Fatec Cruzeiro Prof. Waldomiro May
Revista Científica H-TEC Humanidades & Tecnologia

TRIZ – FERRAMENTA CASE PARA PRODUÇÃO DE SOFTWARE COM METODOLOGIA ÁGIL PARA PEQUENAS E MÉDIAS EQUIPES

Marcos Paulo da Silva[1]

Bruno Bustamante Ferreira Leonor[2]

RESUMO

O desenvolvimento de software requer uma abordagem sistematizada e dinâmica. A dificuldade em conciliar esses fatores pode trazer problemas relacionados a prazos e custos. Este trabalho propõe uma ferramenta CASE que auxilie nos processos gerenciais, visando eficiência e eficácia. O aplicativo foi programado para web com design responsivo, por meio das linguagens HTML, CSS, PHP e SQL. A metodologia de desenvolvimento foi o *Scrum*. O projeto foi submetido a controle de versões e a testes. O resultado alcançado consiste na aplicação online denominada Triz. Espera-se com isso, fornecer um meio para a diminuição do índice de fracasso de projetos e aumento da qualidade de softwares produzidos. Além de familiarizar pequenas e médias equipes com a sistematização de tarefas e responsabilidades, bem como com outros conceitos de gerência e Engenharia de Software.

Palavras-chave: Ferramenta CASE; Gerência de Projetos; Engenharia de Software.

TRIZ - CASE TOOL FOR SOFTWARE PRODUCTION WITH AGIL METHODOLOGY FOR SMALL AND MEDIUM TEAMS

ABSTRACT

A software development program requires a systematic and dynamic approach. The difficulty in reconciling these factors can lead to problems related to time and costs. This project proposes a CASE tool that assists a management process efficiently and effectively. The application has been programmed on the web with responsive design, through the languages HTML, CSS, PHP and SQL. The development methodology was Scrum. The project was submitted to a version control and testing. The result achieved is the online application Triz. It is expected thereby the decrease of the failure rate in the projects and the increase of the quality of the software programs produced. Besides of this it familiarizes small and medium teams with the systematization of the tasks and responsibilities. In it was used other concepts of management and Software Engineering.

Keywords: CASE tool; Project Management; Software Engineering.

[1] Possui graduação em Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia Prof. Waldomiro May - FATEC – CRUZEIRO. E-mail: mpsacademico@gmail.com

[2] Doutorando em Computação Aplicada pelo Instituto Nacional de Pesquisas Espaciais – INPE e professor na Faculdade de Tecnologia Prof. Waldomiro May - FATEC – CRUZEIRO. E-mail: brunobfl@yahoo.com.br

1. INTRODUÇÃO

Os diversos segmentos humanos estão sendo freneticamente informatizados. Tarefas que antes eram feitas por pessoas, agora são desempenhadas eficientemente por máquinas e outros recursos computacionais. A tecnologia mostra-se eficaz no processo da evolução do homem. Os processos que conduzem a concepção desses recursos computacionais estão envolvidos em equipes com vários integrantes distintos, com diferentes habilidades. Porém, conforme dados históricos mostram e recentes pesquisas apontam, equipes de desenvolvimento de software possuem sérios problemas com prazos, custos e, consequentemente, com a qualidade do projeto e do produto.

Nesse sentido, surge o seguinte questionamento: como auxiliar esses processos sem impactar a política de desenvolvimento de software em pequenas e médias equipes? Uma ferramenta CASE (*Computer-Aided Software Engineering)* gerencial altamente amigável seria uma proposta para solução desse problema. Dessa maneira, o objetivo desse projeto foi desenvolver tal ferramenta, priorizando a integração entre o time de desenvolvimento, visando eficiência e eficácia. Sendo que, a aplicação propõe uma maneira de trazer melhorias no processo de desenvolvimento, aplicando uma metodologia ágil, inicialmente o *Scrum*, visando diminuir problemas relacionados a prazos, custos e qualidade de projeto e produto.

Os principais autores na Fundamentação Teórica são Sommerville e Pressman devido as suas vastas obras na área de Engenharia de Software. Além de contribuições de autores sobre metodologias ágeis, incluindo as do cocriador do Scrum, Jeff Sutherland. A metodologia de desenvolvimento utilizado durante todo o projeto foi o Scrum. O aplicativo foi programado para web com design responsivo, por meio das linguagens HTML, CSS, PHP e SQL. O projeto foi submetido a controle de versões e a testes para validação dos códigos.

O resultado alcançado consiste na criação da aplicação denominada Triz. Espera-se com isso, fornecer um meio para a diminuição do índice de fracasso de projetos e aumento da qualidade de softwares produzidos. Além de familiarizar pequenas e médias equipes com a sistematização de tarefas e responsabilidades, bem como com outros conceitos de gerência e Engenharia de Software.

2. FUNDAMENTAÇÃO TEÓRICA

O trabalho utilizou a pesquisa bibliográfica (SEVERINO, 2007, p. 122) para a fundamentação teórica. Nesta seção são citados conceitos sobre a Engenharia de Software e como ocorre o desenvolvimento de programas sob o aspecto gerencial. Assim como demonstra a organização do ciclo de vida deles em relação às metodologias tradicionais e ágeis. Além disso, apresenta uma rápida visão do cenário nacional de software.

2.1 Engenharia de Software

A concepção de um programa de computador (software) tende a possuir um alto nível de complexidade, com um amplo número de atividades que precisam ser executadas para se chegar a uma provável conclusão do trabalho de criação. De acordo com Rezende (2005), a Engenharia de Software é a disciplina de planejamento, desenvolvimento e manutenção de sistemas computacionais, em que ocorre a formulação de soluções inteligentes para resolução de problemas conciliando atividades, recursos, custos e datas. Na engenharia de software aplicam-se teorias, métodos e ferramentas, para se encontrar soluções, além disso, relacionada aos aspectos da produção de software. Nesse sentido, Sommerville (2007, p. 5) afirma que a engenharia de software "não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software".

Os benefícios da aplicação dos conceitos da Engenharia de Software estão em tornar a execução das etapas do projeto mais claras e organizadas. Segundo Sommerville (2007, p. 5), a equipe deveria adotar "uma abordagem sistemática e organizada em seu trabalho, que é, frequentemente, a maneira mais eficaz de produzir software de alta qualidade".

Além do caráter técnico das tarefas, "qualquer abordagem de engenharia (inclusive engenharia de software) deve estar fundamentada em um comprometimento organizacional com a qualidade" (PRESSMAN, 2006, p. 39). Sendo que, é papel da equipe criar um comprometimento interno a fim de produzir melhores resultados, com foco sempre na manutenção da qualidade do projeto e do produto.

A Engenharia de Software é composta e assistida por diversos métodos e uso de ferramentas que, por exemplo, trazem apoio automatizado ou semiautomatizado. A integração entre esses elementos é um facilitador das atividades de projeto e para Pressman (2006, p. 40) "quando as ferramentas são integradas, de modo que as informações criadas por uma ferramenta possam ser usadas por outra, é estabelecido um sistema para o suporte ao desenvolvimento de software, denominado *engenharia de software com o auxílio do computador"*.

O planejamento, modelagem, construção e emprego de um software com a não utilização dos recursos da Engenharia de Software, torna o projeto mais frágil, com maiores taxas de insucesso perante cenários cada vez mais complexos de atuação da humanidade.

2.2. Desenvolvimento de Software Sob o Aspecto Gerencial

Por muitas vezes, o aspecto gerencial envolvido à Engenharia de Software é omitido. No entanto, a correta organização e gerência de elementos como prazos e orçamentos são de vital importância para o sucesso de qualquer empreendimento. Os programas de computador, normalmente, são desenvolvidos com base em projetos. De modo geral, um projeto é um empreendimento temporário que tem por objetivo fornecer um produto ou serviço singular, geralmente com restrições orçamentárias e de prazo (MAXIMIANO, 2006, p. 26). No início do projeto, é de vital importância saber com clareza do que se trata o produto; ainda segundo Maximiano (2006, p. 58), "nenhuma equipe de projeto deve avançar no planejamento operacional (planejamento das atividades e dos recursos) sem ter uma noção clara do produto a ser fornecido".

Dessa forma, o projeto de software, como qualquer outro tipo de projeto, necessita da disciplina de Administração de Projetos. Senão, os aspectos gerenciais da criação do produto acabam por deteriorar as bases do projeto e ocasionar o fracasso de todos os esforços.

A gerência do projeto, pela disciplina de administração, deveria conforme Rezende (2005, p. 6) 'prover os fundamentos para o gerenciamento de projetos de desenvolvimento de software, incluindo as atividades, de planejamento que envolva estimativas de recursos e cronogramas, bem como de definição de estrutura organizacional, formas de controle e de liderança'.

O descuido no controle e tratamento de questões relacionadas à gerência podem afundar um projeto, fato que não é tão raro de acontecer. Para Phillips (2003, p. 11), o problema principal está na visão. Que em gerência de projetos, significa "[...] a capacidade de ver com objetividade o

intangível e reconhecer as ações necessárias para chegar até ele". A Engenharia de Software munida de controle administrativo traz uma abordagem capaz de solucionar a maior parte desses problemas no aspecto gerencial. Conforme Maximiano (2006, p. 114), com o encerramento de um projeto é possível verificar-se o grau de sucesso dele, que pode ser visto como o nível de satisfação dos interessados no resultado.

2.2.1 Crise do Software e Anticrise

Durante o desenvolvimento de um software, podem surgir vários problemas que diminuem a qualidade do produto. O termo "Crise do Software", usado desde os anos 60, refere-se "[...] quando o software não satisfaz seus envolvidos, sejam clientes ou usuários, desenvolvedores ou empresa (organização privada ou pública)" (REZENDE, 2005, p. 8).

O que ocorre são imensas dificuldades na determinação de estimativas de cronogramas e custos, que acabam por extrapolar todas as expectativas de conclusão do projeto. As falhas nas medidas do tempo e custo causam prejuízos a todos os envolvidos. A equipe de desenvolvimento se desgasta e o cliente perde somas de investimentos. Não existem métricas capazes de medir com precisão quantitativamente esses elementos envolvidos no projeto (REZENDE, 2005, p. 9), tornando essas estimativas, na maioria das equipes, baseadas apenas em achismos.

A "Anticrise do Software" trata-se da união entre a organização, os usuários do sistema e a equipe de desenvolvimento, a fim de minimizar a ocorrência desses problemas, através de melhores atitudes gerenciais e estratégicas (REZENDE, 2005, p. 11).

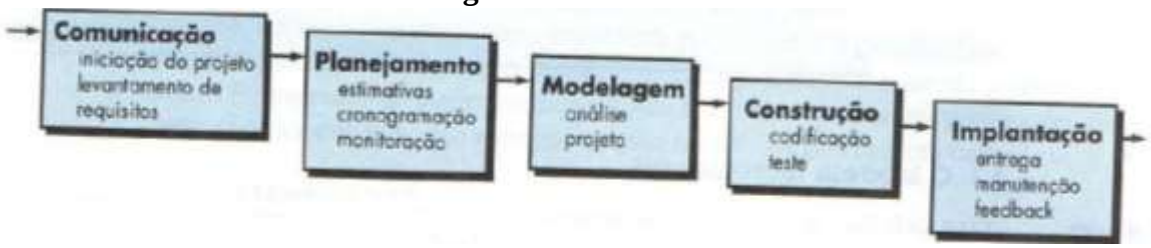
2.3 Ciclo de Vida de um Software

Um software tem um início, um meio e um fim. Isso se trata de um ciclo de vida, que dura alguns anos. Conforme Rezende (2005, p. 41), esse ciclo é composto pelas fases de concepção, construção, implantação, implementações, maturidade e utilização plena, declínio, manutenção e descontinuidade do software. Essas etapas são parecidas em todos os ciclos.

Todas essas fases podem ser organizadas de diferentes formas, em relação à ordem de execução e dependência entre elas. Cada uma dessas organizações dá origem à denominação a um tipo de ciclo de vida, juntamente com a metodologia de desenvolvimento. Alguns ciclos de vida

mais tradicionais, como o Modelo em Cascata (*Waterfall*), determinam fases bem definidas, em ordem fixa, e com alta dependência entre a execução das etapas. O modelo em cascata já é aplicado há muitos anos e foi revolucionário nos primórdios da organização das fases de um projeto (SOMMERVILLE, 2007. p. 44).A Figura 1, a seguir, ilustra o Modelo em Cascata, que por vezes é chamado de ciclo de vida clássico, segundo Pressman (2006, p. 59).

Figura 1: O modelo cascata



Fonte: Pressman (2006, p. 60).

Conforme se observa na Figura 1, segundo Pressman as cinco grandes etapas do Modelo em Cascata são subdivididas em atividades bem definidas desde o início até o fim. No entanto, com o passar do tempo, pela mudança dos mercados e o incrível aumento da dinâmica da tecnologia da informação ocasionada pela evolução das tecnologias de telecomunicação, surgiu outra visão quanto o desenvolvimento. Certos projetos determinantemente não se encaixam nos modelos tradicionais de desenvolvimento. Com base nessas necessidades nasce o conceito de agilidade.

2.4 Metodologias Ágeis

O Manifesto Ágil (COCKBURN et al., 2001) consiste em uma lista dos itens mais valorizados no desenvolvimento ágil de software, acertado por um conjunto de importantes nomes da área de desenvolvimento. Nele são definidos quatro (4) importantes valores, formados pela maior valorização de um elemento em contraposição ao outro. São os valores: indivíduos e a interação entre eles mais que processos e ferramentas; software em funcionamento mais que documentação abrangente; colaboração com o cliente mais que negociação contratual; respostas a mudanças mais que seguir um plano (GOMES, 2013, p. 3). Os métodos ágeis são baseados no manifesto ágil. Eles compartilham dos mesmos princípios, por isso, a maioria deles opera em ciclos curtos, com duração de algumas semanas estabelecendo contato frequente com os interessados. Cada um desses ciclos é completo e

> [...] contém todas as etapas necessárias para que se realize um incremento no produto, ou seja, no software: planejamento, análise, design, codificação, testes e documentação. Em métodos não ágeis, também conhecidos como métodos tradicionais, geralmente se encontra um processo em cascata em que todas as etapas citadas são executadas uma única vez e em sequência (GOMES, 2013, p. 4).

A agilidade não é contrária aos processos e as ferramentas, a documentação e aos planos; o que procura é que estejam a serviço das atividades das equipes (BRASIL, 2013, p. 17). Para Pressman (2006, p. 58), o desenvolvimento ágil traz uma nova filosofia quanto ao modelo de desenvolvimento, onde são priorizados elementos diferentes das mais tradicionais, tornando o processo mais liberal e orientado às pessoas que estão envolvidas. Nesse sentido, tendo em vista a mutabilidade dos cenários mercantis atuais, a engenharia ágil "representa uma alternativa razoável para a engenharia de software convencional para certas categorias de software e certos tipos de projeto de software. Tem sido demonstrado que ela entrega rapidamente sistemas bem sucedidos" (PRESSMAN, 2006, p. 58).

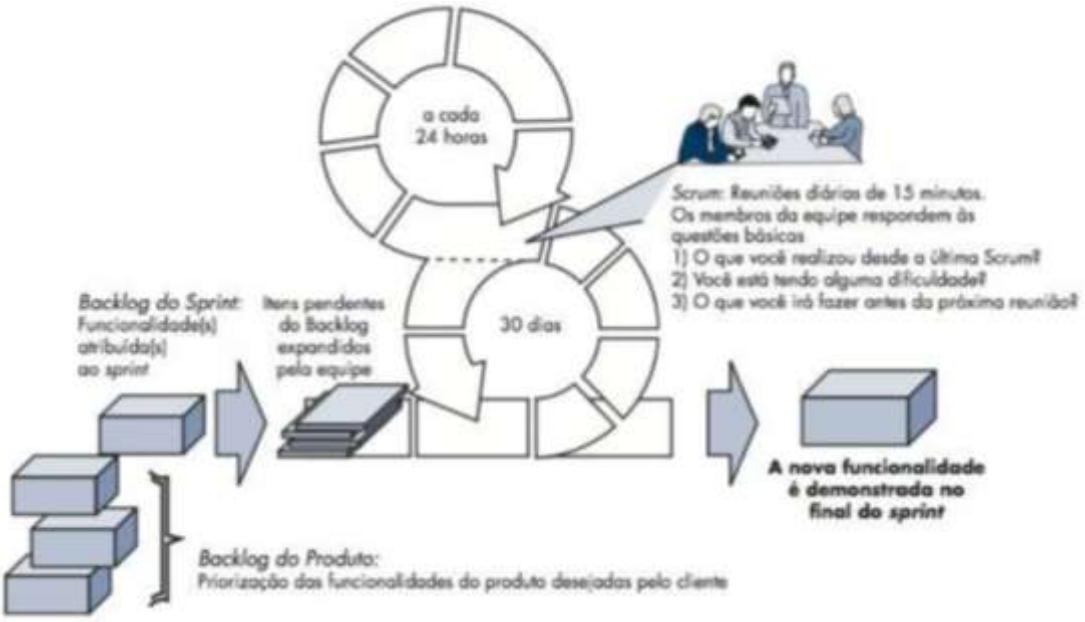
Por esse motivo, as metodologias ágeis acabam por ganhar cada vez mais espaço no mercado, visto que estão sendo amplamente adotadas pelas equipes de desenvolvimento. Por outro lado, a agilidade não está apenas vinculada a dinâmica dos mercados e agilidade em respostas, mas também a uma filosofia que cria um fenômeno agregador entre a equipe e o cliente. Segundo Pressman (2006, p. 83), "ela incentiva a estruturação de atitudes em equipe que tornam a comunicação mais fácil [...] e enfatiza a entrega rápida do software operacional e diminui a importância dos artefatos intermediários". Existem vários exemplos de metodologias ágeis, o mais difundido é o Scrum.

2.4.1 Visão Geral Sobre o Scrum

O Scrum existe desde o início dos anos 1990. O termo vem de um movimento do jogo de rúgbi, o movimento consiste em passar a bola enquanto todo o time avança (SUTHERLAND, 2014, p. 39), trata-se de uma clara analogia de como o time deveria funcionar. Alguns dos benefícios que ele traz são: entregas frequentes de retorno ao investimento dos clientes; redução dos riscos do projeto; maior qualidade no produto gerado; mudanças utilizadas como vantagem competitiva; visibilidade do progresso do projeto; redução do desperdício; aumento de

produtividade (SABBAGH, 2013, p. 4). O Scrum é composto por papéis, eventos e artefatos. A Figura 2, abaixo, mostra o fluxo do processo Scrum com algumas de suas práticas.

Figura 2: O fluxo do processo Scrum



Fonte: Pressman (2006, p. 96).

De acordo com a Figura 2, é possível perceber como é dinâmico o fluxo do processo Scrum. As diferenças desse fluxo com o Modelo em Cascata (Figura 1) são contrastantes. Algumas pesquisas que buscavam mostrar de maneira científica o impacto do método ágil Scrum demonstram sucesso na iniciativa. Em um caso especial, a implantação do Scrum nos projetos de software em uma empresa de base tecnológica trouxe a correlação sobre o que a literatura do Scrum atribui a ele e os impactos reais. Sendo que,

> a equipe participante da pesquisa sentiu diretamente quatro dos benefícios do Scrum: melhoria na comunicação e aumento da colaboração entre envolvidos; aumento da motivação da equipe de desenvolvimento; diminuição no tempo gasto para terminar o projeto (prazo); diminuição do risco do projeto (menor possibilidade de insucesso). Outros dois benefícios presentes na literatura foram medidos e notados pelos dados quantitativos do projeto piloto e de outros projetos da empresa: diminuição dos custos de produção (mão de obra) e aumento de produtividade da equipe. (CARVALHO; MELLO, 2012).

Entretanto, para implantar o Scrum, é necessário realizar mudanças profundas. Carvalho e Mello (2012) afirmam que "uma das grandes barreiras para implantação Scrum é a necessidade da mudança da cultura organizacional".

2.5 Análise das Metodologias Ágeis no Cenário Nacional

A análise de projetos na área da tecnologia da informação é realizada décadas atrás. Apesar de o interesse pela medição dos processos de software ser antigo, são escassos os dados. Em meados de 1994, em um contexto global, o Standish Group pesquisou e concluiu que 31% dos projetos eram cancelados, 88% ultrapassavam o prazo e/ou orçamento e as médias de estouro de prazo e orçamento eram, respectivamente, 222% e 189% (PHILLIPS, 2003, p. 10). No contexto brasileiro, segundo Fernandes e Teixeira (2004, p. 27), uma das pesquisas que melhor descrevem o meio nacional sobre qualidade e produtividade em software é executada pelo Programa Brasileiro de Qualidade e Produtividade (PBQP). Os dados da última pesquisa datam de 2009.

Em relação às metodologias ágeis estima-se o tamanho do produto de software, 21,9% de 264 organizações respondentes utilizam métricas relacionadas a métodos ágeis, tais como *Story Points*, *Planning Poker*, *User Stories* e *Ideal Days*. Sobre os critérios utilizados na seleção de fornecedores de desenvolvimento e manutenção de software, algumas organizações citaram o uso de Software Livre e metodologia Scrum (BRASIL, 2010, p. 81). A Figura 3 exibe a relação dos outros critérios que são utilizados pelas organizações respondentes para seleção de fornecedores.

Figura 3 – Critérios para seleção de fornecedores

Fonte: Brasil (2010, p. 81).

Conforme se verifica na Figura 3, a metodologia ágil é um dos requisitos com tendências de crescimento. Nessa pesquisa, ficou claro que "existe a presença de metodologias ágeis como

forma de organização para o processo de desenvolvimento de software“ (BRASIL, 2010, p. 89). Com esses dados, é possível perceber que as metodologias ágeis estão ganhando seu espaço.

2.6 Ferramentas CASE

O desenvolvimento de software pode necessitar de muito esforço e possuir alto grau de complexidade. A combinação desses elementos pode ocasionar sérios problemas no projeto, causando atrasos e defeitos. Sendo que, dentro da administração de projetos, deveriam ser almejados ideais como a diminuição da margem de erro na previsão e tempos e a conclusão dentro do prazo (MAXIMIANO, 2006, p. 27). Para isso, existem técnicas e programas de computador que podem automatizar certas etapas no projeto a fim de remover a responsabilidade e controle diretamente do desenvolvedor. Essas ferramentas são denominadas CASE, que significa *Computer-Aided Software Engineering* (Engenharia de Software Auxiliada por Computador). O termo compreende diversos programas que auxiliam o profissional desde a especificação até a manutenção do software. Por exemplo, existem ferramentas para análise de requisitos, modelagem do sistema, documentação etc. (SOMMERVILLE, 2007, p. 9). Como o principal objetivo delas é facilitar o trabalho, as interfaces de comunicação com o usuário são geralmente de fácil utilização, ou seja, normalmente “as ferramentas de tecnologia CASE possuem facilidades gráficas para o planejamento e projeto de sistemas” (REZENDE, 2005, p. 181).

Atualmente, existem inúmeras ferramentas CASE dedicadas aos mais diversos propósitos. Conforme Sommerville (2007, p. 9), certas “ferramentas CASE podem também incluir um gerador de código que gera automaticamente o código-fonte com base no modelo do sistema, e algumas orientações sobre o processo para os engenheiros de software”. Os diagramas lógicos produzidos durante a análise e modelagem do sistema são convertidos em códigos fonte na fase de implantação, explica Rezende (2005, p. 213).

2.6.1 Ferramentas CASE similares ao Triz

Existem inúmeras ferramentas que oferecem algum tipo de suporte ao Scrum. O Scrumwise (SCRUMWISE APS, 2009) e o JiraAgile (ATLASSIAN, 2016) oferecem recursos de controle de fluxo do trabalho Scrum, interação do time, controle de releases e integração com outras

ferramentas. Para pequenos times, o valor médio é de 9 dólares (cerca de 31 reais) mensais por usuário. A aplicação Triz propõe funcionalidades similares a essas, por outro lado, seu diferencial é buscar uma compatibilização organizacional entre métodos tradicionais e ágeis.

3 Materiais e Métodos

O método de desenvolvimento do software foi o Scrum, tendo em vista a importância das metodologias ágeis. O Triz consiste em uma ferramenta web com uso online, ou seja, criada com linguagens web e hospedada em um servidor para utilização via Internet. Para a modelagem, foram desenvolvidos diagramas de Casos de Uso e Modelo Lógico de Banco de Dados Relacional para facilitar a compreensão da proposta do projeto. A utilização e criação do banco de dados relacional também ocorreram de forma iterativa.

O sistema possui design responsivo; isso significa que sua programação é capaz de adaptar as páginas do aplicativo as diferentes resoluções dos dispositivos atuais. Abranger essa gama de dispositivos mostra-se uma importante vertente no mercado, visto que o acesso a Internet no Brasil cresce em ritmo acelerado, sendo que grande parte dos acessos parte de dispositivos móveis, como tablets e smartphones (BARRUCHO, 2015).

3.1 Escopo do Projeto Triz

O número de funcionalidades que modelam completamente a utilização de qualquer metodologia de desenvolvimento de software é muito grande. Por esse motivo, optou-se pela adoção de um escopo flexível, em que foram escolhidos os principais requisitos funcionais para modelagem e codificação. Dentre as principais funções, o aplicativo permite a criação de projetos, a definição de times de trabalho e a divisão de tarefas e responsabilidades entre eles.

3.2 Linguagens e Ferramentas

Durante o projeto foram utilizadas diversas linguagens, bibliotecas e frameworks para codificação do software. O HTML (*Hyper Text Markup Language*) compôs a estrutura básica das páginas, que, posteriormente, foram formatadas com folhas de estilo em CSS (*Cascading Style Sheets*), juntamente com o framework Bootstrap. As páginas HTML contêm notações da

biblioteca Twig, que é um mecanismo que facilita a renderização de páginas com dados vindos de modelos de dados. O PHP (*Hypertext Preprocessor*), que é uma poderosa linguagem de servidor, foi utilizado para todo o processamento e acesso ao banco de dados. E para facilitar algumas tarefas complexas, foi utilizado o microframework Silex que implementa vários recursos para a linguagem PHP. O SQL (*Structured Query Language*) atuou juntamente ao PHP para recuperar os registros no Sistema Gerenciador de Banco de Dados (SGBD) MySQL. A maioria dessas ferramentas são as melhores do mercado atual.

Dentre as ferramentas utilizadas, estão o editor básico de texto Notepad++, que possui recursos extremamente interessantes para codificação. Alguns navegadores de Internet (Google Chrome, Mozilla Firefox etc.) para visualização dos resultados da codificação. O ambiente web provido pelo XAMPP, extremamente necessário para possibilitar o ambiente de desenvolvimento na arquitetura cliente-servidor. O Composer foi escolhido como gerenciador de dependências utilizado, pois se despontou como uma das melhores ferramentas para esse fim atualmente. Também foi usado um programa para edição de imagens para ajustes, correções e tratamento de cores das imagens e ícones das interfaces gráficas do Triz. O Trello que foi usado para controle das tarefas que seriam executadas em cada Sprint. E o Astah Community e o MySQL Workbench para criação de diagramas e modelos do projeto.

3.3 Controle de Versão do Trabalho

O código-fonte e os demais artefatos sofreram controle de versão desde o início da codificação. O sistema de controle de versão utilizado foi o Subversion. Um dos maiores benefícios do versionamento é diferenciar as versões do código desenvolvidas durante o projeto. É mantida uma cópia de trabalho na máquina do programador e suas modificações são submetidas ao repositório SVN hospedado em um servidor. O repositório utilizado é provido pelo serviço Assembla que disponibiliza contas gratuitas com um (1) GB de espaço.

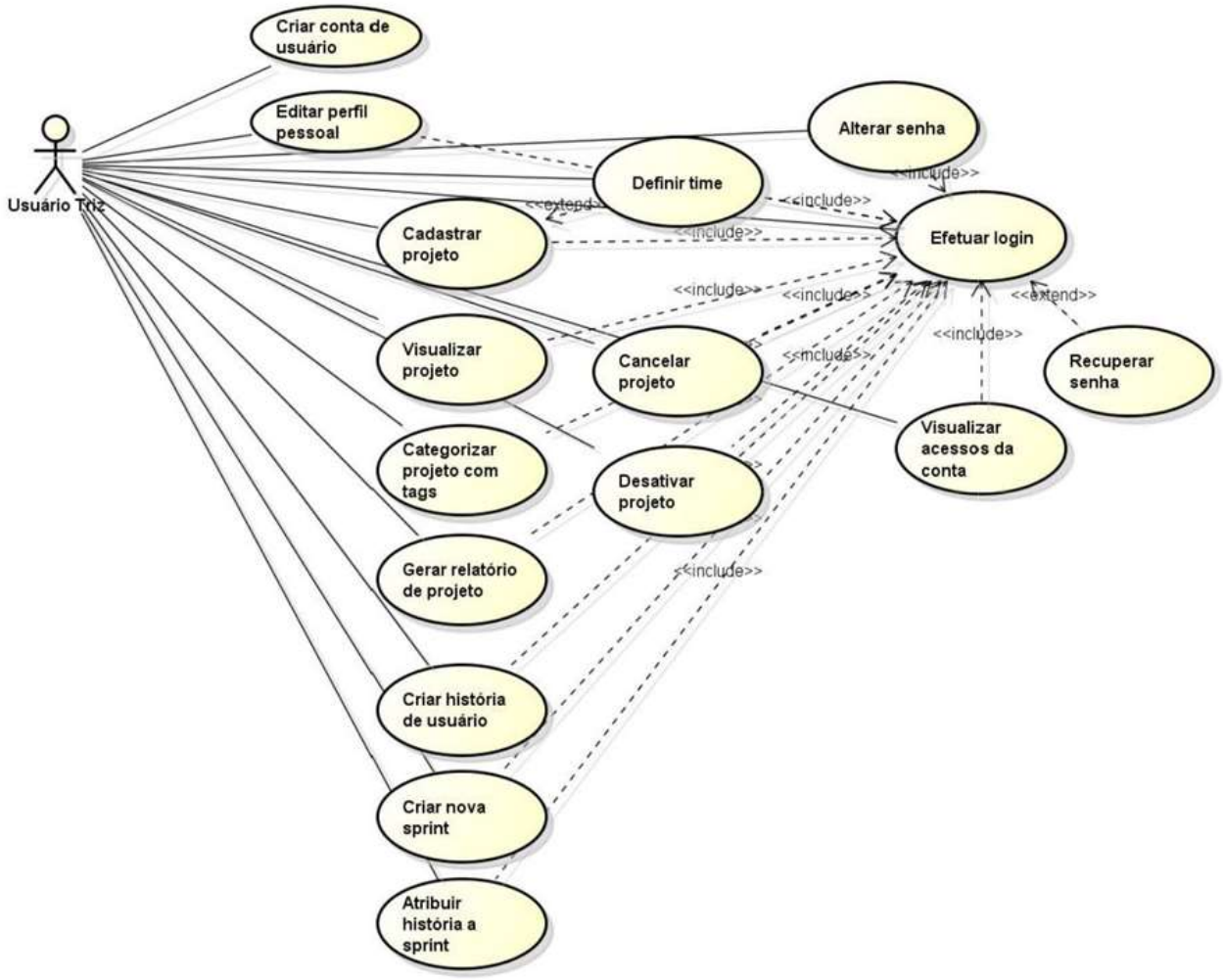
4. Triz

As primeiras iterações para construção do Triz tiveram foco nas principais funcionalidades relacionadas à gerência de projetos de software com uso do Scrum.

4.1 Diagrama de Casos de Uso

As principais funcionalidades do Triz estão representadas no diagrama de Casos de Usos apresentado na Figura 4, onde é possível observar as atividades que podem ser desempenhadas pelo usuário devidamente cadastrado e ativo no sistema.

Figura 4: Diagrama de casos de uso do Triz

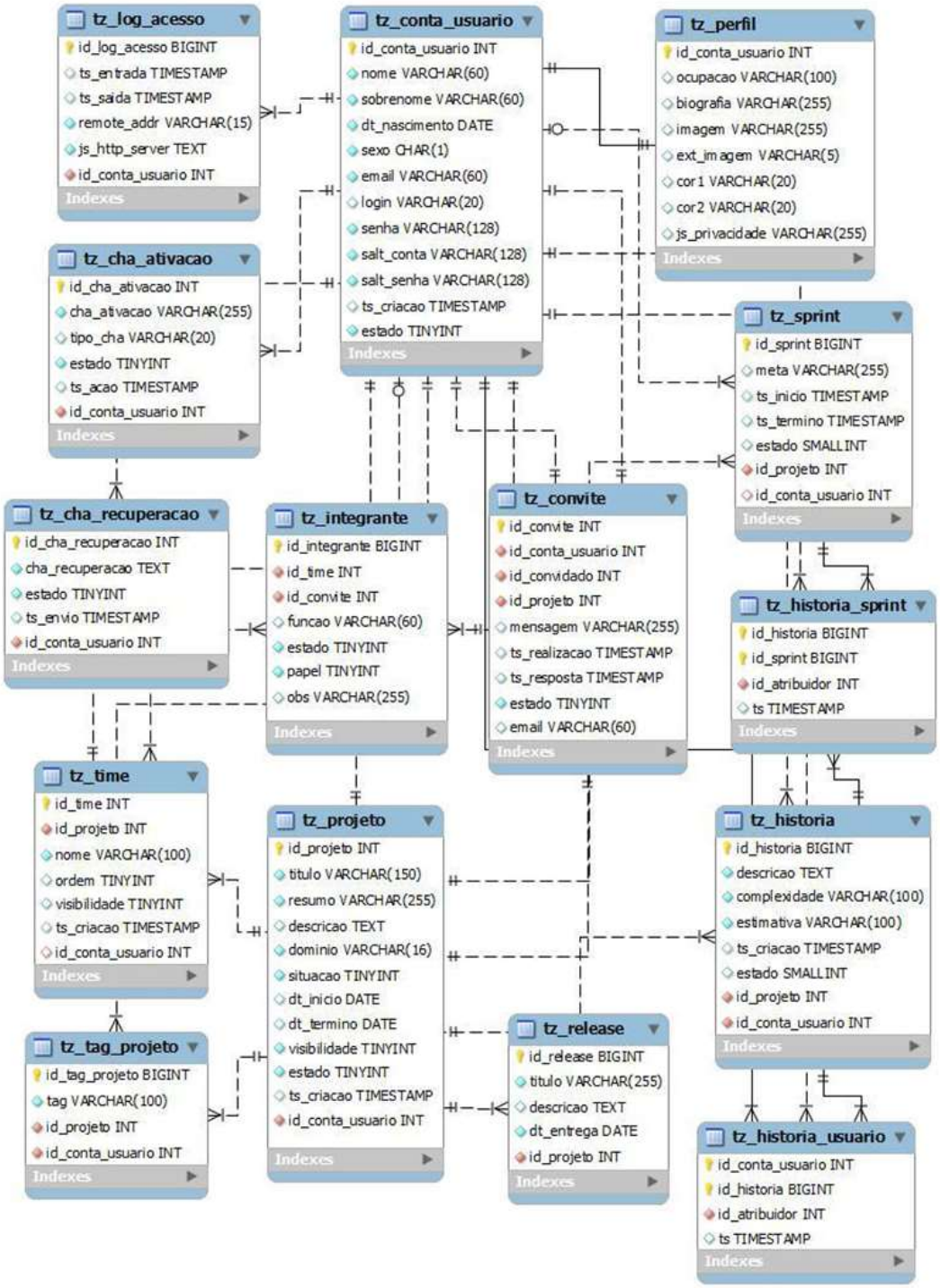


Fonte: Os autores (2016).

4.2 Modelagem do Banco de Dados Relacional

O modelo lógico apresentado na Figura 5 demonstra as tabelas com suas colunas e tipos de dados. Após a modelagem, a ferramenta MySQL Workbench permite a geração automática dos scripts SQL para criação do banco de dados com as restrições de integridade.

Figura 5: Modelo lógico do banco de dados do Triz



Fonte: Os autores (2016).

4.3 Arquitetura e Organização do Software

Os diretórios do projeto foram dispostos seguindo-se alguns conceitos como a necessidade do gerenciamento de dependências, o agrupamento das *views* (páginas HTML), armazenamento dos modelos e os controles, bem como o isolamento dos arquivos estáticos e de usuário. A estrutura principal é formada por quatro (4) diretórios: web, src, views e vendor, permitindo fácil manutenção e configuração do projeto. O Quadro 1descreve o conteúdo de cada um deles.

Quadro 1: Organização de Diretórios do Triz

Diretório	Descrição do Conteúdo
web	Contém os controles juntamente com os modelos de dados, faz a definição das rotas de navegação e o acesso ao banco de dados da aplicação
src	Disponibiliza códigos de funções úteis para o controle de conexão com o banco de dados, autenticação de usuários e envio de e-mail
views	Possui páginas em HTML usando as notações do mecanismo de renderização de templates Twig. Elas são invocadas pelos controles e recebem dados vindos dos modelos de dados
vendor	Armazena os códigos de terceiros, como os do microframework Silex e as bibliotecas utilizadas, como o Twig
static	Armazena recursos que são atualizados com pouca frequência e são importantes para composição das páginas de uma maneira geral. São folhas de estilo CSS, scripts em Javascript e imagens e ícones do sistema
ducs	Armazena imagens e arquivos postados pelos usuários, sendo o repositório de dados do Serviço de Conteúdo de Usuário

Fonte: Elaborado pelos autores (2016).

O conteúdo de vendor contém os códigos fonte de bibliotecas de terceiros e são manipulados pelo Gerenciador de Dependências Composer. Seu principal objetivo é retirar a responsabilidade dos desenvolvedores de instalar e atualizar bibliotecas. O Composer utiliza um arquivo chamado composer.json. Nele são declaradas as bibliotecas que o projeto depende e com comandos apropriados, os conteúdos são baixados e instalados. Além dos diretórios principais, existem o duc e o static que armazenam dados de usuários e dados estáticos.

4.4 Definição de Rotas para Navegação

As rotas de navegação são os endereços que podem ser acessados pelos usuários e que disponibilizam algum tipo de recurso, seja ele estático ou dinâmico. As rotas do Triz foram

determinadas com objetivo de manter coerente e intuitiva a navegação do usuário pelo sistema. Por exemplo, ao se acessar o domínio web da aplicação, utiliza-se automaticamente a rota raiz ("/") que aciona o controle raiz, que renderiza a página inicial. A rota /conta aciona o componente conta com várias outras rotas, como a criação de uma nova conta ou a visualização de acessos de uma conta. A autenticação é feita por Middlewares que são mediadores entre as ações solicitadas. As rotas que exigem que um usuário autentique-se verificam a presença de uma sessão de uso. Caso não exista, redirecionam para a tela de entrada.

Quadro 2: Exemplificação do Padrão de Rotas de Navegação do Triz

	Módulo controlador	Definição da Rota de navegação	Variáveis complementares	Nível de Segurança
Domínio Triz na Internet	Raiz	/		Livre
		/entrar		
	Conta	/conta/criar		
	Perfil	/perfil		Autenticação
		/perfil/usuário/{id}	Identificador único da conta de usuário	
	Projeto	/projeto/{url}/historia/{id}	URL do projeto e identificador	Autenticação e autorização

Fonte: Elaborado pelos autores (2016).

O processamento de rotas é realizado pelo microframework Silex, todas as requisições devem ser feitas ao arquivo web/index.php que é responsável por carregar as dependências do projeto em tempo de execução. Para restringir o fluxo de requisições a apenas ele, é necessário adicionar restrições em um arquivo htaccess do servidor Apache ou usar o servidor embutido do PHP apontando para tal arquivo. A segunda opção é recomendada apenas para ambientes de desenvolvimento, visto que é recurso apenas para testes e depuração.

4.5 Identidade Visual e Marca

A identidade visual consiste em empregar adequadamente os recursos visuais para representar a marca perante o público. O principal objetivo foi manter o sistema atrativo, jovem e vibrante, mas equilibrado. Foram escolhidas quatro cores principais para compor a paleta de cores do sistema. Sendo elas, tonalidades de vermelho, roxo, amarelo e verde. A

cor vermelha simboliza o desejo para criar e realizar projetos. A cor amarela desperta a criatividade. Foram utilizados degrades entre essas cores vibrantes e o branco para manter o equilíbrio, através de padrões de formas geométricas. O logotipo é a representação estilizada de letra T (a inicial de Triz). O nome Triz busca transmitir a ideia que até mesmo os projetos mais caóticos podem ser salvos por um triz. Foram realizados esforços para aplicar técnicas de SEO na codificação e organização do conteúdo. Algumas páginas, como para a visualização de um projeto foram marcadas com o protocolo Open Graph para melhorar a aparência dos links compartilhados em redes sociais.

4.6 Sistema de Atividade da Conta

O serviço Saticon foi implementado para gerenciar o estado das contas de usuário. Seu objetivo é prover a comunicação entre o Triz e o usuário para ativar a conta, mantê-la ativa, cancelar a conta em caso de uso contrário à política de uso do Triz entre outras coisas. Atualmente, seu uso consiste na ativação de conta de usuário.

Para garantir a fidedignidade e propriedade do e-mail informado na criação de uma nova conta, o usuário é obrigado a usar um link com uma chave única enviada para o e-mail informado. Quando o usuário cadastra-se é gerada uma chave única, que consiste em um HASH 512 do salt_conta concatenado ao horário atual. Essa chave é armazenada com o status 0 (zero) que significa "conta criada e não ativada". Ao mesmo tempo, é enviado para o e-mail cadastrado um link composto com a chave. Se o usuário obtiver acesso a esse link, significa que ele realmente tem acesso a conta de e-mail. Ao acessar o link, as comparações são efetuadas, se com sucesso, muda-se o status para 1 (um), "conta criada e ativada por e-mail". Registrando também o momento da ativação e redirecionando o usuário para a tela de login. O salt_conta consiste em um HASH único para cada usuário, baseado no e-mail (que é único no sistema) com o horário de criação da conta. Esse salt é usado quando propício.

4.7 Sistema de Acesso com Hashs

Para um usuário acessar o Triz, ele precisa ter uma conta de usuário devidamente cadastrada e ativa no sistema. A conta tem estados que representam a aptidão de uso. Cada vez que o usuário ingressa no sistema, são armazenados dados como o horário e o IP da máquina de origem, no sistema. Para isso ele precisa informar o e-mail e a senha. As senhas

no triz são armazenadas juntamente com um salt aleatório com o objetivo de aumentar a segurança delas em caso de invasão do banco dados.

4.8 Interfaces Gráficas do Triz

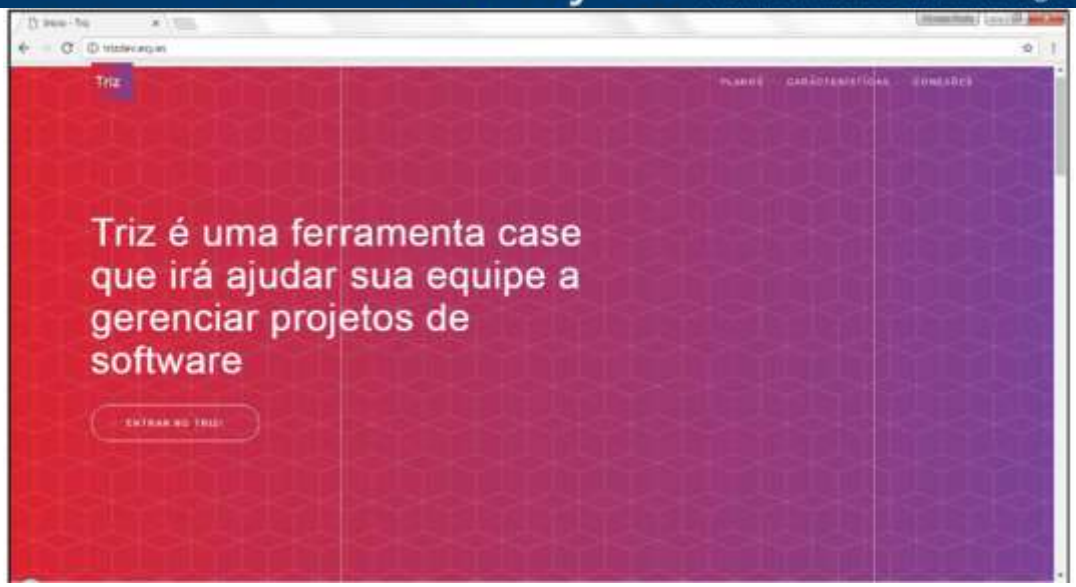
O processo de criação das interfaces gráficas iniciou-se com o desenvolvimento de protótipos. A Figura 6 apresenta o protótipo da aplicação, (a) página de login, (b) criação de novo projeto e (c) criação de equipe. O design gráfico foi projetado seguindo um padrão minimalista. Dentre as divers as funções, é possível a criação de projetos de forma dinâmica, a definição de equipes de trabalho e a divisão de tarefas e responsabilidades entre elas.

Figura 6: Protótipos da interface gráfica do aplicativo Triz

Fonte: Os autores (2016).

Após a criação de alguns protótipos, iniciou-se a codificação das páginas finais. Ao ingressar no sistema, o usuário é redirecionado para a página inicial apresentada na Figura 7.

Figura 7: Página inicial do Triz



Fonte: Os autores (2016).

O Triz está disponível para qualquer usuário acima de 18 anos de idade e que possua uma conta de e-mail ativa. Ao entrar na página de criação de nova conta, ele precisa informar dados básicos que o caracterizem perante o sistema. Todos os dados apresentados na tela são obrigatórios e apresentam a configuração mínima daquilo que o sistema precisa controlar. O nome completo é um dado público. A data de nascimento, gênero e e-mail são mantidos em sigilo. O e-mail será futuramente substituído por um nome de usuário. É possível observar a página de criação de conta na Figura 8.

Figura 8: Página de criação de nova conta Triz

Fonte: Os autores (2016).

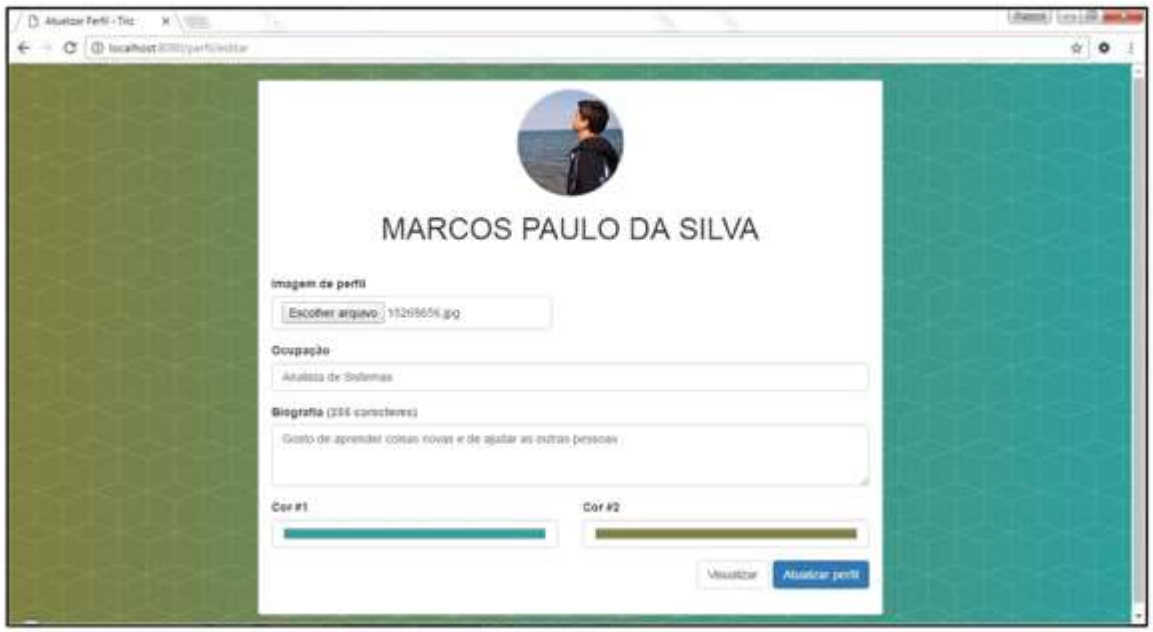
O aspecto social do sistema é abrangido por um sistema de perfis e cartões de visita virtuais. O cartão possui nome, foto e algumas informações adicionais, conforme Figura 9.

Figura 9: Exemplo de cartão de visita virtual de usuário

Fonte: Os autores (2016).

Todo usuário, quando criado, possui um perfil vazio. Ele pode atualizar seu perfil para conferir mais dados sobre sua identidade perante os outros usuários. Ele pode informar ocupação, imagem de perfil e sua biografia. A Figura 10 mostra a página de edição de perfil.

Figura 10: Exemplo de página de edição de perfil de usuário



Fonte: Os autores (2016).

Os membros do projeto são definidos através da seção chamada Membros. Nela é possível convidar usuários para participarem dos projetos através do e-mail associado à conta. Os convites enviados podem ser visualizados e gerenciados pelo administrador do projeto. E os papéis do Scrum podem ser atribuídos aos usuários que pertencem ao time de desenvolvimento. A Figura 11 apresenta a página de definição de membros.

Figura 11: Exemplo de página de definição de membros de um projeto

Fonte: Os autores (2016).

Na seção Backlog é possível criar histórias de usuário e gerenciar as já existentes. Cada uma delas possui os campos de complexidade e estimativa, ambos são valores relativos que permitem o time se organizar. A Figura 12 apresenta o backlog de um projeto.

Figura 12: Exemplo de Página de listagem de histórias do Backlog de um projeto

Fonte: Os autores (2016).

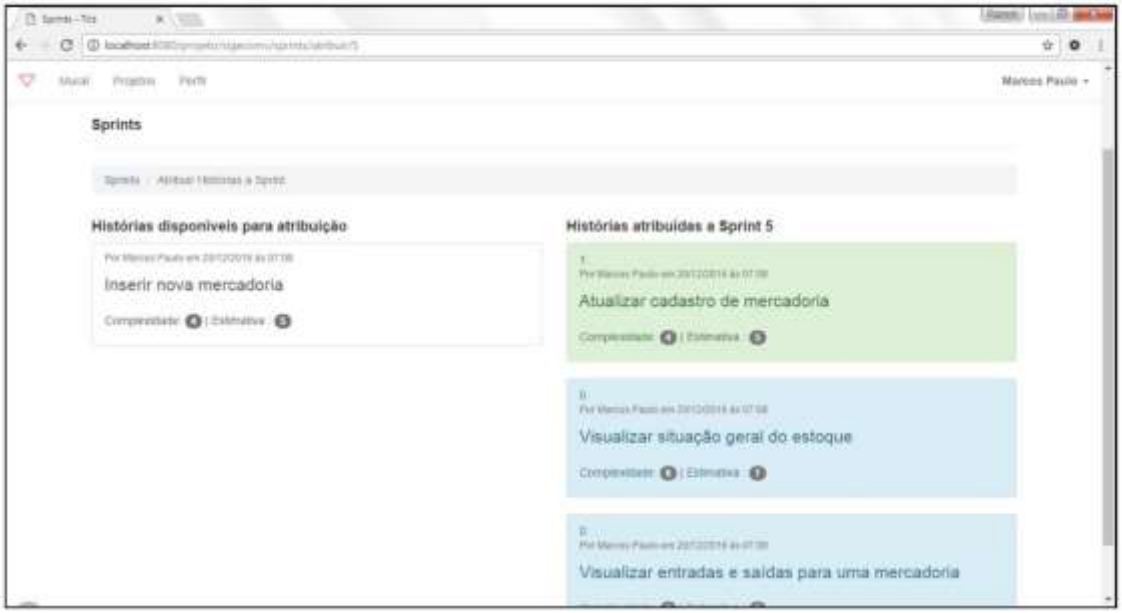
A seção de Sprints exibe a listagem de ciclos de trabalho realizados durante o projeto. O topo da seção possui a indicação da sprint vigente, conforme apresenta a Figura 13.

Figura 13: Exemplo de página de exibição de sprints

Fonte: Os autores (2016).

Ao ingressar na sprint vigente é possível atribuir histórias ao Backlog da Sprint. De um lado são exibidas as histórias disponíveis e do outro as histórias da Sprint em andamento, em azul, e as histórias concluídas, em verde, conforme mostra a Figura 14.

Figura 14: Exemplo de página de atribuição de histórias a Sprint



Fonte: Os autores (2016).

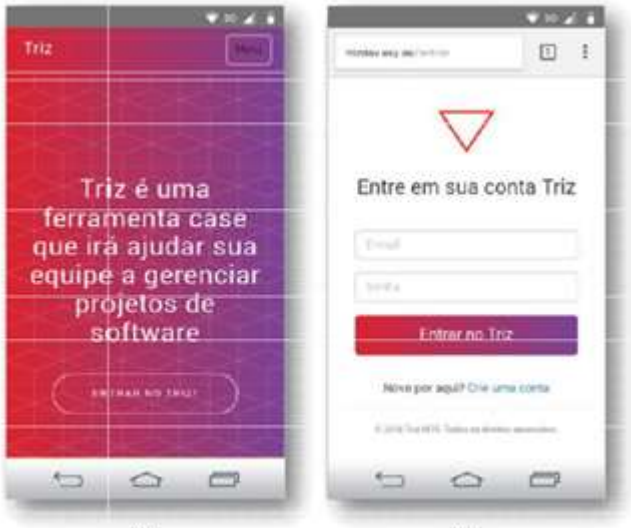
Os estados das histórias podem ser gerenciados acessando-se cada uma delas, além disso, é possível atribuir membros responsáveis por elas, como apresenta a Figura 15.

Figura 15: Exemplo de página de visualização de uma história de usuário

Fonte: Os autores (2016).

Algumas interfaces, contaram com maior atenção quanto a responsividade. A página de entrada no sistema deve fornecer o máximo de facilidade para que o usuário ingresse no sistema, diminuindo a taxa de evasão. A Figura 11 exibe páginas em um smarphone.

Figura 16: Páginas visualizadas em smartphone: (a) pág. inicial Triz, (b) pág. de login



Fonte: Os autores (2016).

4.9 Envio de E-mails de Notificação Formatados

A conta de e-mail, atualmente, é o meio mais aceito e utilizado para se criar vínculo do usuário com serviços e sistemas online. É com ele que é possível executar tarefas de notificação e validações de um outro sistema. O Triz utiliza uma conta de e-mail do usuário para a criação desse vínculo, permitindo o envio de notificações e validação. A Figura 12 apresenta a aparência de um e-mail de ativação de conta recebido por um usuário.

Figura 17: Visualização do e-mail de confirmação de conta de um usuário

Fonte: Os autores (2016).

Os e-mails do Triz podem ser enviados de um servidor operacional que suporte SMTP (protocolo de envio de e-mails). Os e-mails foram criados em HTML e refletem a identidade visual do Triz. A formatação aplicada permite a visualização legível em vários dispositivos.

4.10 Implantação e Hospedagem do Sistema

Para implantação do sistema, é necessário hospedar o código fonte da aplicação em um servidor WEB e instalar as dependências listadas em composer.json. As variáveis

globais ducs e statis devem ser configuradas para os destinos corretos. O ducs necessita de acesso a escrita e leitura para gravação dos dados do usuário.

4.11 Testes e Validação

Com o desenvolvimento do trabalho, houve a preocupação de manter as páginas HTML devidamente organizadas conforme as "melhores práticas" de codificação. A W3C (*World Wide Web Consortium*), que é principal organização para padronização da rede mundial de computadores, oferece serviços de validação de marcação de documentos WEB, como HTML, XHTML etc. São verificadas questões como o uso correto das tags e seus atributos. Foi realizado o uso desses validadores, sendo possível encontrar defeitos ocasionados por desatenção, entre outros. Também foram realizados testes de conexão e gravação no banco de dados de maneira manual. Além da verificação do sistema em um ambiente operacional. Futuramente, deseja-se realizar testes automatizados.

5 Discussão dos Resultados

Com este trabalho, concebeu-se uma ferramenta CASE denominada Triz, que busca auxiliar efetivamente a pequenas e médias equipes alcançarem melhores rendimentos em seus projetos, através de um ambiente intuitivo e agradável para o gerenciamento de tarefas e responsabilidades. E, consequentemente, alcançar projetos e produtos com mais qualidade.

Espera-se com isso, fornecer um meio para a diminuição do índice de fracasso de projetos e aumento da qualidade de softwares produzidos. Além de familiarizar pequenas e médias equipes com a sistematização de tarefas e responsabilidades, bem como com outros conceitos de gerência e Engenharia de Software.

6. CONSIDERAÇÕES FINAIS

A área de gerência de projetos mostra-se com grandiosa importância, afinal, são em projetos que os produtos e serviços nascem há algum tempo. A criação e adoção de ferramentas e metodologias que reflitam as novas maneiras das pessoas trabalharem são esforços que devem ser empreendidos, a fim de trazer maiores índices de sucesso. A pesquisa e desenvolvimento do Triz trouxeram bons resultados e uma maior consciência

sobre a necessidade de controle, planejamento e organização de qualquer tipo de projeto, acadêmico ou profissional. A aplicação Triz foi modelada tendo em vista as funcionalidades que mais agregariam valor perante os usuários. Sendo que suas características de usabilidade são diferenciais competitivos em relação às ferramentas similares. O Triz trata-se de uma boa opção para a gerência de projetos para equipes que desejem aplicar uma ferramenta gerencial do tipo. A realização deste trabalho trouxe aprendizado nas áreas pesquisadas, bem como no planejamento e execução de um projeto por meio de uma metodologia ágil. A experiência de trabalhar com as tecnologias, ferramentas e métodos aqui aplicados foi enriquecedora. Pretende-se, futuramente, ampliar o escopo do aplicativo a fim de atender a mais áreas do desenvolvimento do software, tornando a ferramenta mais robusta, como, por exemplo, criar uma nova versão que permita trabalho offline com maior número de funcionalidades.

REFERÊNCIAS

ATLASSIAN. Features for software development: Jira software is the project management tool for agile teams. Atlassian, Sydney, [2016]. Disponível em: <https://www.atlassian.com/software/jira/agile>. Acesso em: 9 nov. 2016.

BARRUCHO, L. G. IBGE: Metade dos brasileiros estão conectados à internet; Norte lidera em acesso por celular. 2015. Disponível em: <http://www.bbc.com/portuguese/noticias/2015/04/150429_divulgacao_pnad_ibge_lgb>. Acesso em: 12 out. 2015.

BRASIL. Ministério da Ciência Tecnologia e Inovação. Secretaria de Política de Informática. A história da Tahini-Tahini: melhoria de processos de software com métodos ágeis e modelo MPS. 9. ed. Brasília: [s.n.], 2013. Disponível em: <http://www.mct.gov.br/upd_blob/0228/228094.pdf>. Acesso em: 5 set. 2016.

_____. Ministério da Ciência e Tecnologia. Secretaria de Política de Informática. Pesquisa de qualidade no setor de software brasileiro 2009. [S.l.: s.n., 2010]. Disponível em: <http://www.mct.gov.br/upd_blob/0214/214567.pdf>. Acesso em: 10 out. 2016.

CARVALHO, B. V. de; MELLO, C. H. P. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. Gest. Prod., São Carlos, v. 19, n. 3, p. 557-573, 2012. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2012000300009&lng=en&nrm=iso>. Acesso em: 18 out. 2016.

COCKBURN, A. et al. Manifesto para o desenvolvimento ágil de software. 2001. Disponível em: <http://www.manifestoagil.com.br/>. Acesso em: 21 nov. 2016.

FERNANDES, A. A.; TEIXEIRA, D. de S. Fábrica de software. São Paulo: Atlas, 2004.

GOMES, A. F. Agile: desenvolvimento de software com entregas frequentes e foco no valor de negócio. São Paulo: Casa do Código, 2013.

MAXIMIANO, A. C. A. Administração de projetos: como transformar idéias em resultados. 2. ed. São Paulo: Atlas, 2006.

PHILLIPS, J. Gerência de projetos de tecnologia da informação. 6. ed. Rio de Janeiro: Elsevier, 2003.

PRESSMAN, R. S. Engenharia de software. 6.ed. Porto Alegre: Bookman, 2006.

REZENDE, D. A. Engenharia de software e sistemas de informação. 3.ed. Rio de Janeiro: Brasport, 2005.

SABBAGH, R. Scrum: gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013.

SCRUMWISE APS. The most intuitive scrum tool you've ever tried. Scrumwise Better Scrum, Frederiksberg, Denmark, 2009. Disponível em: https://www.scrumwise.com/. Acesso em: 9 nov. 2016.

SEVERINO, A. J. Metodologia do trabalho científico. 23. ed. rev. e atual. São Paulo: Cortez Editora, 2007.

SOMMERVILLE, I. Engenharia de software. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

SUTHERLAND, J. Scrum: a arte de fazer o dobro na metade do tempo. São Paulo: LeYa, 2014.