equipe de desenvolvimento de software é fundamental para que, no momento de chegar ao desenvolvimento, os artefatos estejam bem descritos, consolidados e de fácil entendimento. Assim, será evitado retrabalho desnecessário por parte de desenvolvedores, evitando custos extras e extrapolação dos prazos.

Sommerville (2011), afirma que muitas vezes o requisito é apenas uma informação muito vaga sobre um determinado serviço, desta forma, Sommerville os divide entre requisitos de usuário, onde as informações são descritas em uma linguagem natural, sem muito detalhe e requisitos de sistema, onde as informações são descritas de forma mais detalhada sobre o comportamento do sistema.

Os requisitos expressam as características e restrições do produto de software do ponto de vista de satisfação das necessidades do usuário e, em geral, independem da tecnologia empregada na construção da solução, sendo a parte mais crítica e propensa a erros no desenvolvimento de software (MACHADO, 2014).

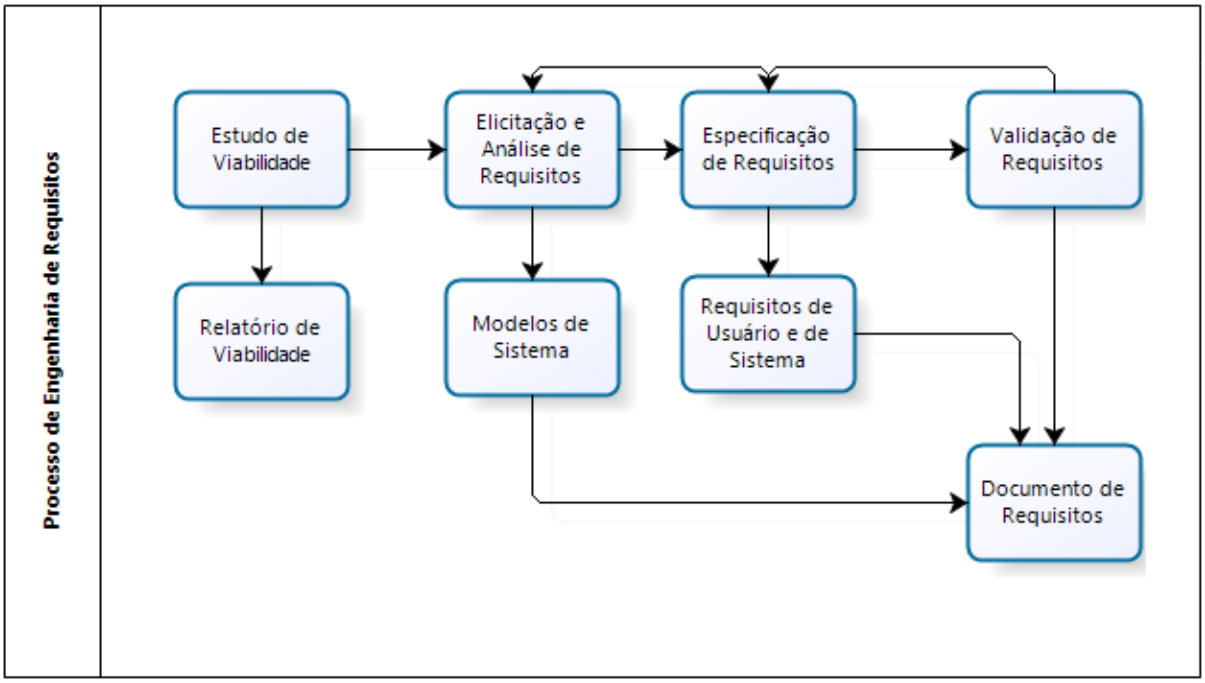
Os dois principais grupos de requisitos de software são classificados em: funcionais e não funcionais. Os requisitos funcionais são os que descrevem como o sistema deve se comportar, quais deverão ser as ações específicas para cada entrada, ou seja, descreve o que deve ser feito e o que deve sair do sistema. Já os requisitos não funcionais descrevem como deve ser feito, do ponto de vista de performance, confiabilidade, padrões de qualidade e robustez.

De acordo com Sommerville (2012), a mudança no desenvolvimento de software significa aumento de custos, pelo fato de que geralmente o trabalho deve ser refeito, ou seja, ocorre o retrabalho. Por exemplo, caso os relacionamentos entre requisitos do sistema forem alterados, toda análise de requisitos deve ser refeita, e muitas vezes, o sistema deve ser reprojetado de acordo com as novas alterações.

Muitas vezes o requisito é apenas uma informação muito vaga sobre um

determinado serviço (SOMMERVILLE, 2011). A mudança de requisitos ocorre em quase todos os sistemas existentes, devido ao fato de ocorrer melhor entendimento do objetivo do sistema pelo cliente, mudanças no hardware, software e na organização do sistema. Na Figura 2 demonstra-se o fluxo do processo de requisitos, desde o estudo de viabilidade até o documento de requisitos.

Figura 2 – Processos de requisitos



Fonte: Sommerville (2007, p.96)

2.3 Qualidade de software

A qualidade de software é um fator muito importante a ser levado em consideração no desenvolvimento de projetos. A falta de investimento e devida atenção nesse aspecto, pode levar empresas a perdas de clientes ou descrença no sistema por parte do usuário.

A garantia de qualidade deve ser imparcial para realmente encontrar e reportar problemas. Os processos de inspeção, revisão, verificação, validação e testes apoiam esse processo (KOSCIANSKI, 2007).

Segundo Engholm (2010), qualquer empresa que deseja obter ao menos em parte os benefícios fornecidos por processos, deveria institucionalizá-los na execução de suas tarefas relacionadas ao negócio. O objetivo é o desenvolvimento de software com qualidade e menores custos, para que os prazos do projeto sejam atingidos. A qualidade de um sistema é bastante influenciada pela qualidade do processo que é utilizado em seu

desenvolvimento e manutenção.

Para que uma empresa possa melhorar seus processos e nível de qualidade, existem certificações que permitem a comprovação de qualidade e maturidade na fabricação de software, como MPS.BR e CMMI.

O projeto MPS.BR, que foi iniciado no final de 2003, conta com a participação de sete instituições brasileiras, sendo a SOFTEX a principal. Esse projeto serve como um selo indicador do nível de maturidade da empresa no que diz respeito às boas práticas de desenvolvimento de software. O selo é classificado em níveis, sendo que cada um possui certas práticas associadas.

Os níveis de maturidade estão classificados em: A (Em Otimização); B (Gerenciado Quantitativamente); C (Definido); D (Largamente Definido); E (Parcialmente Definido); F (Gerenciado); G (Parcialmente Gerenciado). Cada um desses níveis possui suas áreas específicas de processo, em que são analisados os processos fundamentais, processos organizacionais e processos de apoio.

No MPS.BR, conforme a organização evolui nos níveis de maturidade, um nível maior de capacidade para desempenhar o processo deve ser atingido pela organização. Os níveis de capacidade estão divididos em cinco atributos de processos (AP), sendo eles: AP 1.1 - O processo é executado; AP 1.2 - O processo é gerenciado; AP 2.2 - Os produtos de trabalho do processo são gerenciados; AP 3.1 - O processo é definido; AP 3.2 - O processo está implementado.

2.4 Prototipação

A prototipação ajuda a entender qual o propósito do que foi desenvolvido, o negócio do cliente, a proposição de melhorias e a minimização de riscos e maximização de lucros.

De acordo com Engholm (2010), o protótipo da solução é um instrumento extremamente útil e importante para validação de requisitos. Esses protótipos servem para demonstrar o sistema, como será a navegação entre as interfaces, os relatórios previstos e assim por diante, reproduzindo o comportamento do futuro sistema a ser implementado.

Outra vantagem do protótipo, é que ele permite que sejam revelados erros cometidos nos requisitos propostos inicialmente, pois, com o passar do tempo, o usuário muitas vezes percebe que estava com uma visão incorreta ou incompleta do que realmente necessitava. Desta forma, a especificação do sistema pode ser modificada para

o correto entendimento dos requisitos.

Segundo Sommerville (2012), “um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções.”

Este processo tem como meta facilitar o entendimento dos requisitos, apresentando conceitos e funcionalidades do software. Desta forma, podemos elucidar melhor e sermos mais assertivos no que o cliente realmente quer antecipadamente, o que consequentemente aumentará sua percepção de valor.

De acordo com Pressman (2011):

> A engenharia de requisitos fornece o mecanismo apropriado para entender aquilo que o cliente deseja, analisando as necessidades, avaliando a viabilidade, negociando uma solução razoável, especificando a solução sem ambiguidades, validando a especificação e gerenciando as necessidades à medida que são transformadas em um sistema operacional [Tha97]. Ela abrange sete tarefas distintas: concepção, levantamento, elaboração, negociação, especificação, validação e gestão. É importante notar que algumas delas ocorrem em paralelo e todas são adaptadas às necessidades do projeto (PRESSMAN, 2011, p. 127).

Os protótipos podem ser grandes aliados das metodologias ágeis de desenvolvimento, garantindo um maior alinhamento do cliente com a equipe. Quanto mais fiel ele for, mais se assemelhará ao resultado entregue, no entanto, um protótipo de alta fidelidade levará mais tempo para ser criado. Para que seja escolhido o nível de fidelidade ideal é recomendável se levar em consideração a complexidade dos requisitos, prazo e orçamento do projeto.

Segundo Sommerville (2012):

> A prototipação de sistema constitui uma versão do sistema, ou de parte dele, que é desenvolvida rapidamente para a verificação das necessidades do cliente e a viabilidade de algumas decisões de projeto. Esse processo previne mudanças, já que permite aos usuários experimentarem o sistema antes da entrega e, então, refinarem seus requisitos. O número de propostas de mudanças de requisitos a ser feito após a entrega é, portanto, suscetível de redução (SOMMERVILLE, 2012, p. 29).

Em contrapartida, Pressman (2011) apresenta que:

> Frequentemente, o cliente define uma série de objetivos gerais para o software, mas não identifica, detalhadamente, os requisitos para funções e recursos. Em outros casos, o desenvolvedor encontra-se inseguro quanto à eficiência de um algoritmo, quanto à adaptabilidade de um sistema operacional ou quanto à forma em que deve ocorrer a interação homem/máquina. Em situações como essas, e em muitas outras, o paradigma de prototipação pode ser a melhor escolha de abordagem (PRESSMAN, 2011, p. 62).

Protótipos de software também podem ser utilizados com a finalidade de antecipar as mudanças que podem ser requisitadas, tanto no processo de engenharia de requisitos,

ajudando a elicitação e validação dos requisitos de sistema, quanto no processo de projeto, ajudando a apoiar o projeto de interface de usuário.

2.5 Desvantagens da prototipação

Pode-se considerar desvantajoso utilizar uma ferramenta complexa de prototipação em alguns casos e contextos. Um problema que pode ocorrer, é de uma empresa não ter pessoal com as habilidades especialistas necessárias na equipe de desenvolvimento.

Outro fator que deve ser levado em consideração, é que incluir a prática de prototipação, de forma rígida, no ciclo de desenvolvimento, pode deixar o processo de manutenção a longo prazo mais cara. Uma alternativa para isso, pode ser a utilização da prototipação descartável, ou seja, o protótipo é desenvolvido de uma especificação inicial, entregue para avaliação e então descartado.

2.6 Testes

O teste de software visa determinar se ele funcionou corretamente e atingiu suas especificações e objetivos para o qual foi projetado. A partir dele, é possível revelar e identificar falhas em um produto, para que estas possam ser corrigidas pela equipe de desenvolvimento antes que o mesmo seja entregue para o cliente, causando frustrações. Sendo assim, a fase de testes está diretamente ligada ao processo de software, e é um processo fundamental para garantir a qualidade do produto.

Como a maioria das atividades de engenharia, a construção de software depende principalmente da habilidade, da interpretação e da execução das pessoas que o constroem. Por esse motivo, erros acabam surgindo, mesmo com a utilização de métodos e ferramentas de engenharia de software (DELAMARO, 2007).

Como os softwares estão ficando cada vez mais complexos e presentes na vida das pessoas, devido ao surgimento de novas tecnologias, muitas têm alguma experiência com algum sistema que não funciona como deveria, e isto acaba diminuindo muito a confiança no mesmo. Para que isto não ocorra, é necessário investir em testes, pois estes reduzem os riscos de ocorrer defeitos em ambiente de produção.

Um desafio que as empresas encontram nos dias de hoje é produzir softwares de qualidade, com custos e prazos reduzidos, e atendendo as expectativas e requisitos impostos pelo cliente. Com a incorporação dos testes no processo, é agregada uma maior qualidade ao produto, antecipando falhas e incompatibilidades, e maior facilidade de serem realizadas métricas a partir dos defeitos encontrados, o que pode revelar se o

software será mais confiável, isto faz com que o custo de projeto fique reduzido.

O Quadro 1 descreve os tipos de testes de software.

Quadro 1 – Tipos de testes de software

Tipo	Descrição
Teste de Unidade	Teste em um nível de componente ou classe. É o teste cujo objetivo é um “pedaço do código”.
Teste de Integração	Garante que um ou mais componentes combinados (ou unidades) funcionam. Podemos dizer que um teste de integração é composto por diversos testes de unidade
Teste Operacional	Garante que a aplicação pode rodar muito tempo sem falhar.
Teste Positivo-negativo	Garante que a aplicação vai funcionar no “caminho feliz” de sua execução e vai funcionar no seu fluxo de exceção.
Teste de regressão	Toda vez que algo for mudado, deve ser testada toda a aplicação novamente.
Teste de caixa-preta	Testar todas as entradas e saídas desejadas. Não se está preocupado com o código, cada saída indesejada é vista como um erro.
Teste de caixa-branca	O objetivo é testar o código. Às vezes, existem partes do código que nunca foram testadas.
Teste Funcional	Testar as funcionalidades, requerimentos, regras de negócio presentes na documentação. Validar as funcionalidades descritas na documentação (pode acontecer de a documentação estar inválida)
Teste de Interface	Verifica se a navegabilidade e os objetivos da tela funcionam como especificados e se atendem da melhor forma ao usuário.
Teste de Performance	Verifica se o tempo de resposta é o desejado para o momento de utilização da aplicação.
Teste de carga	Verifica o funcionamento da aplicação com a utilização de uma quantidade grande de usuários simultâneos.
Teste de aceitação do usuário	Testa se a solução será bem vista pelo usuário. Ex: caso exista um botão pequeno demais para executar uma função, isso deve ser criticado em fase de testes. (aqui, cabem quesitos fora da interface, também).
Teste de Volume	Testar a quantidade de dados envolvidos (pode ser pouca, normal, grande, ou além de grande).
Testes de stress	Testar a aplicação sem situações inesperadas. Testar caminhos, às vezes, antes não previstos no desenvolvimento/documentação.
Testes de Configuração	Testar se a aplicação funciona corretamente em

	diferentes ambientes de hardware ou de software.
Testes de Instalação	Testar se a instalação da aplicação foi OK.
Testes de Segurança	Testar a segurança da aplicação das mais diversas formas. Utilizar os diversos papéis, perfis, permissões, para navegar no sistema.

Fonte: MOLINARI, 2014

De acordo com Sommerville (2012):

> Teste de usuário ou de cliente é um estágio no processo de teste em que os usuários ou clientes fornecem entradas e conselhos sobre o teste de sistema. Isso pode envolver o teste formal de um sistema que foi aprovado por um fornecedor externo ou processo informal em que os usuários experimentam um produto de software novo para ver se gostam e verificar se faz o que eles precisam. O teste de usuário é essencial, mesmo em sistemas abrangentes ou quando testes de release tenham sido realizados. (SOMMERVILLE, 2012, p. 159)

2.7 Usabilidade de interface

A usabilidade é, salvo exceções, o fator mais decisivo em relação à aprovação ou rejeição do usuário para com o software. Por esse motivo, é importante que em todo projeto seja feito um grande esforço para que as telas do sistema fiquem com a melhor ergonomia possível.

De acordo com Koscianski (2007), a usabilidade é uma das características mais impactantes de um software, e a interação entre programa e usuário tem uma influência determinante sobre a sua impressão de qualidade. Ainda que outros fatores como segurança e confiabilidade possam ser de importância particular em uma aplicação específica, problemas com o uso do software devem ser tratados com atenção pelos desenvolvedores. Caso um usuário fique frustrado ou irritado com a experiência de uso em um programa, irá desempenhar mal suas tarefas.

Sobre o apoio de protótipos de interfaces para construções de aplicações, Engholm (2010) afirma que para se obter bons resultados na usabilidade, é de extrema importância que sejam implementados protótipos das aplicações a serem desenvolvidas, pois é através deles que teremos uma visualização imediata do sistema pelos envolvidos no projeto para validação. Além disso, temos uma maneira de analisar os fluxos de trabalho e a usabilidade do sistema.

3 METODOLOGIA

Este capítulo tem como objetivo descrever a estrutura e pesquisa deste trabalho como um todo, bem como apresentar detalhadamente quais tecnologias foram utilizadas.

Para saber qual opinião que os profissionais da área tem em relação ao uso de protótipos de interfaces em projetos de software, foi realizado um questionário de múltipla escolha, *online*, fazendo diversos questionamentos sobre o tema.

3.1 Tipo de pesquisa

O desenvolvimento inicial do trabalho teve como base uma pesquisa experimental, onde o autor delimitou um problema a ser abordado e a partir disso realizou um estudo bibliográfico e de casos de uso.

A pesquisa exploratória estabelece critérios, métodos e técnicas para a elaboração de uma pesquisa e visa oferecer informações sobre o objeto desta e orientar a formulação de hipóteses (CERVO, 2006).

Para Gil (2007), a pesquisa experimental consiste em determinar um objeto de estudo, selecionar as variáveis que seriam capazes de influenciá-lo, definir as formas de controle e de observação dos efeitos que a variável produz no objeto.

Logo após o tema ter sido definido, começou-se o processo de delimitação dos objetivos e levantamento de requisitos. Essa fase foi importante para que o foco do assunto fosse mantido, evitando o aprofundamento em outras áreas não relacionadas.

Para atingir os objetivos esperados, foi realizado um estudo sobre as ferramentas existentes no mercado que possuem a mesma finalidade do tema abordado. A partir

disso, foi aplicado o questionário da seção 3.4 deste capítulo, depois realizada a implementação e validação de uso da ferramenta com profissionais de TI.

A implementação da ferramenta foi codificada e sendo refinada no decorrer do processo de estudo teórico e sugestões feita pelos profissionais que avaliaram a mesma, tanto no primeiro quanto no segundo questionário. Assim, puderam ser melhorados aspectos que ainda não estavam contemplados, imaturos ou insatisfatórios.

3.2 Tecnologias e ferramentas utilizadas

Nesta seção, é dada uma visão geral sobre as tecnologias predominantes, padrões de arquitetura de software e estruturas de dados que foram utilizadas no presente trabalho.

Tentou-se evitar o uso de soluções e bibliotecas genéricas que propõe-se a fazer uma gama de coisas diferentes, em prol de utilizar a ferramenta mais adequada para cada tipo de problema e necessidade, tentando seguir a filosofia da “ferramenta certa para o trabalho certo”.

Outros aspectos que pesaram para a escolha das ferramentas foram: existência de boa documentação disponível, desempenho, facilidade de uso e boa adaptabilidade. Todas as ferramentas e bibliotecas utilizadas possuem código livre e são abertas quanto a seu uso e contribuição na comunidade.

3.2.1 PHP

De acordo com S. Lopes (2007), PHP é uma linguagem que permite criar *websites* dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros de URL e links. A diferença de PHP em relação à linguagens semelhantes a Javascript é que o código é executado no lado servidor, sendo enviado para o cliente apenas HTML puro como resposta.

O software foi construído em PHP 5.3, utilizando conceitos recentemente implementados na linguagem, como *Closures* para passagem de *callbacks* e separação de classes utilizando *Namespaces*. O padrão de estrutura seguido é MVC.

3.2.2 YAML

Para que as alterações e criações de telas feitas pelo utilizador do software fiquem salvas, optou-se por utilizar como técnica de *storage* o formato YAML[1], pelo fato de ser

[1] YAML: YAML Ain’t Markup Language

uma linguagem de marcação simples e de fácil entendimento para um depurador de sistema. Este formato é muito utilizado por *frameworks* e bibliotecas modernas no PHP, como Symfony e Laravel. Pode ser vista como uma alternativa ao XML[2], e foi criado para ser uma linguagem legível aos seres humanos. Na Figura 3, demonstra-se como a estrutura é declarada.

Figura 3 – Exemplo da estrutura JSON da ferramenta

```
{
    "pagina": "Contato",
    "endereco": "/contato",
    "permissoes": [ ”administrador”, “usuario”]
    “componentes”:[
        {
            "tipo": "formulario",
            “configuracoes”:[ … ]
        },
        {
            "tipo": "listagem",
            “configuracoes”:[ … ]
        }
    ]
}
```

Fonte: Elaborado pelo autor

Estes metadados são salvos no próprio servidor, gerando um arquivo diferente para cada tela que é desenhada no sistema. A vantagem disto é que esta aplicação não dependerá de banco de dados, e pode ser facilmente extensível para os chamados *Cloud storages*, que correspondem ao armazenamento de dados na nuvem, como Dropbox, Amazon S3 e Rackspace, caso seja necessário, utilizando bibliotecas consolidadas do PHP para esta finalidade.

3.2.3 JQuery

Na parte administrativa do software, local onde o utilizador desenha os protótipos de interface e gerencia as telas, foi utilizada biblioteca jQuery para facilitar as chamadas

[2] XML: Extensible Markup Language

remotas AJAX[3] e codificação da parte JavaScript de forma facilitada, melhorando a produtividade e dinamicidade da aplicação.

De acordo com definição de Carvalho (2010), o jQuery é uma biblioteca JavaScript criada em 2006 por John Resig, que além de ser poderosa e de fácil utilização, pode também ser utilizada e modificada sem qualquer custo. Ela consegue abstrair do desenvolvedor web a maior parte da programação exigida para criação de interatividade com o usuário em um *website*, por exemplo.

3.2.4 JQuery UI

Ainda na parte administrativa do sistema, foram utilizados componentes do jQuery UI, que corresponde a um complemento da biblioteca Javascript jQuery. Com isso, é possível criar de forma facilitada funcionalidades como expandir e retrair, arrastar e soltar e caixas de diálogos sem a necessidade de programar em baixo nível.

De acordo com Diogo Souza (2013), o jQuery UI é uma ferramenta baseada no núcleo da jQuery, que permite a interação e animação dos diferentes elementos HTML, fazendo com que resulte em impressionantes interações *frontend*, com relativa facilidade de uso. É uma biblioteca rica e extensível de componentes gráficos, criada para expandir e maximizar o poder da jQuery, fornecendo uma melhor interação entre o usuário e o cliente, com recursos ricos como animação, efeitos e componentes estilizáveis.

3.2.5 Bootstrap

Para a parte *frontend* da ferramenta, foi escolhido o *framework* CSS Bootstrap 3, pelo fato de que o mesmo melhora o aspecto visual e de usabilidade, inclusive mobile, para os campos padrões HTML existentes nos navegadores web atuais. Além disso, o *framework* melhora a produtividade no que diz respeito a construir a parte de design, já que possui o design pré-fabricado bastando apenas ao desenvolvedor customizar as partes necessárias.

Segundo Utterback (2014), o Bootstrap pode ser definido como uma coleção de vários elementos e funções personalizáveis para projetos web, empacotados previamente em uma única ferramenta. Ao projetar-se um *website* utilizando Bootstrap, desenvolvedores podem escolher os elementos que querem utilizar, e, ainda mais importante, podem ter a certeza de que os elementos escolhidos não conflitarão entre sí.

---

[3] AJAX: Designa um conjunto de técnicas para programação e desenvolvimento web que utiliza tecnologias como Javascript e XML para carregar informações de forma assíncrona.

3.2.6 HTML5 Local Storage

Foi utilizado um recurso implementado no HTML5, intitulado Local Storage. Este recurso permite que seja feito a escrita e leitura de dados através da linguagem *JavaScript*, sendo que os dados escritos ficarão salvos apenas enquanto o navegador estiver aberto.

A decisão de se utilizar esta tecnologia foi de que os dados podem ser modificados dinamicamente, independentemente de ter um servidor remoto para que funcione. Isso permitiu que a ferramenta pudesse implementar o recurso de exportação da aplicação, que pode ser utilizada de forma isolada e *offline*.

3.3 Resultados e análise dos dados preliminares

Após o tema e os escopos da ferramenta e trabalho terem sido definidos, foi aplicado um questionário com profissionais da área de TI. Esse questionário teve como objetivo verificar quanto ao grau de relevância que uma ferramenta de prototipação pode ter a nível profissional.

O público alvo do questionário foi dividido em 2 grupos. O primeiro, representa profissionais que realizam funções diretas ao ramo de software, como análise e testes. O segundo grupo são de profissionais que trabalham na área, mas não têm relação direta com a concepção e desenvolvimento, como por exemplo, técnico em informática e consultor de TI. O questionário foi enviado para 12 profissionais diretos e indiretos da área.

O quadro Quadro 2 apresenta as perguntas feitas no questionário *online*, e enviadas para profissionais diretos e indiretos da área de software. Dos 12 profissionais enviados, 7 responderam, sendo que 5 são profissionais diretos da área, e 2 são indiretos.

Quadro 2 - Questionário sobre o tema: Protótipos de interfaces

PROFISSIONAIS DIRETOS	**PROFISSIONAIS INDIRETOS**
1. Você acredita que, no momento de negociação do software, utilizar uma ferramenta para a criação de protótipos visuais ajudaria na fase de modelagem de interfaces e levantamento de requisitos?	
75% - Ajudaria bastante 25% - Ajudaria um pouco	50% - Ajudaria bastante 50% - Ajudaria um pouco

2. Você já possui alguma experiência com o uso de ferramentas visuais para a criação de protótipos? Já utilizou ferramentas do tipo?	
100% - Já utilizei, porém atualmente não trabalho mais com ferramentas desse tipo	50% - Utilizo atualmente 50% - Nunca utilizei
3. Caso já tenha utilizado, as ferramentas de mercado atuais atenderam suas expectativas?	
25% - Atenderam plenamente 75% - Atenderam parcialmente	50% - Atenderam parcialmente 50% - Não tenho como opinar
4. Se a ferramenta de prototipação permitisse a demonstração para o cliente e *feedback* de forma rápida, direta e *online*, você acha que seria positivo no processo de análise e requisitos?	
75% - Acho positivo 25% - Acho que não faz diferença	50% - Acho positivo 50% - Acho que o cliente não deveria dar palpite nessa etapa
5. Você acredita que, com base nos protótipos gerados, seria possível fazer testes de usabilidade e validações básicas do software pelos testadores?	
75% - Sim, pois já poderiam ser encontrados problemas nas interfaces antecipadamente 25% - Não, pois só seria possível realizar testes muito simples	50% - Sim 50% - Não
6. Você acredita que a partir dos protótipos gerados pela ferramenta, já seria adequado gerar código-fonte inicial das telas para que os desenvolvedores tenham um ponto de partida?	
75% - Acho que seria adequado, pois melhoraria a produtividade no desenvolvimento 25% - Acho que não seria adequado, pois geradores automáticos costumam ser lentos e ineficientes	100% - Acho que seria adequado

Fonte: Elaborado pelo autor

Com base nas respostas do questionário sobre o tema prototipação, conclui-se que a maior parte dos profissionais vê com otimismo a utilização de protótipos na fase de elicitação de requisitos e negociação com o cliente, e as ferramentas atuais existentes, para muitos, não atendem plenamente essa tarefa.

Desta forma, entende-se que a criação de uma ferramenta que englobe de forma

integrada a criação de protótipos, demonstração *online*, sistema de *feedbacks* e geração de código inicial é vista como positiva para a maior parte dos respondentes.

4 FERRAMENTAS RELACIONADAS

Neste capítulo, são apresentadas três ferramentas de mercado estudadas pelo autor, e que englobam objetivos semelhantes ao da ferramenta proposta deste trabalho. Ao final, é apresentado um quadro comparativo estas ferramentas, seguido por uma conclusão geral do estudo.

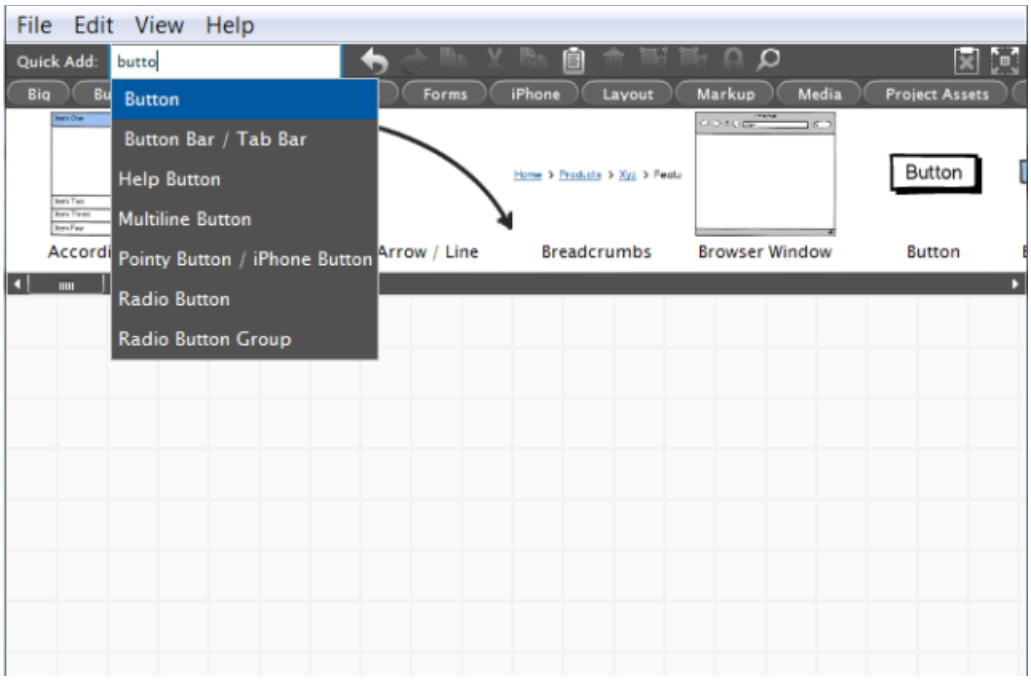
4.1 Ferramenta Balsamiq Mockup

Balsamiq Mockup (BALSAMIQ, 2011) da Balsamiq Studio é uma aplicação desenvolvida na linguagem de programação ActionScript, que executa adobe AIR (Adobe Integrated Runtime). Essa aplicação é utilizada para desenvolver protótipos ou modelos (mockups), como as telas de sistemas desktop, web ou mobile.

Para que seja possível a colaboração dos protótipos gerados com terceiros, essa ferramenta possui um serviço adicional, pago, chamado de *myBalsamiq*. Através dele, é possível criar uma conta e compartilhar os protótipos com uma equipe, onde todos os membros podem visualizá-los e melhorá-los.

A Figura 4 apresenta um *screenshot* da ferramenta Balsamiq Mockup. A interface do aplicativo é simples, possui uma barra de ferramentas chamada *Quick Add*, onde, através dela, faz-se possível encontrar e adicionar componentes.

Figura 4 – Screenshot da ferramenta Balsamiq Mockup



Fonte: DevMedia

4.2 Ferramenta Axure

O Axure é um programa líder de mercado para a criação de protótipos de telas para *websites,* sistemas web, software para computadores e aplicativos para celulares ou tablets.

Ele permite, ainda, a criação de protótipos simulados e com possibilidade de interação do usuário, que podem ser publicados na internet. Para criar esses protótipos *online*, a ferramenta gera automaticamente os códigos HTML, CSS e JavaScript. Porém, estes não são adequados para serem utilizados para o desenvolvimento final, pelo fato do código não estar otimizado para o uso em produção.

Essa ferramenta ainda possui um serviço extra, chamado de *Axure Share*, que é gratuito em quantidades limitadas de uso. Através desse serviço, é possível criar um ambiente de discussão dos protótipos com outros usuários. A Figura 5 apresenta um *screenshot* da ferramenta Axure.

Figura 5 – Screenshot da ferramenta Axure

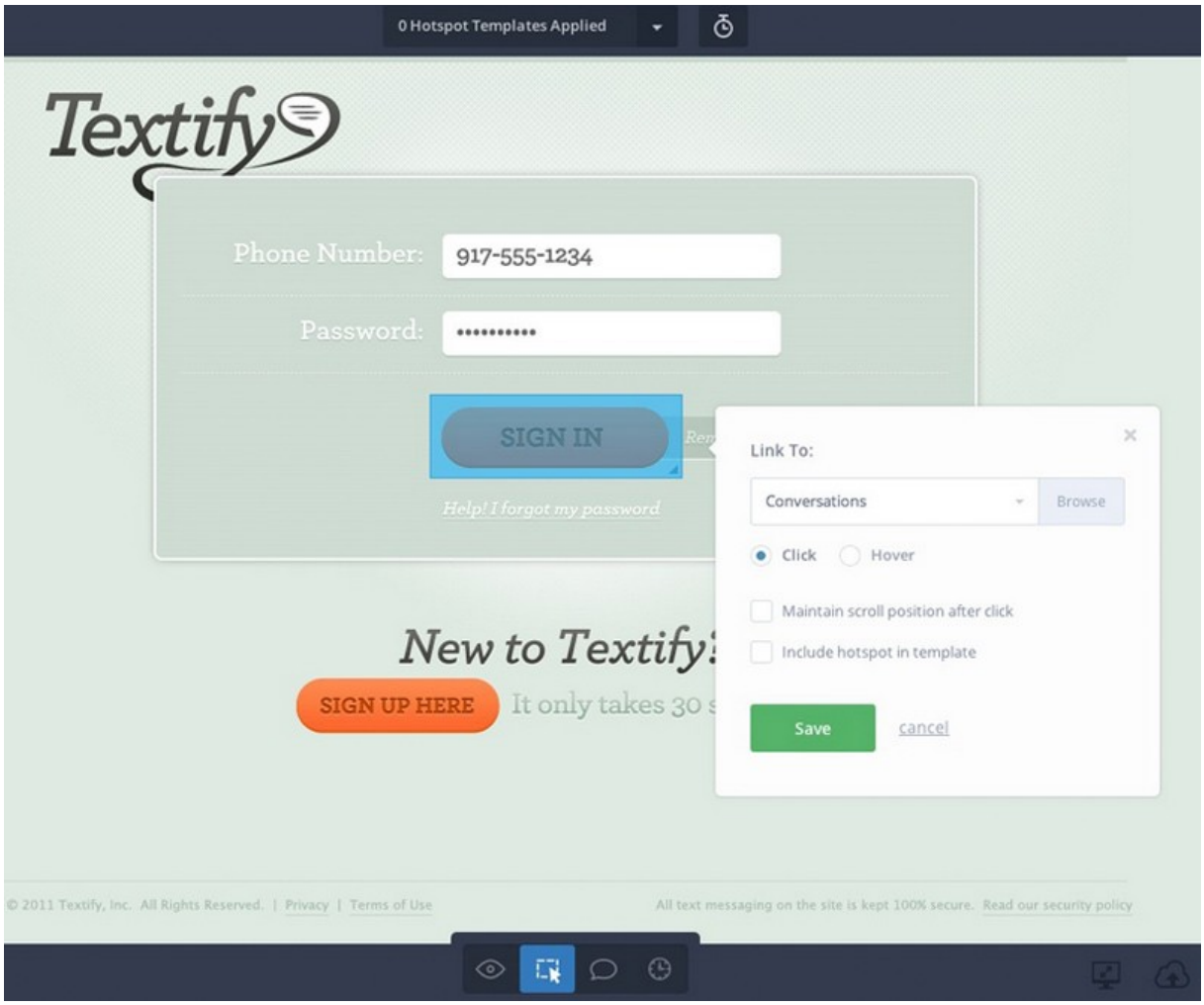
Fonte: Linowski

4.3 Ferramenta InVision

O InVision é uma ferramenta que oferece funcionalidades pertinentes à criação de protótipos, e possui alguns recursos que permitem melhorar a comunicação entre designers e clientes em relação a protótipos. Ele possui funcionalidades úteis como apresentações remotas para clientes.

Para facilitar o compartilhamento com clientes ou parceiros, o InVision possui uma ferramenta integrada chamada InVision Sync, que permite sincronizar facilmente os protótipos gerados no Dropbox ou Google Drive. Atualmente, entretanto, essa ferramenta está disponível apenas para Mac OSX. A Figura 6 ilustra a tela de criação de um protótipo exemplo de formulário de cadastro.

Figura 6 – Screenshot da ferramenta InVision



Fonte: Constructive

4.4 Análise comparativa entre ferramentas estudadas

Após ter sido realizado o estudo entre algumas ferramentas disponíveis no mercado com objetivos similares ao presente trabalho, conclui-se que existem boas soluções já existentes. No entanto, grande parte destas soluções não engloba o fluxo completo de prototipação, demonstração *online* e retorno de *feedbacks* por parte do cliente em uma só solução, ou seja, não possuem um foco muito específico.

O Quadro 3 apresenta um comparativo entre as ferramentas analisadas, levando em consideração as seguintes características, que são entendidas como relevantes na avaliação de sistemas:

a) Open source: Identifica se a ferramenta é de livre distribuição;

b) Multiplataforma: Identifica se a ferramenta pode ser instalada e utilizada em

diferentes sistemas operacionais;

c) Geração de código-fonte: Identifica se a ferramenta possui o recurso de geração de código-fonte para desenvolvedores e/ou testadores;

d) Compartilhamento de protótipos: Identifica se a ferramenta possui o recurso de compartilhar os protótipos gerados com terceiros, de forma integrada;

e) Nível de usabilidade: Apresenta uma classificação da ferramenta quanto à sua usabilidade;

f) Nível de fidelidade: Apresenta uma classificação da ferramenta quanto ao nível de fidelidade dos componentes em relação à interfaces reais, que podem ser utilizados no protótipo;

Quadro 3 - Comparativo entre ferramentas de mercado

Característica	**Balsamiq Mockup**	**Axure**	**InVision**
Open source	Não	Não	Não
Multiplataforma	Sim	Sim	Sim
Geração de código-fonte	Não	Sim	Não
Compartilhamento de protótipos	Não	Sim	Sim
Nível de usabilidade	Alto	Médio	Alto
Nível de fidelidade	Médio	Baixo	Alto

Fonte: Elaborado pelo autor

Ao analisar as ferramentas que são referência no nicho de prototipação, conclui-se que cada uma possui características peculiares e diferentes formas de uso. Todas as ferramentas estudadas possuem o objetivo em comum de serem utilizadas como uma forma de criar desenhos e protótipos rápidos, com a finalidade de facilitar a comunicação com terceiros em relação a ideias que ainda requerem amadurecimento.

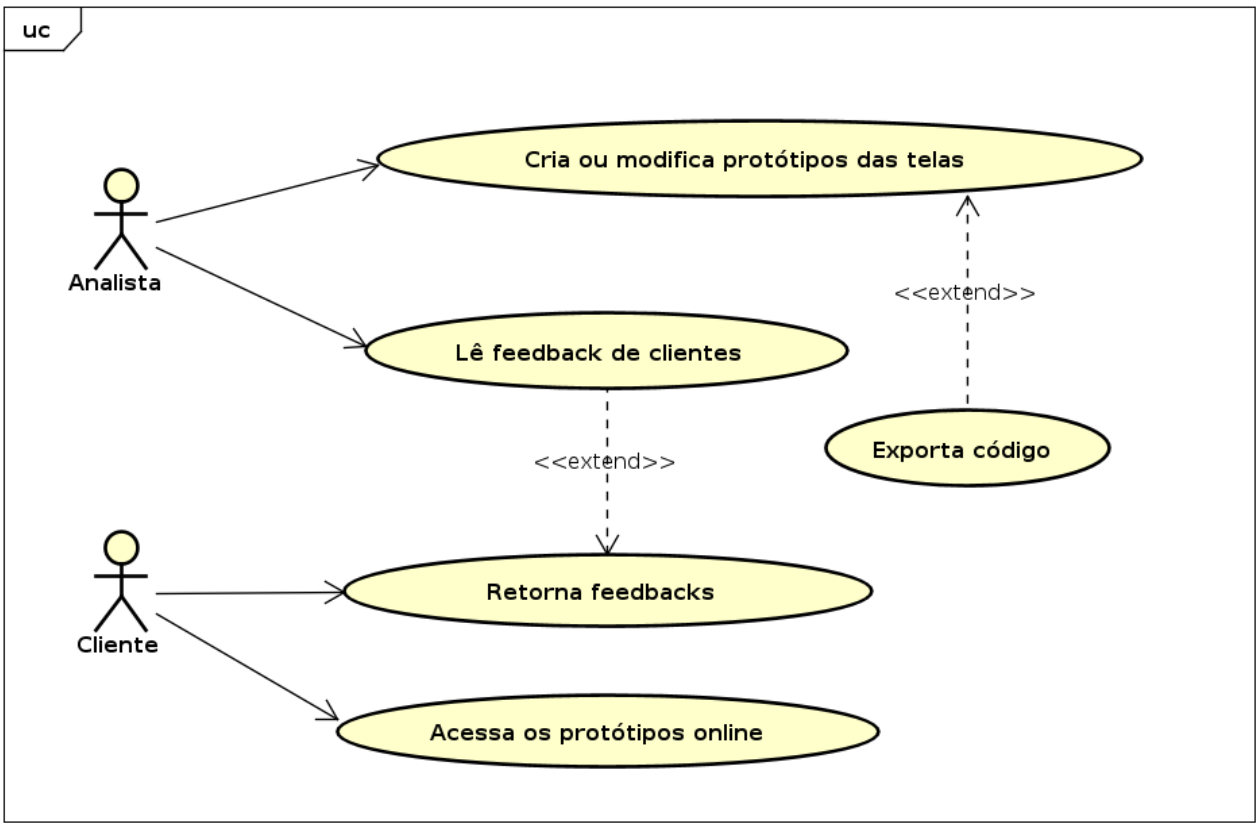
Ainda que as ferramentas apresentem uma boa usabilidade e atendam ao objetivo principal, percebe-se que de uma forma geral podem não atender a uma necessidade mais complexa de uma empresa de software, apenas se utilizadas em conjunto com outras. Isso porque são voltadas apenas e exclusivamente para a interface, não havendo, assim, um link com o projeto e requisitos do sistema.

5 IMPLEMENTAÇÃO DA SOLUÇÃO

Neste capítulo são apresentados os requisitos da ferramenta, telas do sistema real em funcionamento e os papéis de usuário, ou seja, como os atores principais irão interagir com o sistema.

A Figura 7 ilustra um diagrama de caso de uso que descreve os papéis dos principais atores do sistema.

Figura 7 – Caso de uso do sistema



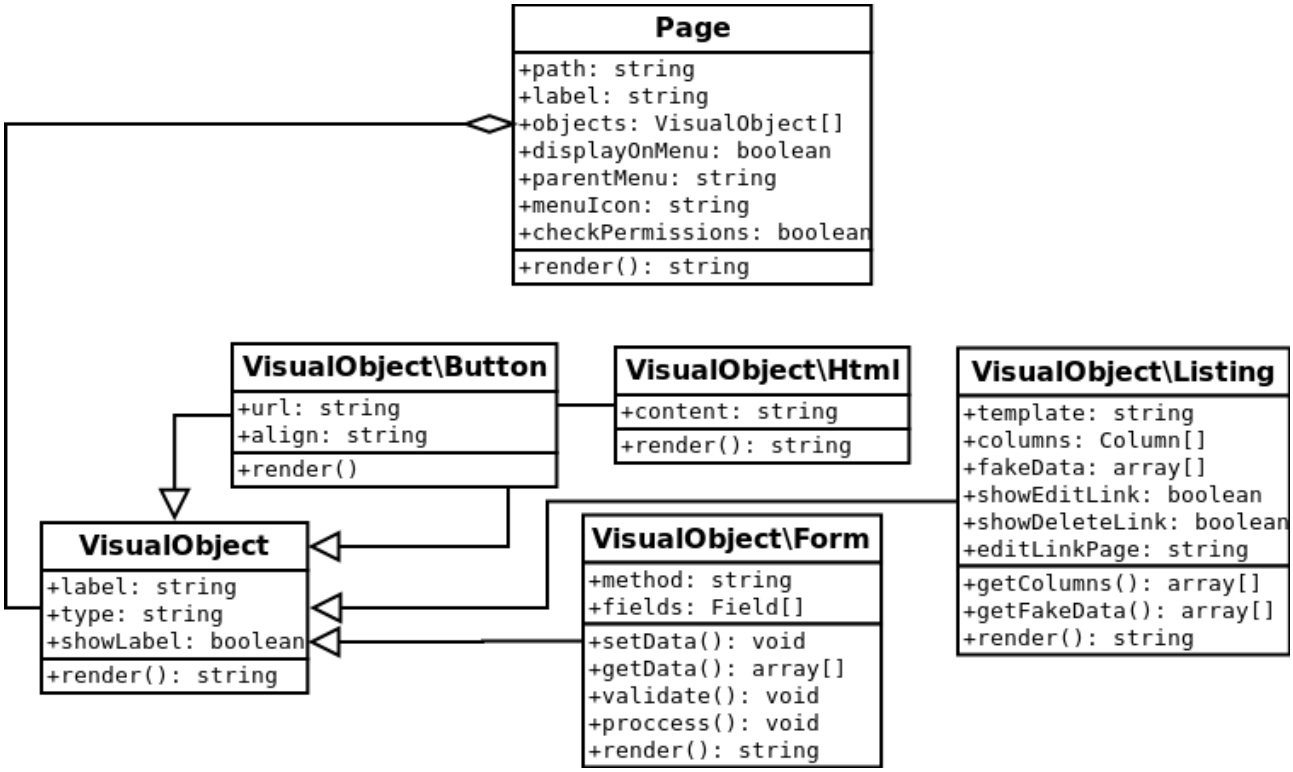
Fonte: Elaborado pelo autor

De acordo com a Figura 7, a especificação de casos de uso se faz da seguinte maneira:

- cria ou modifica protótipos das telas: esse requisito especifica que o analista tem permissão para gerenciar os protótipos das telas mediante interface administrativa;
- lê *feedback* de clientes: permite que o analista leia os *feedbacks* submetidos pelos clientes que acessam as interfaces geradas;
- exporta código: permite ao analista a exportação dos protótipos gerados;
- retorna *feedbacks*: permite que o cliente envie um *feedback* de alguma tela e que o analista responda-o, através da ferramenta;
- acessa os protótipos *online*: permite que o cliente possa visualizar os protótipos criados de forma *online*, através da ferramenta.

A Figura 8 apresenta o diagrama com as principais classes implementadas no software construído.

Figura 8 – Diagrama de classes



Fonte: Elaborado pelo autor

Como pode ser observado na Figura 8, cada tela criada pelo analista na ferramenta é representada pelo objeto *Page*, que contém um *array* de um ou mais componentes, representados pela classe *VisualObject*. Todas as classes citadas no diagrama contém um método *render*, que retorna a renderização HTML do que é exibido para o usuário final.

A classe *Page* possui alguns atributos principais, que são: *path,* representando qual o endereço de URL da tela; *label*, que representa o título da tela acessada; *displayOnMenu*, que indica se a tela deve ou não ser exibida no menu do sistema que o cliente acessa; e *checkPermissions*, que define se a tela deve verificar se existe algum usuário logado no sistema ou pode ser acessada publicamente.

Cada componente que compõe uma tela, a exemplo, um formulário ou listagem,

herda a classe pai *VisualObject*. Essa classe possui atributos e métodos em comum que são reaproveitados ou sobrescritos em casos específicos, como o atributo *label* e o método *render*. Isso facilita a identificação e manutenibilidade do código.

5.1 Requisitos funcionais

Esse item tem como objetivo descrever os requisitos funcionais para o desenvolvimento do trabalho proposto.

Os requisitos funcionais aqui expostos estão classificados em: obrigatório (O), importante (I) e desejável (D).

O Quadro 4 apresenta a lista de requisitos funcionais do sistema proposto.

Quadro 4 – Lista de requisitos funcionais

Código	**Descrição do requisito**	**Prioridade**
RF001	Gerenciar protótipos de telas	O
RF002	Gerenciar componentes específicos de cada tela	O
RF003	Visualizar uma prévia em tempo real das telas	D
RF004	Permitir a exportação de código	I
RF005	Permitir visualização *online* das telas	O
RF006	Permitir *feedback* individual de cada tela	O

Fonte: Elaborado pelo autor

Na sequência, será apresentada uma especificação e detalhamento de cada requisito funcional descrito na Quadro 4.

RF001 (Obrigatório) - Gerenciar protótipos de telas: através desse requisito, o analista poderá adicionar, modificar ou excluir qualquer protótipo desejado, desde que esteja autenticado e tenha devida permissão no sistema.

RF002 (Obrigatório) - Gerenciar componentes específicos de cada tela: através desse requisito, o analista poderá adicionar, mover, modificar ou excluir qualquer componente visual ou lógico pertencente a uma tela.

RF003 (Desejável) - Visualizar uma prévia em tempo real das telas: esse requisito permitirá com que o analista que estiver gerenciando algum protótipo de interface possa

visualizar as modificações que estão sendo feitas em tempo real, facilitando a visualização do resultado final dessa tela.

RF004 (Importante) - Permitir a exportação de código: através desse requisito, o analista poderá exportar uma ou mais telas criadas no sistema para código *frontend* HTML e CSS, que poderá ser utilizado independentemente da ferramenta de prototipação por desenvolvedores como um ponto de partida no desenvolvimento, e testadores para que realizem testes básicos de usabilidade.

RF005 (Obrigatório) - Permitir visualização *online* das telas: através desse requisito, o cliente ou usuário final terá a possibilidade de visualizar as telas através de um endereço público *online*, podendo interagir com as telas da mesma forma como se estivesse acessando o sistema real.

RF006 (Obrigatório) - Permitir *feedback* individual de cada tela: através desse requisito, o cliente ou usuário poderá, de forma simplificada, enviar uma avaliação ou comentário rápido sobre sua experiência em relação à tela que está sendo acessada, mediante identificação de seu nome e opcionalmente e-mail.

5.2 Requisitos não funcionais

Essa seção tem como objetivo apresentar os requisitos não funcionais do software desenvolvido, ou seja, todos aspectos que caracterizam o meio ambiente do sistema, definindo suas características e restrições.

O Quadro 5 apresenta os requisitos não funcionais, classificando-os em: obrigatório (O), importante (I) e desejável (D).

Quadro 5 – Lista de requisitos não funcionais

Código	Descrição do requisito	Prioridade
RNF001	Ser construído na linguagem de programação PHP	D
RNF002	Possuir disponibilidade de no mínimo 99% do tempo	D
RNF003	Poder ser acessado através de dispositivos móveis	D
RNF004	Possuir interfaces amigáveis e de boa usabilidade	D
RNF005	Ter um tempo de resposta máximo de até 1 segundo	D
RNF006	Gerar código *frontend* limpo e leve	D
RNF007	Ser compatível com o padrão HTML5	O
RNF008	Ter licensa definida como OpenSource	O

Fonte: Elaborado pelo autor

Na sequência, será apresentada uma especificação e detalhamento de cada requisito funcional descrito no Quadro 5.

RNF001 (Desejável) - Ser construído na linguagem de programação PHP: esse requisito define qual a linguagem a ser utilizada na escrita do programa.

RNF002 (Desejável) - Possuir disponibilidade de no mínimo 99% do tempo: esse requisito define que é desejável que a aplicação esteja disponível, de preferência, 24 horas por dia, 7 dias por semana, sem sofrer interrupções, já que usuários poderão acessar os protótipos gerados em qualquer momento do dia ou da noite.

RNF003 (Desejável) - Poder ser acessado através de dispositivos móveis: esse requisito define que o acesso das telas *online* deve funcionar bem tanto em PC's quanto dispositivos móveis, como celulares e *tablets*, visto que usuários ou clientes costumam utilizar dispositivos mistos e não há como exigir uma padronização neste sentido. Esse requisito não é obrigatório para a parte administrativa do sistema, no qual apenas analistas acessam para gerar as telas.

RNF004 (Desejável) - Possuir interfaces amigáveis e de boa usabilidade: esse requisito define que as interfaces do sistema como um todo tenham boa usabilidade e sejam fáceis de serem utilizadas.

RNF005 (Desejável) - Ter um tempo de resposta máximo de até 1 segundo: esse requisito define que o tempo de resposta tolerado de qualquer página do sistema, com exceção de processos em lote, seja de no máximo até um segundo, para que a navegabilidade não seja prejudicada.

RNF006 (Desejável) - Gerar código *frontend* limpo e leve: esse requisito define que o código *frontend* gerado, baseado em HTML e CSS deverá ser limpo e leve, evitando a inclusão de bibliotecas Javascript terceirizadas desnecessárias, provendo assim uma melhor usabilidade ao usuário final e mais facilidade na manutenção do software.

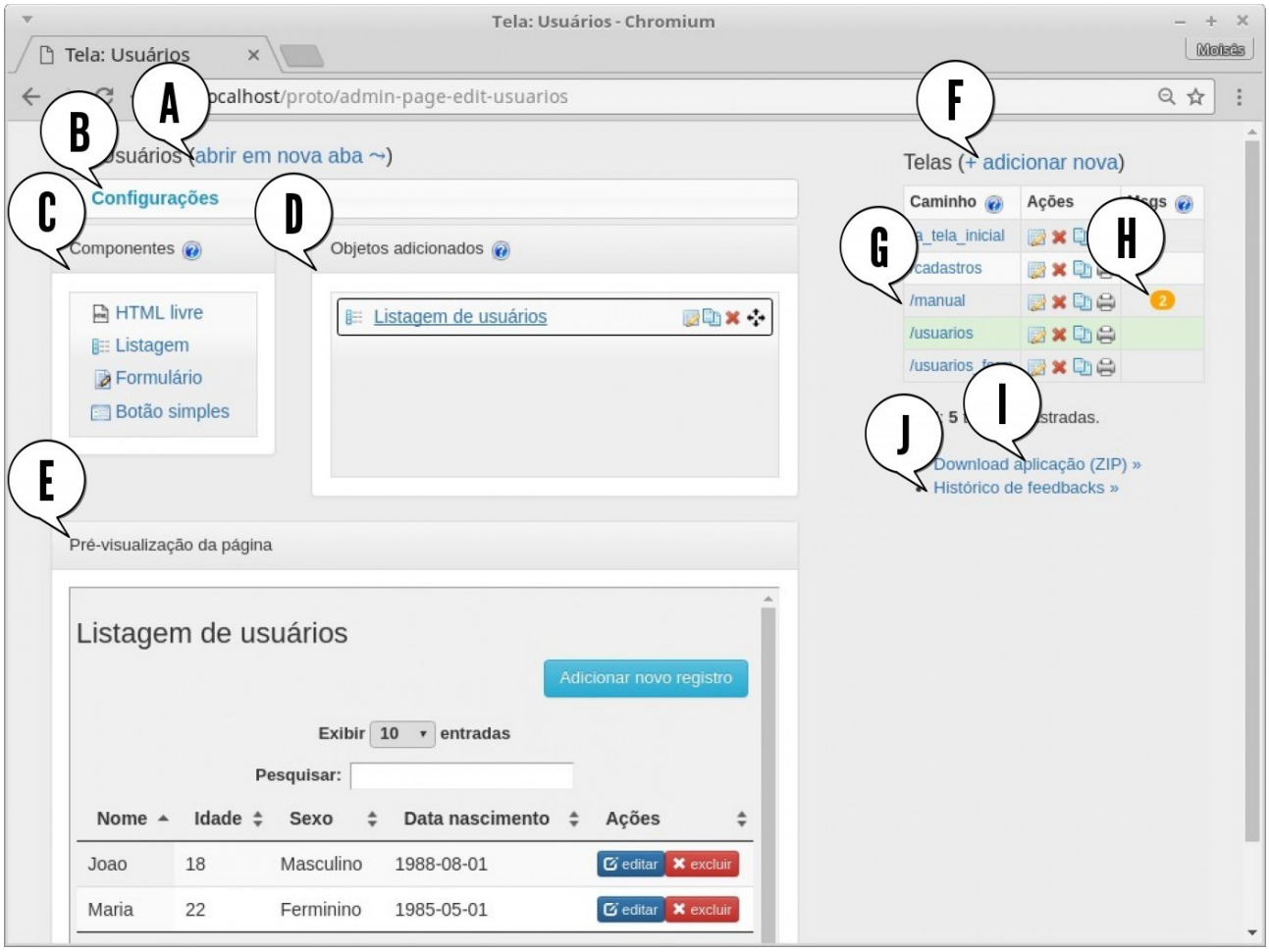
RNF007 (Obrigatório) - Ser compatível com padrão HTML5: esse requisito define que as telas que podem ser acessadas pelo usuário funcionem adequadamente em qualquer navegador que suporta HTML5. Esse requisito não é obrigatório para a parte administrativa do sistema.

RNF008 (Obrigatório) – A ferramenta proposta deverá ter sua licensa definida como OpenSource, ou seja, ter seu código de fonte disponibilizado publicamente, podendo ser visualizado e modificado por terceiros.

5.3 Criação de interfaces

O sistema possui uma área administrativa onde apenas analistas têm permissão para acessar. Nessa área, eles podem gerenciar todas telas de protótipos do futuro projeto. A Figura 9 demonstra como é a visão administrativa principal.

Figura 9 – Gerenciamento de interfaces



Fonte: Elaborado pelo autor

O Quadro 6 descreve detalhadamente a funcionalidade de cada item da tela administrativa principal do sistema, ilustrada na Figura 9.

Quadro 6 – Descrição da tela administrativa principal do sistema

O item A abre a tela atual que está sendo editada em uma nova aba do navegador, para que o analista possa visualizar o resultado da mesma forma que o cliente, através da demonstração *online*.
Ao clicar em cima do link expansível do item B, serão disponibilizadas algumas opções de configurações básicas da tela, como acesso ao menu, ícone, título e caminho (endereço) da página.
Os links contidos na caixa do item C, podem ser arrastados com o mouse para dentro da caixa “Objetos adicionados” (item D). Isso irá compor os elementos (componentes) que a tela irá ter.
Cada elemento adicionado na caixa “Objetos adicionados” (item D), possui as seguintes ações: Editar, Duplicar, Excluir e Mover (reposicionar acima ou abaixo). Na edição, o componente é editado separadamente, em uma popup, sendo que a tela que irá abrir possuirá propriedades diferentes de acordo com seu tipo, ex.: Formulário ou Listagem.
No lado esquerdo inferior, a caixa “Pré-visualização da página” (item E) tem a finalidade de permitir que o analista acompanhe em tempo real como a tela que será vista pelo cliente irá ficar (resultado final). Ela é atualizada automaticamente toda vez que alguma alteração é feita nos componentes da tela.
No lado direito superior da tela, existe um link denominado “Adicionar nova” (item F), que leva para um formulário de criação de nova tela.
Abaixo do link de Adicionar nova tela (item G), fica situada a listagem de todas as telas do sistema, sendo que a tela em edição no momento possui uma cor de fundo destacada. Cada tela possui as seguintes ações: Editar, Excluir, Duplicar e Imprimir (imprime a estrutura de metadados da tela). Por fim, existe ainda uma notificação que alerta quando existem *feedbacks* enviados pelos clientes pendentes para serem lidos (item H).
No item I, existe a ação intitulada “Download aplicação (ZIP)”, que tem o propósito de gerar o código final *frontend* de toda a aplicação, em formato HTML e CSS. Isso pode auxiliar no momento de início de desenvolvimento, permitindo que os desenvolvedores tenham um código-base inicial, ou até mesmo testadores, para efetuarem testes básicos de interface.
Na ação do item J, existe a ação intitulada “Histórico de feedbacks”, que redireciona para uma outra tela de listagem detalhada de todos os *feedbacks* de clientes.

Fonte: Elaborado pelo autor.

5.4 Componentes de telas

Cada tela pode conter um ou mais componentes, onde cada qual requere que sejam informados diferentes parâmetros pelo analista, o que resulta em diferentes resultados visuais. Na parte da programação do software, cada componente é representado por uma classe específica, e todas estendem de uma classe pai denominada Componente.

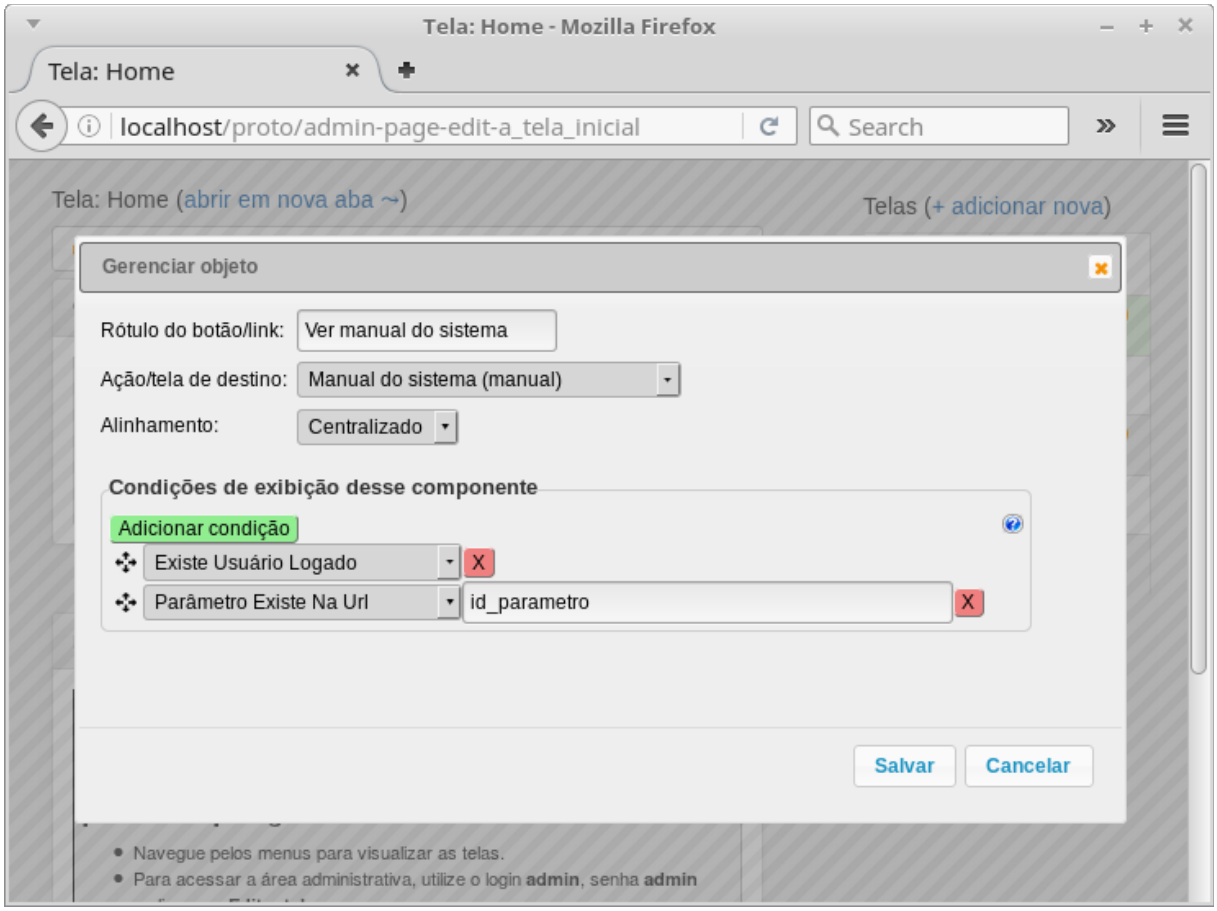
Ainda, na edição de cada componente é possível configurar, opcionalmente, pré-condições de exibição, onde, caso certas condições não sejam satisfeitas no momento de execução, o componente não seja exibido na tela. Assim, aumentam-se as possibilidades de simulação de regras de negócio reais. Alguns exemplos que podem ser citados como condições, são:

- Se: Existir usuário logado, então: Exibe um link de inserção de novo usuário.

- Se: Existir um parâmetro na URL denominado *id_registro*, então: Exibe formulário de edição do registro.

A Figura 10 demonstra a tela de gerenciamento das condições, que aparece atrelado na edição de cada componente.

Figura 10 – Interface de gerenciamento das condições de exibição



Fonte: Elaborado pelo autor

O número de condições de exibições para um componente pode ir de zero até ilimitado, sendo que, caso seja configurado mais de uma condição, a regra retornará verdadeira quando todas as condições forem satisfeitas, utilizando o operador lógico AND. O Quadro 7 descreve todas as opções disponíveis nas condições de exibição de um componente, ilustrado na Figura 10.

Quadro 7 – Opções de condição de exibição de um componente

Existe usuário logado	Somente exibirá o componente na tela quando existir um usuário logado no sistema, seja ele Administrador ou Cliente.
Usuário logado administrador	Somente exibirá o componente na tela quando existir um usuário logado no sistema, e esse usuário tiver permissão de Administrador.
Parâmetro existe na URL	Somente exibirá o componente na tela quando for passado um parâmetro específico na URL, sem considerar seu valor.

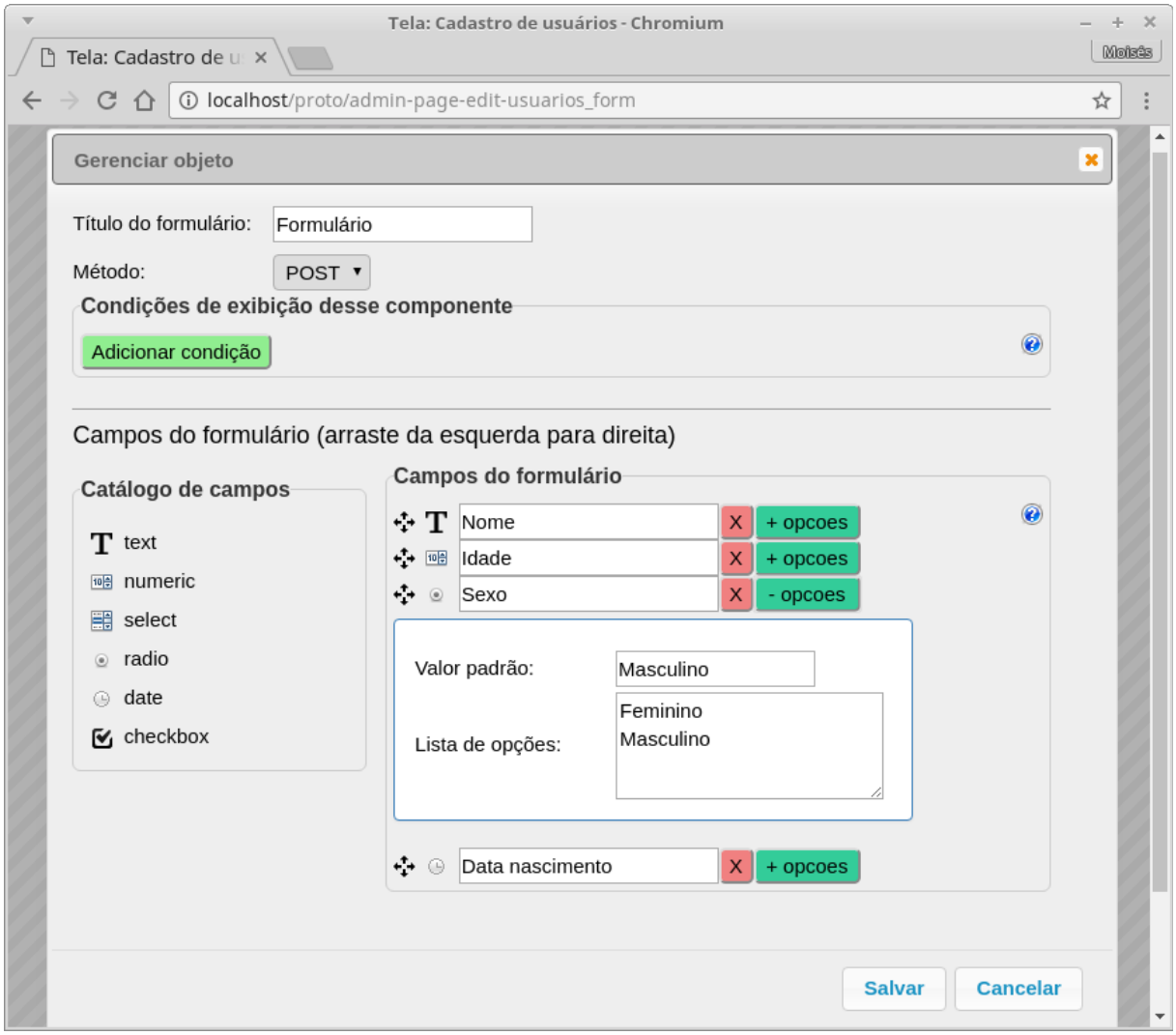
Parâmetro possui valor	Somente exibirá o componente na tela quando for passado um parâmetro específico na URL, e esse parâmetro tiver também um valor específico.

Fonte: Elaborado pelo autor

5.4.1 Formulário

Esse componente tem a finalidade de adicionar um elemento HTML de formulário para a tela de protótipo. A Figura 11 apresenta a tela de edição ou inserção de um componente de formulário.

Figura 11 – Interface de inserção e edição de formulário



Fonte: Elaborado pelo autor

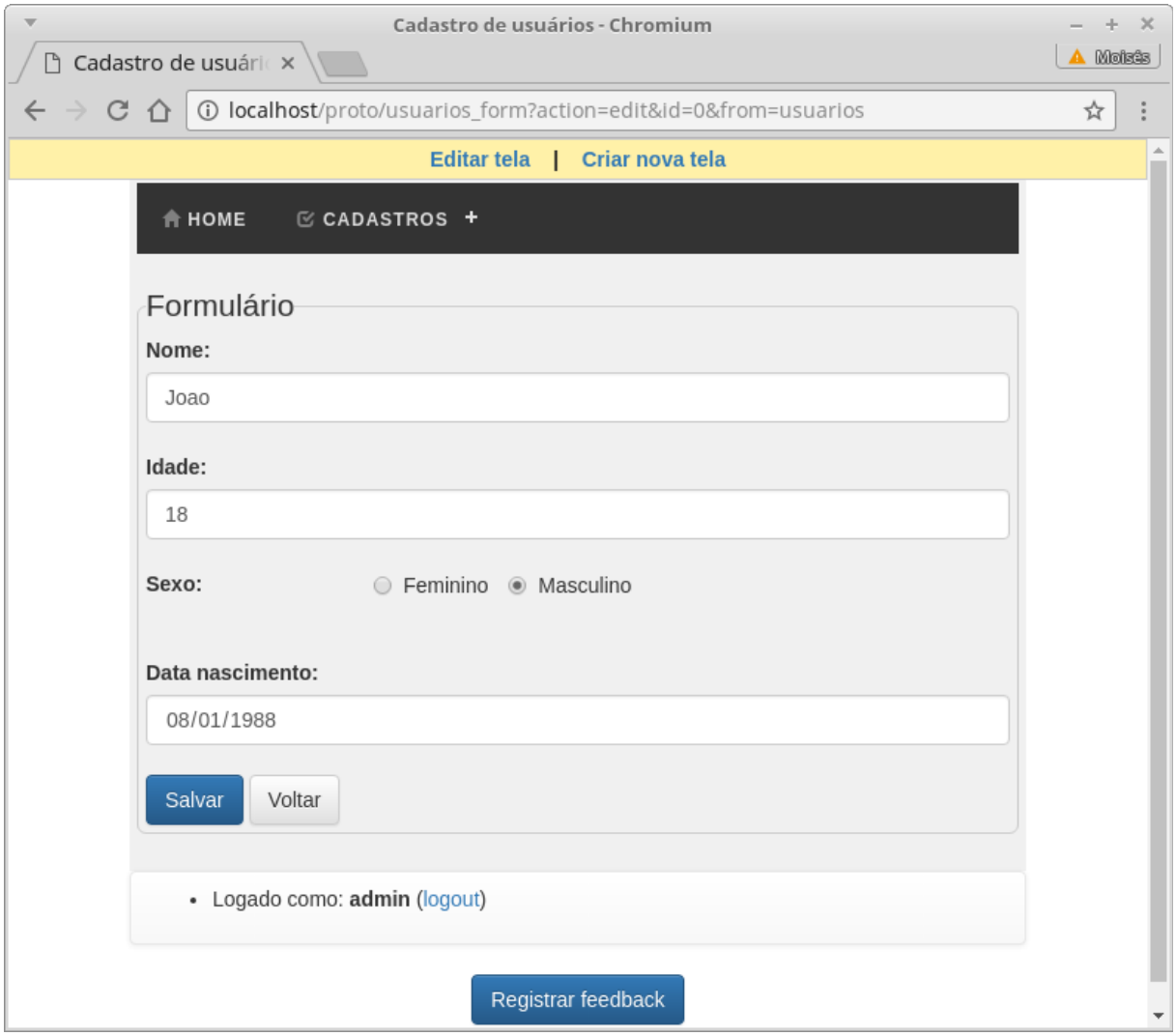
De acordo com o que é apresentado na Figura 11, no campo Título pode ser informado, opcionalmente, um nome amigável para o formulário, e no Método diz-se qual

a forma de submissão que deve ser enviada para o servidor, podendo alternar entre GET e POST.

É possível arrastar e soltar campos do tipo desejado para adicioná-los ao formulário, da esquerda para direita, podendo posteriormente editar seus atributos, como o rótulo ou valor padrão, bem como mover a posição de cada um em relação aos outros.

A Figura 12 apresenta um formulário de edição de registro, que foi construído na área administrativa da ferramenta através do componente de formulário.

Figura 12 – Exemplo de formulário gerado



Fonte: Elaborado pelo autor

O Quadro 8 apresenta um detalhamento de cada tipo de campo disponível no catálogo, ilustrado na Figura 11.

Quadro 8 – Tipos de campos do componente de formulário

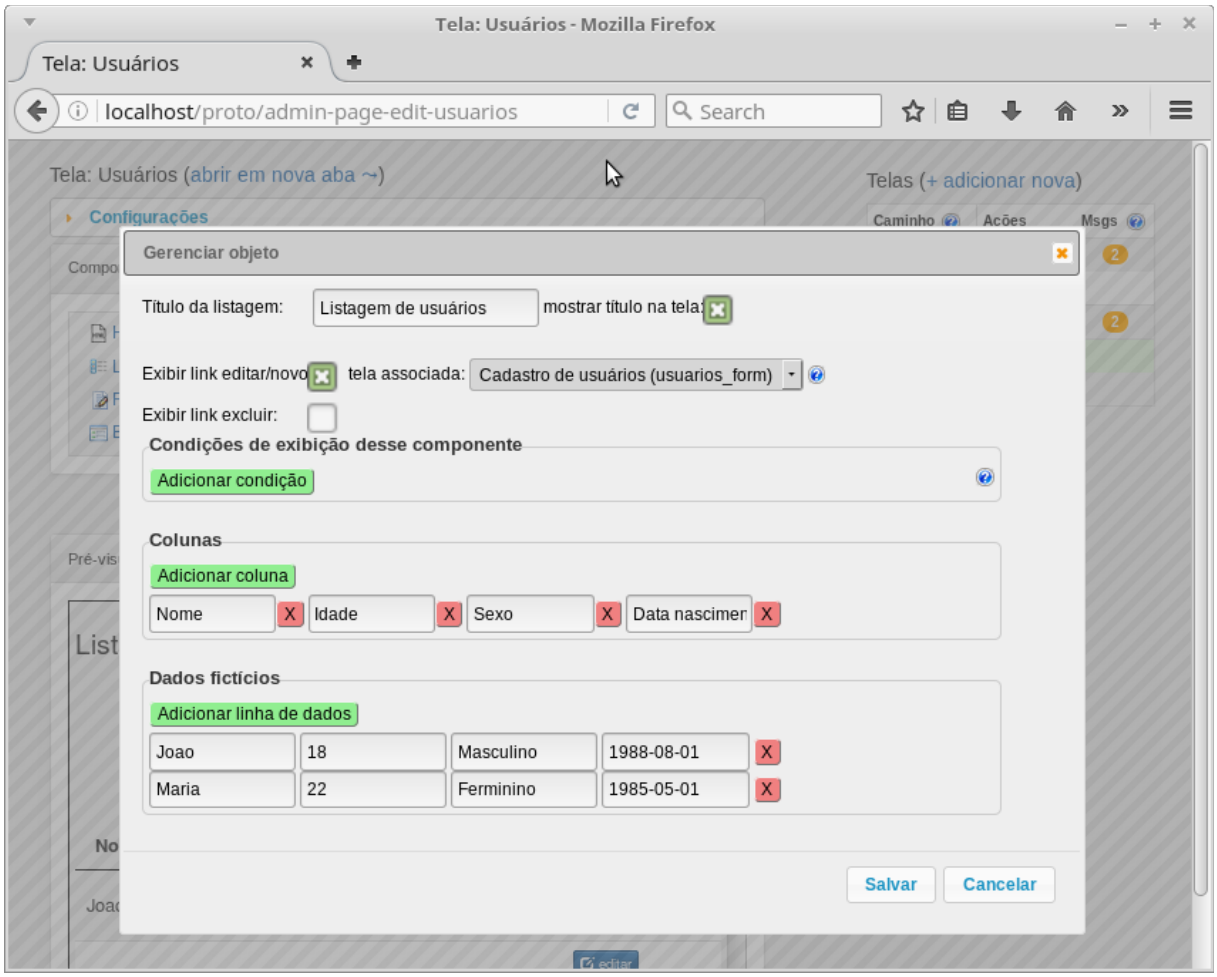
text	Campo tipo texto simples
numeric	Campo tipo número
select	Campo tipo lista de seleção. Nas opções extras, permite que seja configurado a lista de opções disponíveis, separadas por quebra de linha
radio	Campo tipo lista de seleção. Nas opções extras, permite que seja configurado a lista de opções disponíveis, separadas por quebra de linha
date	Campo tipo data
checkbox	Campo tipo lógico. Quando estiver marcado, representa Verdadeiro, e desmarcado, Falso

Fonte: Elaborado pelo autor

5.4.2 Listagem de dados

Da mesma forma que o formulário, este componente serve para criar listagens de dados. Como está sendo lidado com protótipos e não tem-se uma base de dados real, o analista pode inserir dados fictícios fixos para serem exibidos. A Figura 13 demonstra a interface de gerenciamento de listagem de dados.

Figura 13 – Interface de inserção e edição de listagem de dados



Fonte: Elaborado pelo autor

O Quadro 9 apresenta um detalhamento da tela de gerenciamento do componente de listagem de dados, ilustrado na Figura 13, em ordem sequencial.

Quadro 9 – Detalhamento da interface de gerenciamento de listagem de dados

Título da listagem	Rótulo da listagem de dados.
Mostrar título na tela	Possui, na sequência, uma opção que, caso selecionada, fará com que este rótulo seja exibido acima da listagem na tela gerada.
Exibir link editar/novo	Caso selecionado, esta opção habilita um link de edição para cada registro da listagem, e também exibe no topo um botão intitulado “Adicionar novo registro”.
Tela associada	Caso “Exibir link editar/novo” for habilitado, essa opção será liberada para que seja informado qual tela de destino os links devem redirecionar. Essa tela destino geralmente contém um formulário com as colunas idênticas ao da listagem em questão, assim, a ferramenta consegue identificar esses campos para popular e gravar corretamente os dados fictícios.
Exibir link excluir	Caso selecionado, esta opção habilita um link de exclusão para cada registro da listagem.
Colunas	Esta caixa representa a configuração das colunas que serão exibidas na listagem. É obrigatório preencher pelo menos uma coluna.
Dados fictícios	Esta caixa representa a configuração opcional dos dados fictícios que devem vir carregados na listagem. Sempre que uma coluna da caixa “Colunas” é adicionada ou excluída, automaticamente os campos desta caixa são atualizados, provendo uma boa usabilidade para o analista. Como trata-se de um protótipo, os dados fictícios tem a finalidade do cliente ter uma noção mais real dos dados que podem ser cadastrados. Vale ressaltar que esses dados, caso sejam configurados pelo analista, poderão ser excluídos e editados pelo cliente que irá visualizar a tela final de listagem, e serão zerados novamente apenas quando seu navegador for reiniciado.

Fonte: Elaborado pelo autor

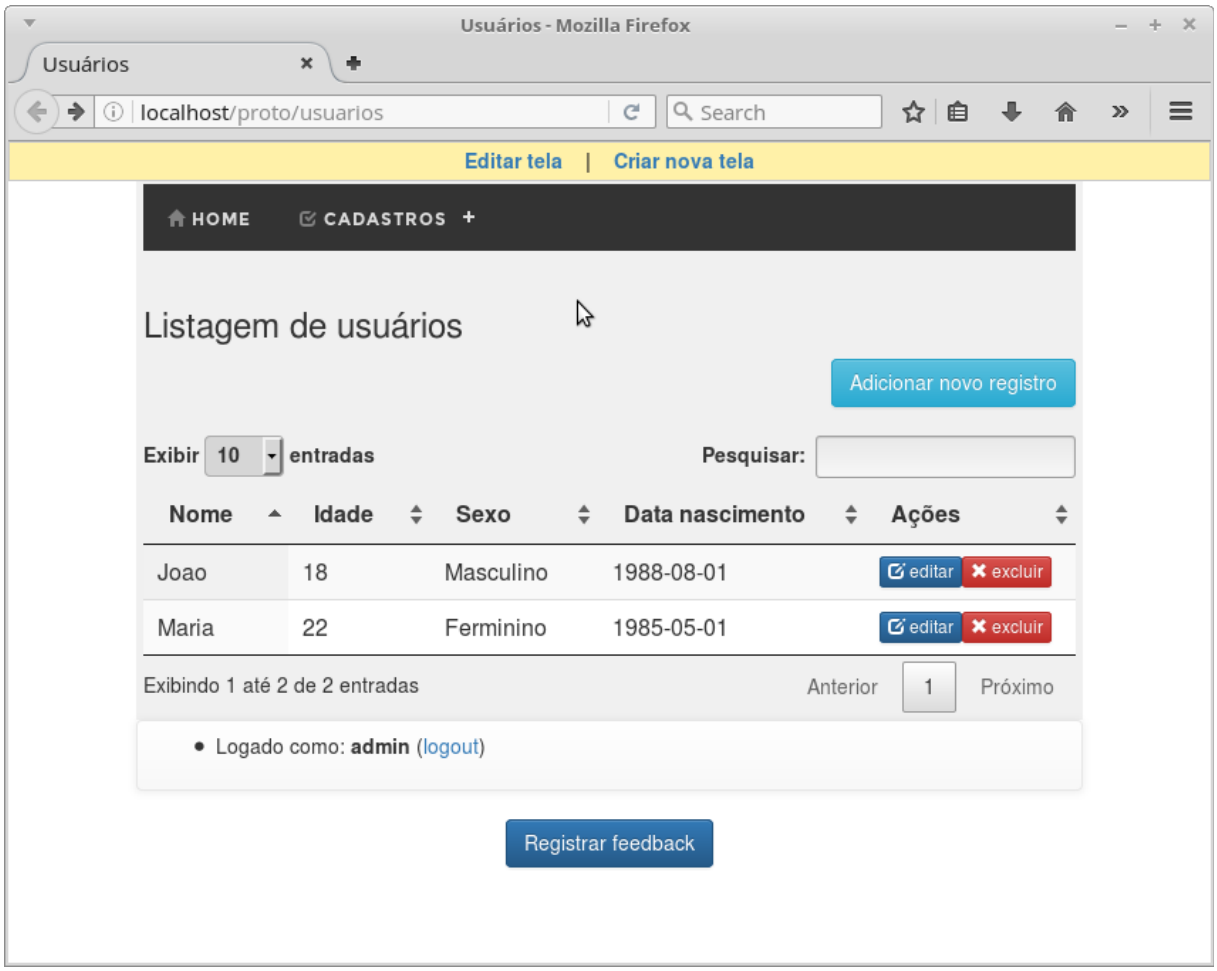
5.5 Visualização das telas

Uma vez que os analistas terminem o fluxo de criação de uma ou mais telas na parte administrativa do sistema, a demonstração *online* para o cliente visualizar como está o progresso já estará automaticamente disponível, bastando que ele acesse o endereço público do sistema. Na demonstração, o cliente poderá visualizar uma prévia de como será o sistema final real, como as interfaces ficarão e irão interagir entre si.

A Figura 14 apresenta um esboço da tela de visualização pela pessoa ou grupo de

pessoas que terão o papel de visualizar como o sistema está sendo modelado.

Figura 14 – Visualização do resultado do protótipo



Fonte: Elaborado pelo autor

5.6 Sistema de *feedbacks*

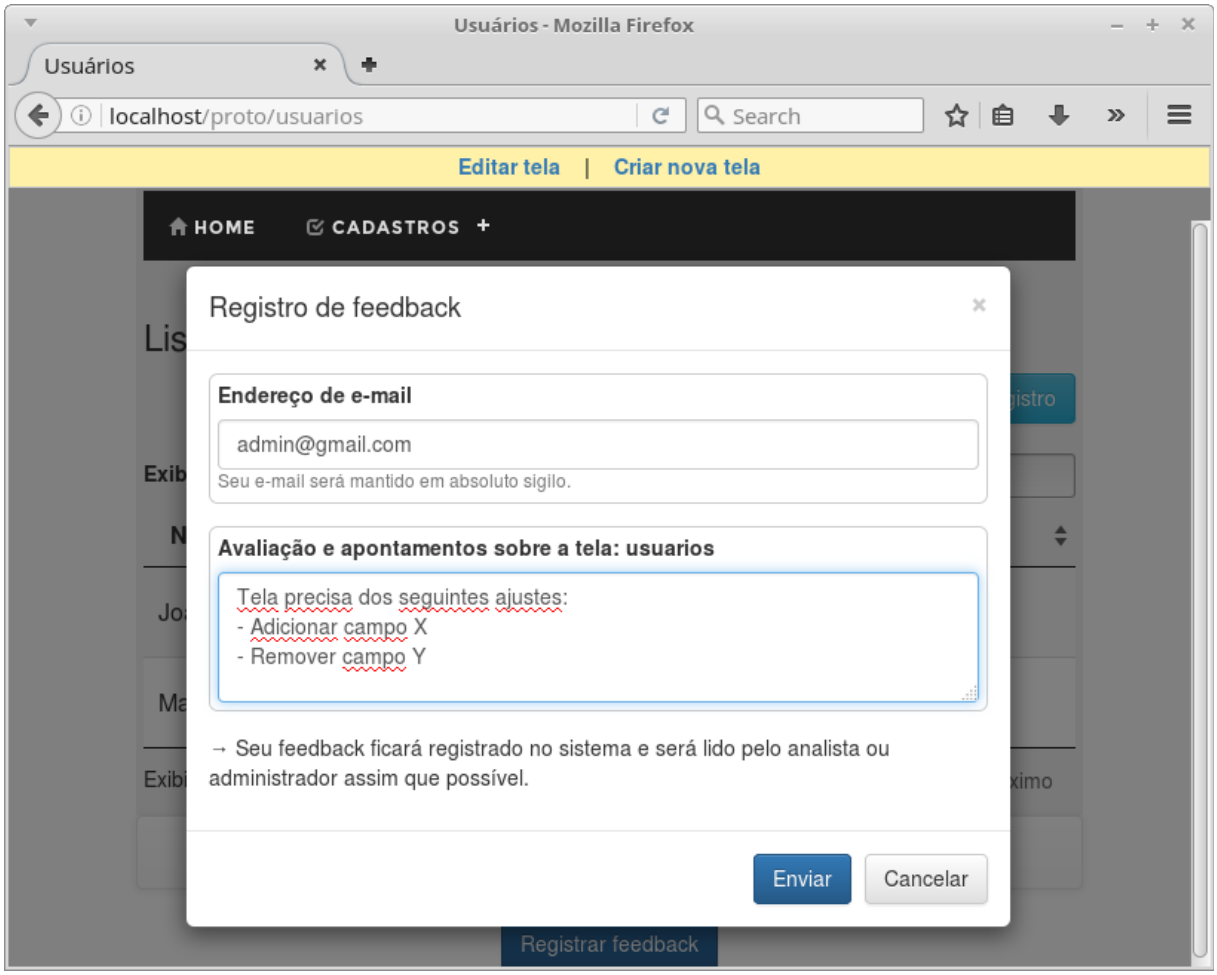
A ferramenta possui um sistema integrado de *feedbacks* que permite uma melhor interação do analista com o cliente. Esse sistema permite que um cliente possa submeter um comentário livre, de qualquer tela que tenha acesso na demonstração *online*.

Após o cliente submeter um *feedback* de alguma tela específica, a mensagem ficará registrada no servidor da aplicação, e será exibida para o analista quando o mesmo autenticar-se na seção administrativa, em forma de alerta de *feedback* pendente.

Caso o analista opte por ver a notificação, será aberta uma *popup* contendo dados de quem submeteu o *feedback*, data e hora, mensagem, e ações de marcar como lido ou responder. Caso seja respondido pelo analista, o cliente irá visualizar no rodapé da página, de forma destacada, a resposta do *feedback* que submeteu.

A Figura 15, representa a caixa de diálogo que é aberta quando o usuário logado clica no botão “Registrar feedback”, localizado no rodapé de cada página.

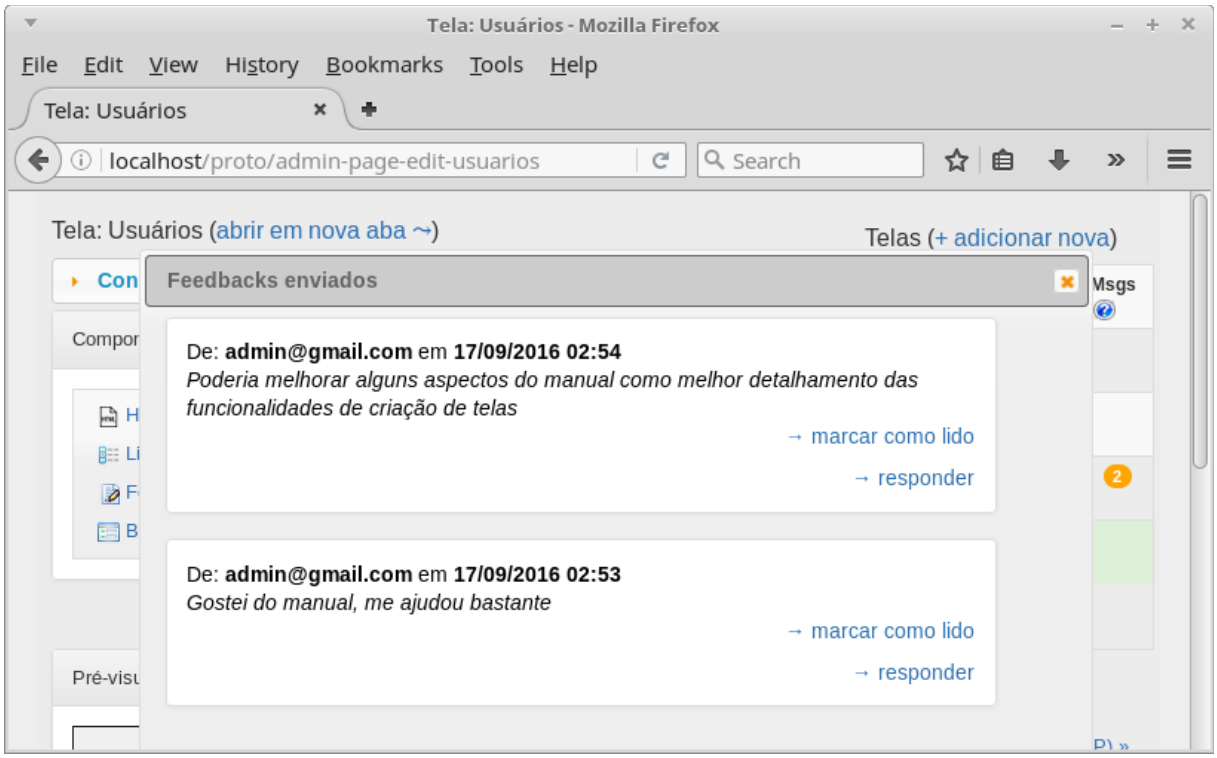
Figura 15 – Retorno de feedback pelo usuário



Fonte: Elaborado pelo autor

A Figura 16 demonstra uma caixa de diálogo que possui a finalidade de visualização e gerenciamento de *feedbacks* de uma tela específica. Cada *feedback* pode ser respondido ou marcado como lido para não ser mais exibido com destaque na área administrativa.

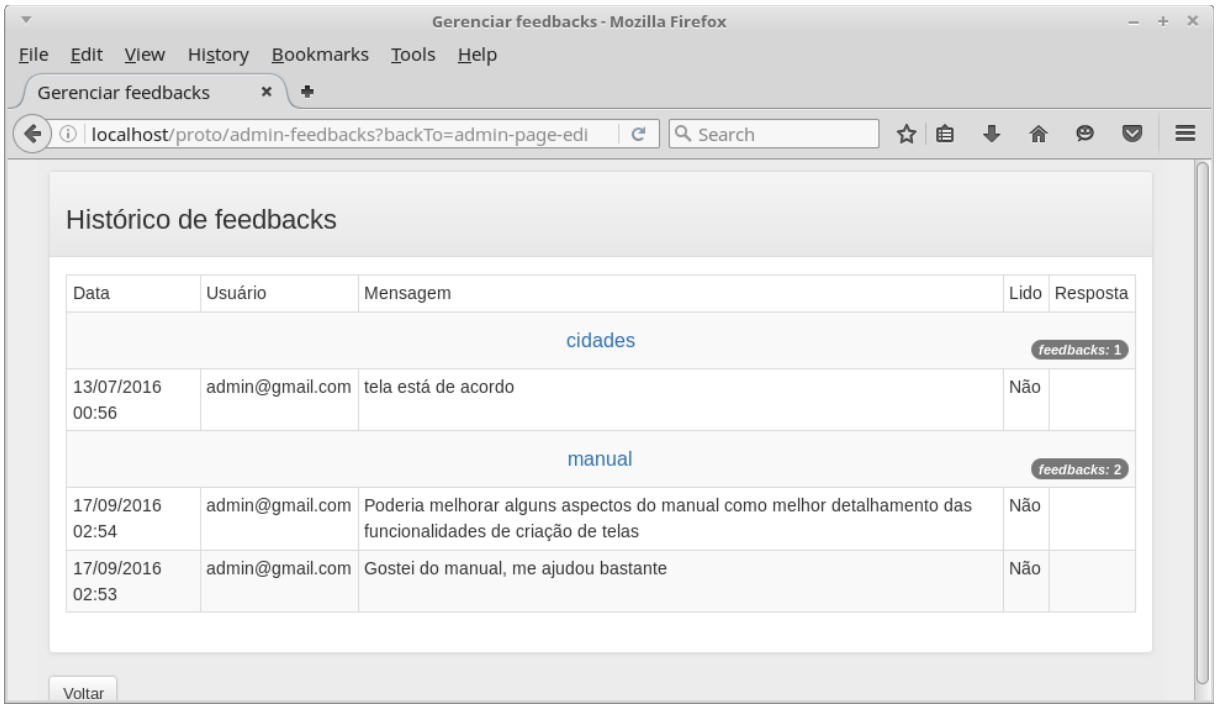
Figura 16 – Gerenciamento dos feedbacks enviados



Fonte: Elaborado pelo autor

A Figura 17 representa a *popup* de visualização do histórico de todos *feedbacks* enviados pelos clientes que acessam as interfaces, incluindo informações extras como, se foi lido ou não e a resposta retornada pelo analista. O histórico é ordenado pela data mais recente para a mais antiga. Essa *popup* é aberta quando o mouse é posicionado sobre a notificação de alerta de *feedbacks* pendentes de uma tela específica, na área administrativa principal.

Figura 17 – Visualização do histórico de feedbacks



Fonte: Elaborado pelo autor

5.7 Estrutura de diretórios e arquivos

A Quadro 10 apresenta a estrutura de diretórios definida na ferramenta, em sua estrutura interna.

Quadro 10 – Estrutura de diretórios da ferramenta

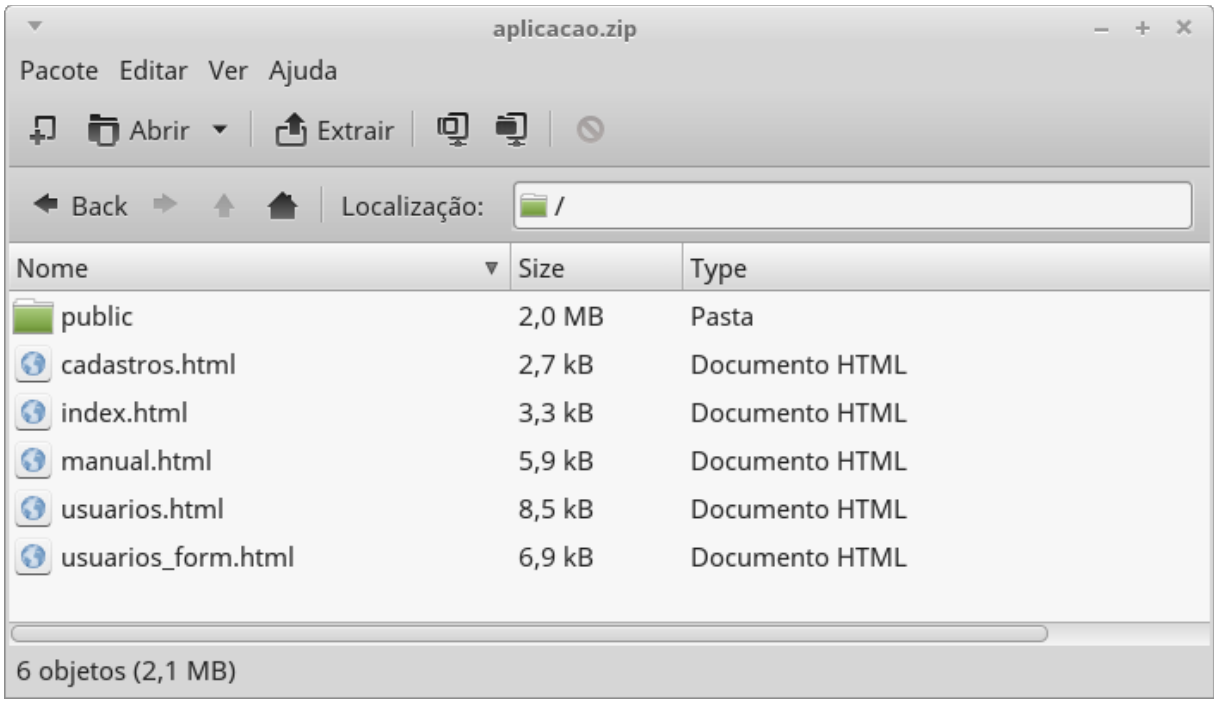
App/	Configurações, definições de páginas desenhadas (metadados)
Controllers/	Lógicas de roteamento (*routing*), transformação e processamento de dados, HTTP Request e Response
Public/	*Assets*, bibliotecas JavaScript e CSS, arquivos de temas e tudo que for arquivos públicos estáticos
Views/	Arquivos de *templates*, contendo basicamente HTML e variáveis que são recebidas dos c*ontrollers*
Vendor/	Contém todos arquivos do *core*, classes úteis e bibliotecas de terceiros, baixadas via Composer[4]

Fonte: Elaborado pelo autor

A Figura 18 ilustra um exemplo da estrutura de arquivos do projeto final baixado, através do link Download da aplicação no sistema, disponível na área administrativa. Cada arquivo HTML engloba uma tela de protótipo construída.

[4] Composer: Ferramenta para gerenciamento de dependências para o PHP

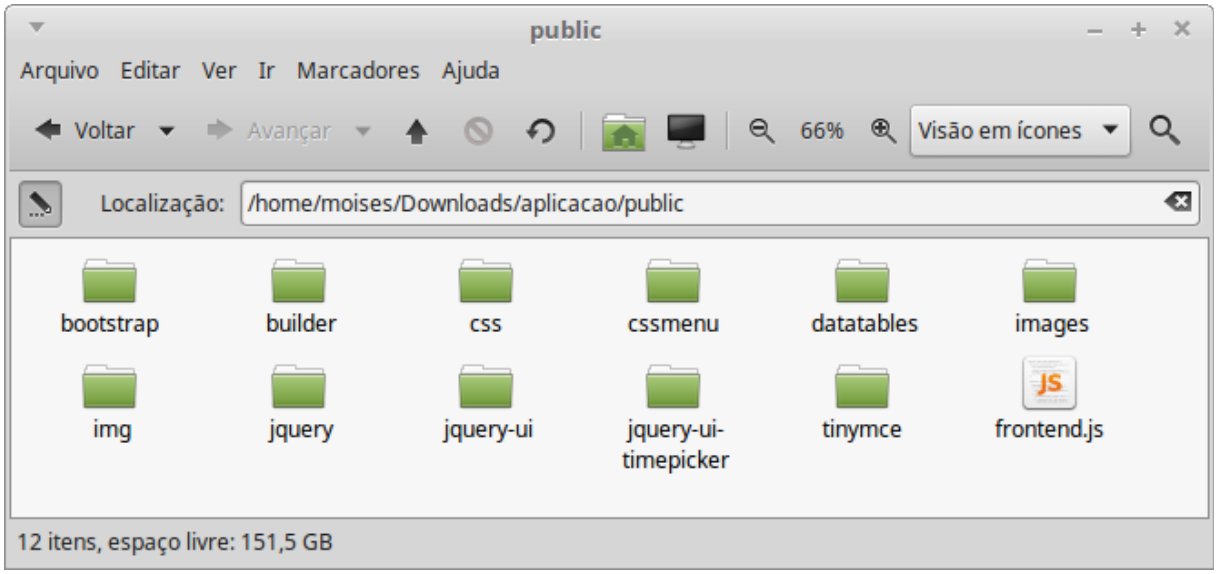
Figura 18 – Estrutura de arquivos da aplicação final gerada



Fonte: Elaborado pelo autor.

O diretório *public*, ilustrado na Figura 18, contém as bibliotecas *frontend*, imagens e estilos requeridos para o correto funcionamento das telas. A Figura 19 ilustra o conteúdo deste diretório.

Figura 19 – Estrutura do diretório de *assets* da aplicação gerada



Fonte: Elaborado pelo autor.

O Quadro 11 apresenta um parecer das características da ferramenta implementada. Os utilizados como critério são idênticos ao do Quadro 3, que compara as ferramentas semelhantes estudadas.

Quadro 11 - Comparativo final da ferramenta implementada

Característica	**Ferramenta implementada**
Open source	Sim
Multiplataforma	Sim
Geração de código-fonte	Sim
Compartilhamento de protótipos	Sim
Nível de usabilidade	Médio
Nível de fidelidade	Alto

Fonte: Elaborado pelo autor

6 VALIDAÇÃO

Este capítulo apresenta os passos efetuados para a realização da validação da ferramenta, ou seja, testá-la com usuários reais. O intuito disso foi de verificar qual a percepção que estes usuários possuem sobre a mesma, verificar se funciona de acordo com o esperado e também realizar pós-melhorias em suas funcionalidades, de acordo com o *feedback* obtido.

6.1 Descrição do teste realizado

Para verificação da percepção sobre a fase de requisitos e protótipos, a ferramenta foi validada por alunos de graduação do curso de informática de uma instituição de ensino superior, além de outros trabalhadores autônomos também da área.

O teste foi realizado por um total de 20 usuários, de áreas diversificadas que englobam: Analista de sistemas, Desenvolvedor de software, Gerente de projetos, Gerente de portfólio e Testador de software.

Cada usuário recebeu um *link* para acessar o sistema *online* em seu estado zerado, ou seja, de um igual ponto de partida. Feito isso, uma introdução e explicação oral foi realizada pelo autor da ferramenta, para posteriormente ser delegada uma tarefa de construção de um cadastro básico. O Quadro 12 representa a tarefa que foi aplicada para os usuários.

Quadro 12 – Tarefa efetuada com profissionais da área

1.	Acessar o endereço: http://ragob.com/proto/ e logar como admin / admin
2.	Criar uma nova tela com o caminho: cidades_form, e título: Formulário de cidades. Nesta tela, adicionar (arrastar) um Formulário com os campos: Cidade (text), Estado (text)
3.	Criar uma nova tela tela com o caminho: cidades, título: Cidades. Nesta tela, adicionar (arrastar) uma Listagem com as seguintes colunas: Cidade e Estado. Nas propriedades desta listagem, associar o Link de editar/novo com a tela: Formulário de cidades.
4.	Disponibilizar o link de acesso público da tela gerada (cidades) para uma outra dupla, simulando como se ela fosse um cliente, e solicitar para que a mesma registre um *feedback* da tela. Responder, através do sistema, o *feedback* registrado pela outra dupla, reproduzindo uma interação entre o analista e cliente.

Fonte: Elaborado pelo autor

Durante a execução da tarefa, alguns usuários fizeram questionamentos ao autor em relação a funcionalidades técnicas, modo de usar e algumas inconsistências encontradas, mas que não afetaram a conclusão da sequência estabelecida. A dúvida mais frequente foi sobre como deveria ser feito para interligar a listagem de Cidades com seu respectivo formulário de edição.

Ao término do exercício prático, cada pessoa respondeu individualmente a um questionário intitulado “Avaliação da ferramenta de protótipos de interfaces”, que continha questões pertinentes à usabilidade, grau de relevância, utilidade, dificuldades e possibilidade de sugestões e melhorias.

Para conclusão da validação da ferramenta, ficou em aberto para quem quisesse fazer mais algum comentário, e alguns usuários citaram que acharam a ferramenta útil, podendo ser utilizada em empresas para reduzir custos com retrabalho no desenvolvimento e melhorar a eficiência de comunicação com o cliente, no que diz respeito aos requisitos.

6.2 Respostas

O Quadro 13 apresenta o questionário de avaliação da ferramenta respondido por

profissionais da área. O questionário continha oito questões, sendo que seis delas eram de múltipla escolha e duas de resposta livre.

Quadro 13 – Questionário de avaliação da ferramenta

Como você avalia a usabilidade da ferramenta?	- Ótimo - Bom - Regular - Ruim - Péssimo
Qual o grau de relevância que você atribui para a ferramenta em processos de software?	- Alto - Médio - Regular - Baixo
Qual foi o grau de dificuldade de uso da ferramenta?	- Alto - Médio - Regular - Baixo
Em relação a utilidade da ferramenta, como você a considera?	- Ótimo - Bom - Regular - Ruim - Muito ruim
Você utilizaria este sistema na empresa em que trabalha?	- Sim - Não
Você acredita que o sistema de feedbacks da ferramenta pode aproximar o cliente na fase de concepção de software?	- Sim - Não
Você possui alguma sugestão de melhoria? Se sim, qual?	(campo de texto livre)
Você encontrou algum problema que o impossibilitou de usar a ferramenta? Se sim, qual?	(campo de texto livre)

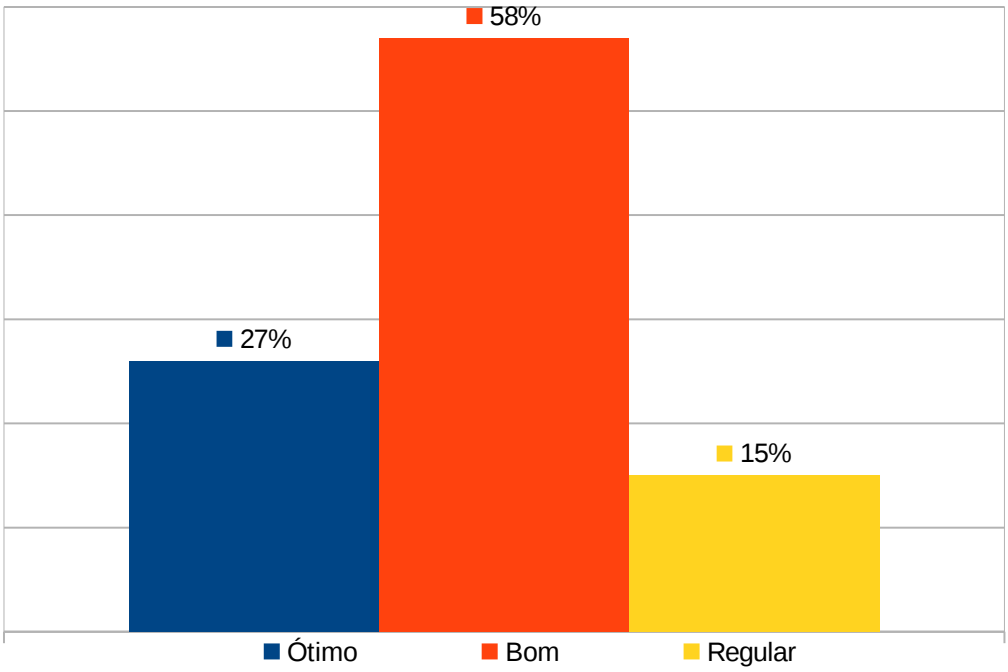
Fonte: Elaborado pelo autor

Sobre a pergunta “Você utilizaria este sistema na empresa em que trabalha?”, 58% respondeu que “Sim” e 42% que “Não”. Em relação a questão “Qual foi o grau de dificuldade de uso da ferramenta?”, 60% respondeu “Médio”, 25% achou “Regular”, e 15% “Baixo”. Já para a questão “Como você avalia a usabilidade da ferramenta?”, 50% achou “Regular”, 40% considera como “Bom” e 10% avaliou como “Ótima”.

A Figura 20 representa o gráfico de respondentes da pergunta sobre o grau de utilidade da ferramenta, indicando que a maior parte dos respondentes considera-a entre

ótima e regular.

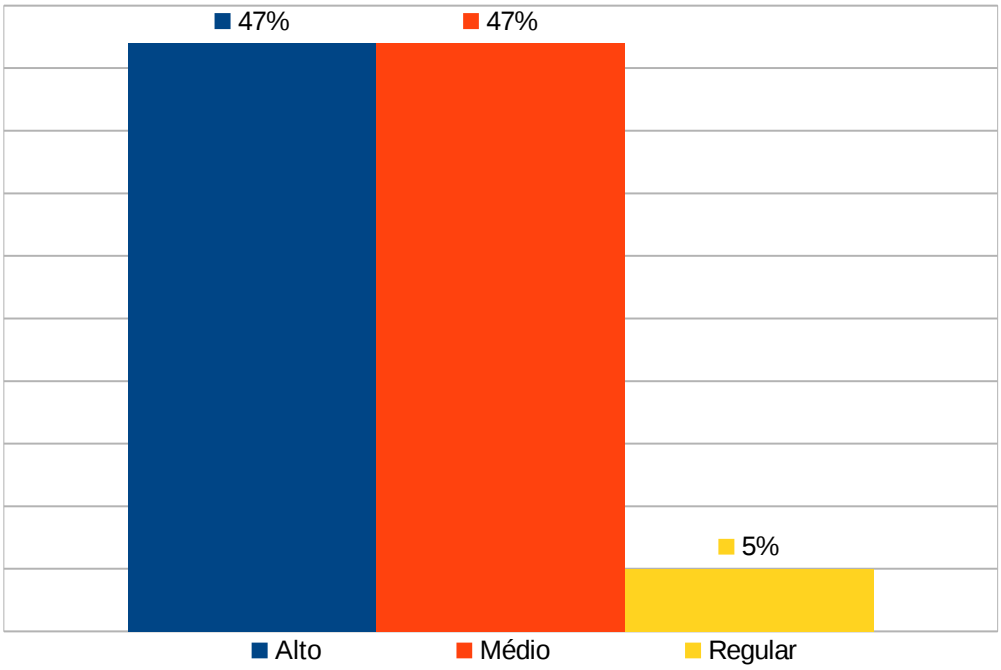
Figura 20 – Questão sobre o grau de utilidade da ferramenta



Fonte: Elaborado pelo autor

A Figura 21 apresenta graficamente o resultado da questão sobre o grau de relevância que o usuário percebe em relação a ferramenta testada. Nela, constata-se que a maior parte dos respondentes considera-a entre médio e alto.

Figura 21 – Questão sobre o grau de relevância da ferramenta



Fonte: Elaborado pelo autor

Além de questões de múltipla escolha, também foram perguntadas outras questões de resposta descritiva, opcionais. O Quadro 14 apresenta as respostas sobre uma questão de sugestão de melhorias.

Quadro 14 – Respostas de questão sobre sugestões de melhorias para a ferramenta

Você possui alguma sugestão de melhoria? Se sim, qual?
Acho o conceito do software muito útil, acredito que pouparia muito custo e reduziria muito a dor de cabeça durante o desenvolvimento. Acredito que toda empresa que desenvolve software personalizado deveria ter algo similar. Acredito que o projeto esta está em protótipo, existem diversos ajustes para tornar a ferramentas comercial, mas está no rumo certo.
Tentar deixar a tela mais intuitiva tanto para o usuário final como para o desenvolvedor, tentar evitar que seja necessário um manual para se fazer o básico.
Um layout melhor, parece muito confuso.
Tornar o processo mais intuitivo. Talvez telas mais limpas ou fazer testes sem muitas explicações para ver como as pessoas tentariam encontrar as respostas.
Ao colocar o mouse sobre um campo, poderia informar o que mesmo faz.
Melhorar a usabilidade, achei meio confuso a forma de cadastro.
Acredito que a ferramenta se tornaria mais útil se fosse possível montar telas mais complexas ao invés de somente crud.
Acho que como protótipo esta muito bom. Mas futuramentr acho que novos componentes deviam ser disponibilizados, pois as telas na prática sao bem mais complexas que simples formulários.
A princípio nenhuma, parece bom e funcional.
Ter como exportar artefatos para formalizar junto à documentação do projeto, talvez colocar screenshots dentro de documentos de especificações de requisitos. Isto é importante para atender às boas práticas de certificações, que precisam de tudo 'redondinho' / documentado.

Fonte: Elaborado pelo autor

A partir da avaliação realizada, conclui-se que a maior parte dos avaliadores considera a ferramenta como positiva, e que caso seu uso fosse incluso no fluxo do processo de análise e elicitação de requisitos de uma empresa, com isso traria benefícios para a mesma no que diz respeito a melhoria na eficiência do mesmo.

Parte dos usuários relataram que a usabilidade para realizar certas tarefas na construção de telas na área administrativa poderia ser melhorada em certos aspectos, o que não invalida a ferramenta, mas abre uma possibilidade de refinamentos e melhorias futuras para ser utilizado em produção em uma empresa real.

7 CONSIDERAÇÕES FINAIS

Após a realização de todas as etapas deste trabalho e a aplicação da metodologia e execução, concluiu-se que a inclusão de uma ferramenta de protótipos de interfaces e demonstração real ao cliente no processo de criação de software pode ser um fator diferencial para o sucesso ou não do mesmo.

A partir dos resultados obtidos através de questionários e testes feitos na ferramenta com profissionais da área, pode-se observar que grande parte a vê como um sistema que pode auxiliar empresas a terem resultados mais assertivos no desenvolvimento de software.

Outro fator que pode ser observado no *feedback* de profissionais da área, é que boa parte questionou sobre a usabilidade e funcionalidades não estarem ainda em plena condição de uso por empresas reais. Isso se deve ao fato de que o sistema proposto ainda não é uma ferramenta homologada para ser utilizada em produção.

Entende-se, então, que os objetivos principais e secundários do trabalho foram atingidos, tendo sido a ferramenta implementada de acordo com o que foi especificado, e podendo ser evoluída ou modificada futuramente, de acordo com os requisitos necessários.

Alguns itens para possíveis trabalhos futuros não implementados foram identificados. Dentre eles, cita-se:

- melhoria na área administrativa para determinadas funcionalidades, como a criação de telas de inserção, atualização, listagem e remoção, de modo que fique

mais intuitivo para o analista;

- notificação via e-mail quando houver novo *feedback* enviado pelo usuário ou uma tela for modificada pelo analista. Adição de novos componentes, como *popups*, campos complexos de formulário como upload de arquivo e auto-preenchimento de termos;
- possibilidade de criação de regras mais complexas de validação e exibição de campos e colunas quando houver determinado contexto;
- realizar o versionamento das telas criadas de acordo com os *feedbacks* enviados pelo cliente, com o intuito de criar um histórico das mesmas.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Carla. **Introdução ao Teste de Software**. Obtida via internet. Disponível em: <http://www.linhadecodigo.com.br/artigo/2775/introducao-ao-teste-de-software.aspx>. Acesso em: 02 abr. 2016.

CAMARINI, Bruno. **PROTOTIPAÇÃO E SUA IMPORTÂNCIA NO DESENVOLVIMENTO DE SOFTWARE**. Obtida via internet. Disponível em: <http://dextra.com.br/prototipacao-e-sua-importancia-no-desenvolvimento-de-software/>. Acesso em: 05 abr. 2016.

CARVALHO, Lucas Augusto. **O que é jQuery?**. Obtida via internet. Disponível em: <http://www.webulando.com.br/jquery/o-que-e-jquery>. Acesso em: 26 abr. 2016.

CERVO, Amado Luiz. **Metodologia Científica**. 6. ed. São Paulo. 2006.

DELAMARO, Márcio Eduardo. **Introdução ao teste de software**. Rio de Janeiro, 2007.

ENGHOLM Júnior, Hélio. **Engenharia de software na prática**. São Paulo. 2010.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2007.

KOSCIANSKI, André. **Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. 2. ed. São Paulo. 2007.

LINOWSKI. **Wireframes**. Obtida via internet. Disponível em: <http://wireframes.linowski.ca/2009/02/interactive-axure-prototype/>. Acesso em: 17 out. 2016.

LOPES, S. **O que é PHP**. Obtida via internet. Disponível em: <https://www.oficinadanet.com.br/artigo/659/o_que_e_php>. Acesso em: 26 abr. 2016.

MACHADO, Felipe Nery Rodrigues. **Análise e gestão de requisitos de software: onde nascem os sistemas**. 2. ed. São Paulo. 2014.

MACRORIE, Quinn. **5 Fantastic UX Prototyping Tools**. Obtida via internet. Disponível em: <https://constructive.co/insights/5-fantastic-ux-prototyping-tools-part-1/>. Acesso em: 18 out. 2016.

MALHERBI, Eduardo. **Prototipação de Sistemas utilizando a Ferramenta Balsamiq Mockup**. Obtida via internet. Disponível em: <http://www.devmedia.com.br/prototipacao-de-sistemas-utilizando-a-ferramenta-balsamiq-mockup/27232>. Acesso em: 17 out. 2016.

MOLINARI, Leonardo. **Inovação e Automação de Testes de Software** . 1. ed. São Paulo, 2014.

OFICINADANET. **O que é MPS.br?**. Obtida via internet. Disponível em: <https://www.oficinadanet.com.br/artigo/desenvolvimento/melhoria-de-processos-do-software-brasileiro—mpsbr>. Acesso em: 7 nov. 2016.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. 7 ed. 2011.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. 2012.

SOUZA, Diogo. **Explorando o jQuery UI**. Obtida via internet. Disponível em: <http://www.devmedia.com.br/explorando-o-jquery-ui/30316>. Acesso em: 26 abr. 2016.

UTTERBACK, Benjamin. **O que é o Bootstrap? – Verdades e mitos**. Obtida via internet. Disponível em: <https://www.prestashop.com/blog/pt/2014/03/06/o-que-e-o-bootstrap-verdades-e-mitos-parte-1-de-2/>. Acesso em: 26 abr. 2016.