Faculdade de Tecnologia de Ourinhos

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LILIAN MADILENE MILAN PERIANEZ
MARIANA DE JESUS DELTREJO
ROBERTA PFAFF MALUF TEIXEIRA

VALIDAÇÃO DO SOFTWARE EASY - SCRUM

OURINHOS–SP

junho de 2019

LILIAN MADILENE MILAN PERIANEZ
MARIANA DE JESUS DELTREJO
ROBERTA PFAFF MALUF TEIXEIRA

VALIDAÇÃO DO SOFTWARE EASY - SCRUM

Trabalho de Conclusão de Curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas da FATEC Ourinhos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Profº Vera Lúcia Silva Camargo

OURINHOS–SP

junho de 2019

LILIAN MADILENE MILAN PERIANEZ
MARIANA DE JESUS DELTREJO
ROBERTA PFAFF MALUF TEIXEIRA

VALIDAÇÃO DO SOFTWARE EASY - SCRUM

Trabalho de Conclusão de Curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas da FATEC Ourinhos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

BANCA EXAMINADORA

Profº Vera Lúcia Silva Camargo
FATEC Ourinhos
Orientador

Prof. Dr. Segundo Membro da Banca
FATEC Ourinhos do Segundo Membro da Banca

Prof. Dr. Terceiro Membro da Banca
FATEC Ourinhos do Terceiro Membro da Banca

Ourinhos–SP, de junho de 2019

Dedico este trabalho primeiramente a Deus. A minha família e amigos, em especial minha mãe e meu padrasto: Rita Milan e Maurício Temporin pelo apoio incentivo e pela capacidade de acreditar em mim, assim como Fabio Augusto que me acompanhou nessa jornada e aos meus filhos: Carlos Eduardo e Lucas Henrique, que dedico a minha vida. - ***Lilian Madilene Milan Perianez***

Dedico primeiramente a Deus, por ter me dado a sabedoria necessária para seguir em frente, a minha mãe Teresinha, e meu namorado Renan, pelo apoio e carinho que sempre tiveram comigo, me auxiliando para que eu chegasse até esta etapa de minha vida. - ***Mariana de Jesus Deltrejo***

Gostaria de dedicar esse trabalho a Deus, que nunca me abandonou nos momentos difíceis. A minha família que sempre me deu apoio e ajuda, e a professora Vera Lucia Silva Camargo, que foi compreensiva nos momentos de dificuldade. - ***Roberta Pfaff***

AGRADECIMENTOS

A minha orientadora Vera Lúcia Silva Camargo por todo apoio e paciência ao longo da elaboração do nosso projeto final. Também gostaríamos de deixar um agradecimento especial a instituição Fatec por possibilitar a execução deste trabalho acadêmico.

"Mas, buscai primeiro o reino de Deus,
e a sua justiça, e todas estas coisas vos serão
acrescentadas.
(Bíblia Sagrada, Mateus, 6, 33)

PERIANEZ,L,M,M; DELTREJO,M,J; TEIXEIRA,R,P,M;. **Validação do Software Easy - Scrum**. 50 p. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – FATEC Ourinhos, Ourinhos–SP, junho de 2019.

RESUMO

O objetivo deste projeto é analisar e avaliar se as funcionalidades, desempenho e usabilidade do Easy Scrum estão adequados a uma ferramenta didática. Easy Scrum, como todo software, deve atender funcionalidades previstas e não apresentar erros, portanto testes de funcionalidade e usabilidade foram aplicados. Para execução foi utilizado teste de usabilidade prático com alunos do 6º semestre do curso de Análise e Desenvolvimento de Sistemas, teste funcional utilizando o método Caixa-Preta, o software "Easy Scrum", NetBeans IDE 8.2, XAMPP, HeidiSQL e Latex. Ao longo dos testes foi possível detectar inúmeros erros tanto de sintaxe como de programação, falhas no banco de dados, entre outros.

Palavras-chave: Análise de Sistema, Educação,Easy Scrum, Teste de Usabilidade, Teste de Funcionalidade, Kanban, Scrum

PERIANEZ,L,M,M; DELTREJO,M,J; TEIXEIRA,R,P,M;. **Easy-Scrum Software Validation**. 50 p. Final Project (Technology in Analisys and Development Systems) – FATEC Ourinhos, Ourinhos–SP, junho de 2019.

ABSTRACT

The goal of this project is to analyze and evaluate if the functionality, performance and usability of Easy Scrum are appropriate to a didactic tool. Easy Scrum, like all software, must meet expected features and present no errors, so functionality and usability tests have been applied. For execution, a practical usability test was used with students from the 6th semester of the Systems Analysis and Development course, functional test using the Caixa-Preta method, Easy Scrum software, NetBeans IDE 8.2, XAMPP, HeidiSQL and Latex. Throughout the tests it was possible to detect innumerable errors of syntax as well as of programming, failures in the database, among others. **TRADUÇÃO LITERAL**

Keywords: System analysis, Education, Easy Scrum, Usability Testing, Functionality Test, Kanban, Scrum

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Métodos ágeis, conjunto de práticas para organizar e facilitar o desenvolvimento de software, a razão pela qual surgiu foi fazer frente aos modelos tradicionais de desenvolvimento, considerados lentos e burocráticos, com o objetivo de reduzir o ciclo de desenvolvimento. Dentre os diversos métodos destaca-se *Scrum* (JUNIOR, 2017).

Scrum(subs): Um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. O *Scrum* é leve, simples de entender e extremamente difícil de dominar (JUNIOR, 2017).

Diversas ferramentas têm sido criadas para ajudar na aplicação do *Scrum*, uma delas é o *ScrumHalf*, este que oferece uma série de facilidades para a definição de requisitos e tarefas a serem executadas pela equipe, para o planejamento de *releases* do produto, para o acompanhamento dos trabalhos e para o aperfeiçoamento do processo sendo utilizado (JUNIOR, 2017).

Daí a necessidade de aperfeiçoar um projeto já existente na Faculdade, para que os alunos concluam o curso da Fatec Ourinhos com conhecimento prévio sobre *Scrum* com a utilização dessa ferramenta. E com isto estarão preparados para o mercado de trabalho após a conclusão da faculdade.

1.1 Problema

Com base nos estudos feitos durante a faculdade de Análise e Desenvolvimento de Sistemas percebemos que existe uma necessidade muito grande de ensinar aos profissionais de *TI*, que estão ou desejam ingressar no mercado de trabalho, o uso do *Scrum*, uma técnica extremamente útil no desenvolvimento de novos projetos, seja de *software* ou não.

Sendo assim, percebemos que o software *"Easy Scrum"* desenvolvido por Noda e Horacio (2018), precisa ser avaliado quanto à funcionalidades e usabilidade para que possa ser disponibilizado a estudantes e docentes.É possível encontrar instruções de instalação e execução do "Easy Scrum" ao final desse trabalho, nos apêndices A e B.

1.2 Objetivo

Analisar e avaliar se as funcionalidades, desempenho e usabilidade do *Easy Scrum* estão adequados a uma ferramenta didática.

Easy Scrum, como todo *software*, deve atender funcionalidades previstas e não

apresentar erros, portanto testes de funcionalidade e usabilidade serão aplicados e eventuais erros relatados.

1.3 Justificativa

Scrum é uma técnica, um método, extremamente útil no desenvolvimento de novos projetos, seja de software ou não. Sendo assim, observou-se a necessidade do desenvolvimento de uma ferramenta didática sobre o mesmo, com o intuito de ensiná-lo aos profissionais de *TI* que estão ou desejam ingressar no mercado de trabalho.

Porque validar?

Easy Scrum deve ter facilidade e clareza para ser compreendido e manipulado pelos estudantes de acordo com os critérios estabelecidos em estudos realizados por Nielsen (1994), responsável pelo conjunto de heurísticas para avaliação de usabilidade de interface de dispositivos computacionais, base para outras contribuições no âmbito de design de interface.

Concluindo essa introdução, apresenta-se como esse trabalho está organizado: no capítulo 2 se discute o referencial teórico sobre usabilidade e testes funcionais de *software*, bem como, os conceitos sobre práticas ágeis, em particular o *Scrum*; no capítulo 3 apresentam-se descrição do *EasyScrum*, os métodos, instrumento e ferramentas de como o trabalho de validação foi desenvolvido; no capítulo 4 os resultados e no capítulo 5, as conclusões.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo aborda assuntos relacionados a avaliações de usabilidade e testes de funcionalidades de software e às práticas ágeis: Manifesto ágil, *Kanban*, *Scrum* e *Scrumban*.

2.1 Usabilidade de Software – Testes e Validação

O teste de usabilidade é um processo no qual os desenvolvedores avaliam em que situação um produto se encontra em relação a critérios específicos de usabilidade.

O desenvolvimento de *Software* vem evoluindo, porém ainda persistem problemas na construção de *softwares* de qualidade que obedeçam a prazos e orçamentos planejados. Buscando resolver este problema, surgiu a Engenharia de *Software* que se destina a prover uma estrutura para a construção de *Software* com alta qualidade (PRESSMAN; MAXIM, 2016).

Com o intuito de criar um modelo de avaliação da qualidade do *software*, foi criada a (ISO/IEC 9126, 2003), de modo a buscar alcançar uma qualidade que atinja as necessidades de seus usuários. Ela é composta por seis características: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

De acordo com Ferreira (2002)

> A validação tem como objetivo verificar como o produto se enquadra em relação a padrões de usabilidade, padrões de performance e padrões históricos. Esses padrões são originados dos objetivos de usabilidade definidos no começo do projeto através de inspeções de mercado, entrevistas com usuários ou simplesmente suposições da equipe que está desenvolvendo.

Valida também a interação entre os componentes do produto, como por exemplo, a forma em que a documentação se apresenta, a ajuda, se o *software* e o *hardware* estão integrados uns com os outros. Outro objetivo é prever o lançamento de um produto novo no mercado que possivelmente necessite logo de manutenção. (BARROS, 2011)

2.2 As Práticas Ágeis

Na atualidade, agilidade se tornou a palavra da moda quando é escrito um processo de *software*. Uma equipe ágil é aquela que é resistente a qualquer mudança ocorrida, mudanças na equipe, mudanças no *software*, mudanças que poderão ter um impacto no sistema que está sendo construído. Uma equipe ágil precisa reconhecer que o sistema

será desenvolvido em equipes, e que as habilidades de todos juntos irão servir para obter sucesso no projeto (PRESSMAN, 2011).

Os Métodos Ágeis, ou Práticas Ágeis, são uma alternativa a gestão convencional de projetos, foram criados inicialmente para gerenciar desenvolvimento de *softwares*, mas, atualmente podem ser utilizados em qualquer tipo de projeto.

Os Métodos Ágeis pretendem estimular um gerenciamento de projetos que busca a inspeção e adaptação frequente, incentivando o trabalho em equipe e a organização. São um grupo de práticas eficazes com o intuito de permitir entrega rápida de projetos, e com alta qualidade, alinhando o desenvolvimento do projeto com a necessidade e vontade do cliente. (JUNIOR, 2017)

2.2.1 Manifesto Ágil

O Manifesto Ágil, nada mais é do que uma declaração de valores e princípios primordiais para o desenvolvimento de *software* (BECK et al., 2001).

Criado em fevereiro de 2001 por 17 profissionais que já praticavam métodos ágeis.

O Manifesto Ágil trata valores que todos os profissionais que participaram da reunião concordaram em seguir e disseminar. Os autores do Manifesto Ágil são:

Kent Beck o criador do XP, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham desenvolvedor do primeiro wiki, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries um dos criadores do XP, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber co-criador do Scrum e Head da Scrum.org, Jeff Sutherland co-criador do Scrum e CEO da Scrum Inc e Dave Thomas (FOWLER; HIGHSMITH, 2001).

Segundo Beck et al. (2001), os valores do Manifesto Ágil são:

Indivíduos e interações mais que processos e ferramentas;

Software em funcionamento mais que documentação abrangente;

Sendo assim, entende-se que há valor nos itens à direita, porém, mais ainda nos itens à esquerda.

O Manifesto Ágil também possui doze princípios que devem ser seguidos, de acordo com Beck et al. (2001), são eles:

> Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor;
> Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas;
> Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;

> Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;
>
> Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho;
>
> O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara;
>
> Software funcional é a medida primária de progresso;
>
> Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes;
>
> Contínua atenção à excelência técnica e bom design, aumenta a agilidade;
>
> Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito;
>
> As melhores arquiteturas, requisitos e designs emergem de times auto organizáveis;
>
> Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Concluindo, o Manifesto Ágil é, sem dúvidas, um marco muito significativo na indústria de desenvolvimento de *software*. (BRASILEIRO, 2018).

2.2.2 *K*anban

Kanban é uma palavra de origem japonesa que significa literalmente **"cartão"** ou **"sinalização"**. Este é um conceito relacionado com a utilização de cartões (*post-it* e outros) para indicar o andamento dos fluxos de produção (SIGNIFICADOS, 2015).

Nesses cartões são colocadas indicações sobre uma determinada tarefa, por exemplo, "para executar", "em andamento" ou "finalizado". A utilização de um sistema *Kanban* permite um controle detalhado de produção com informações sobre quando, quanto e o que produzir (SIGNIFICADOS, 2015).

O método *Kanban* foi inicialmente aplicado em empresas japonesas de fabricação em série e está muito ligado ao conceito de *"just in time"*. Visa evitar falta ou excesso de produção, expor problemas e definir diretamente o que deve ser feito (SIGNIFICADOS, 2015).

A empresa japonesa de automóveis *Toyota* foi a quem incorporou esse método devido a necessidade de manter um eficaz funcionamento do sistema de produção em série (SIGNIFICADOS, 2015).

O *Kanban* eletrônico é utilizado substituindo o método físico, evitando certos problemas, como por exemplo a perda de cartões. Atualmente, o *Kanban* é muitas vezes utilizado em conjunto com o *Scrum*, pois são duas metodologias muito utilizadas no desenvolvimento ágil de *software* (SIGNIFICADOS, 2015).

2.2.3 *S*crum

Scrum é uma ferramenta que facilita a forma de se trabalhar, organizando as tarefas que cada membro da equipe irá desempenhar para alcançar um objetivo. Ele não apenas serve para alcançar um objetivo, também fazer com que os membros da equipe respeitem os prazos estabelecidos. De acordo com Schwaber e Sutherland (2013):

> Times *Scrum* são projetados para otimizar flexibilidade e produtividade. Para esse fim, eles são auto-organizáveis, interdisciplinares e trabalham em iterações. Cada Time *Scrum* possui três papéis: 1) o *ScrumMaster*, que é responsável por garantir que o processo seja entendido e seguido; 2) o *Product Owner*, que é responsável por maximizar o valor do trabalho que o *Scrum* faz; e 3) o Time, que executa o trabalho propriamente dito (SCHWABER; SUTHERLAND, 2013)

No *Scrum* se trabalha com o *product backlog*, uma lista que contêm todas as funcionalidades do produto a ser trabalhado, facilitando seu desenvolvimento (SIGNIFICADOS, 2015).

O time *Scrum*:

O time *Scrum* é composto pelos seguintes membros: *Product Owner*, o Time de Desenvolvimento e o *Scrum Master* (SCHWABER; SUTHERLAND, 2013).

Product Owner:

O *Product Owner*, ou dono do produto, é a pessoa responsável por maximizar o valor do produto e do trabalho do Time de Desenvolvimento, manter toda comunicação com a equipe, definindo quais condições vão fazer parte do *product backlog* (SCHWABER; SUTHERLAND, 2013).

Time de Desenvolvimento:

Profissionais responsáveis por entrar uma versão usável que potencialmente incrementa o produto "Pronto" ao final de cada *Sprint*. São os únicos que criam incrementos. O tamanho adequado para o Time de desenvolvimento é pequeno o suficiente para se manter ágil, e grande o suficiente para completar o trabalho dentro dos limites da *Sprint*, entre três e nove pessoas (SCHWABER; SUTHERLAND, 2013).

Scrum Master:

O *ScrumMaster* é um servo-líder para o time *Scrum*. Tem a responsabilidade de garantir que o *Scrum* seja entendido e aplicado, fazer o monitoramento de todos da equipe, e ter certeza que cada membro adere a teoria, práticas e regras do *Scrum* (SCHWABER; SUTHERLAND, 2013).

2.2.4 *S*crumBan

KanBan e *Scrum* são metodologias ágeis, quando junta a natureza normativa do *Scrum* e a melhoria de processo do *KanBan*, as equipes conseguem chegar a um desenvolvimento ágil, e melhorar frequentemente seus processos. (SCRUM, 2014)

Os dois juntos conseguem melhorar o desenvolvimento da equipe, acabando o estresse das despesas, e melhorando a satisfação do cliente em geral. O que mais chama atenção é a entrega do produto com uma alta qualidade. (SCRUMBAN, 2018).

Princípios básicos do *ScrumBan:*

- Procurar melhorias para atingir um processo que tenha sucesso;
- Estabelecer as reuniões, quadros e funções que vão ser feitos naquele momento.;
- Respeitar todas as funções atuais durante o tempo que tentam fazer melhorias.;

2.3 Usabilidade de Software

Conceito de usabilidade é uma medida na qual um produto possa ser usado por um usuário, para alcançar objetivos como eficácia, eficiência e satisfação. (GONÇALVES, 2009).

Análise Heurística é uma avaliação fácil de aplicar ao projeto. Um processo flexível em que se pode adaptar alguns pontos específicos do projeto e identificar eventuais erros que comprometam a usabilidade (TEIXEIRA, 2018). Geralmente a análise heurística é processada na seguinte ordem:

- Entender os usuários:

Através de pesquisas que serão elaboradas pelas autoras desse trabalho de “Validação do Software Easy Scrum”.

- Definir as heurísticas de usabilidade:

Segundo Bertini e Kimani (2006), o conjunto de heurísticas elaborado por Nielsen (1994) é utilizado como forma de precaver erros de usabilidade e satisfazer requisitos de qualidade de interfaces em vários domínios de aplicação.

- Avaliar a experiência:

Através de analises dos resultados dos alunos que participarem da pesquisa.

- Reportar os resultados:

Após o final da pesquisa serão apontados e estudados os resultados encontrados.

Existem vários tipos de heurísticas que podem ser utilizadas para realizar uma avaliação heurística. Porém, para esta pesquisa foram escolhidas as heurísticas de Nielsen (1994), por existir abundante literatura sobre as mesmas. Por isso, apenas essas heurísticas serão brevemente apresentadas a seguir (NIELSEN, 1994):

1. Visibilidade e reconhecimento do estado ou contexto atual do sistema: o sistema deve dialogar com o usuário por meio de feedback apropriado, em tempo razoável.

2. Compatibilidade com o mundo real: o sistema deve usar termos próximos ao do usuário, utilizando palavras familiares e dispor as informações em ordem lógica mantendo a coerência com o modelo mental do usuário.

3. Controle e liberdade do usuário: Dar ao usuário o controle do processamento de suas ações, oferecendo a opção de desfazer e refazer operações.

4. Consistência e padrões: Contextos e situações similares devem apresentar comportamentos similares. Uma ação deve ser representada por apenas um ícone ou palavra e deverá ser formatada em todas as telas da mesma maneira.

5. Prevenção de erros: O sistema deve prevenir possíveis erros e corrigi-los, caso ocorram.

6. Reconhecimento ao invés de memorização: A interface deve oferecer uma ajuda contextual sendo capaz de orientar o usuário. Comparando Métodos de Avaliações de Usabilidade, de Encontrabilidade e Experiência do Usuário 87 Informação & Tecnologia (ITEC), Marília/João Pessoa, v.3, n.1, p.83-101, jan./jun. 2016.

7. Flexibilidade e eficiência de uso: A interface deve se adaptar ao contexto ao mesmo tempo prover eficiência de uso. O sistema deve ser fácil para usuários leigos, mas também, permitir aos usuários experientes personalizar ações frequentes.

8. Projeto estético minimalista: As interfaces devem ser mais simples possíveis e o fluxo de informações deve ocorrer de acordo com a necessidade do usuário.

9. Diagnosticar e corrigir erros: o sistema deve oferecer suporte aos usuários no reconhecimento de problemas. As mensagens de erros devem ser claras, indicando precisamente o problema e sugerindo soluções.

10. Ajuda e documentação: Caso necessário, a documentação de auxílio do sistema deve ser fácil de usar e estar sempre disponível online.

Pereira (2011) afirma que “uma avaliação heurística deve ser realizada por equipes

de 3 a 5 pessoas que, individualmente, percorrem a interface anotando os problemas encontrados, as heurísticas desobedecidas e apresentando suas considerações em relação à gravidade do problema". Soares (2004) aponta que a principal vantagem deste método é "a não exigência de avaliadores com especialização em usabilidade e nem a necessidade do envolvimento do usuário".

2.3.1 Testes de Funcionalidades de Software

No início da criação de um projeto, como por exemplo um *software*, tem-se uma visão do sistema, e sobre o que ele deve fazer, uma vez que os requisitos já foram levantados. Porém, durante o processo de desenvolvimento, é natural que essa visão se altere, e os requisitos também, e o principal motivo dessas mudanças são solicitações do cliente ou problemas encontrados nas funcionalidades.

2.3.2 Funcionalidade e Desempenho de Software – Testes e Validação

No processo de desenvolvimento de sistemas, funcionalidade (ou atividade) é definida como um comportamento ou uma ação para a qual possa ser visualizado um início e um fim; isto é: algo passível de execução.

Os testes utilizando os requisitos funcionais é a verificação de uma aplicação, e se ela é apta a realizar tais funções para as quais foi desenvolvida. Existem várias formas de fazer o teste funcional, tanto manualmente, como automatizada ou até mesmo mistura de ambos.

A caixa-preta, é um teste baseado nos requisitos funcionais do software. Esta técnica não está preocupada com o comportamento interno do sistema durante a execução do teste, mas sim com a saída gerada após a entrada dos dados especificados. Esse tipo de teste é indicado para detectar erros de interface, de comportamento ou de desempenho, podendo ser aplicada em todas as fases de testes (unidade, integração, sistema e aceitação).

Uma dificuldade dessa técnica, por questões de tempo e recurso, é testar todas as entradas possíveis. Essa técnica de teste apresenta-se necessária durante o desenvolvimento de um sistema, contudo, por sua natureza, mostra-se insuficiente para identificar certos riscos em um projeto de *software*. Sua validação é o processo de confirmar que a especificação de uma fase ou do sistema completo é apropriada e consistente com os requisitos dos *stakeholders*. (BASTOS et al., 2007).

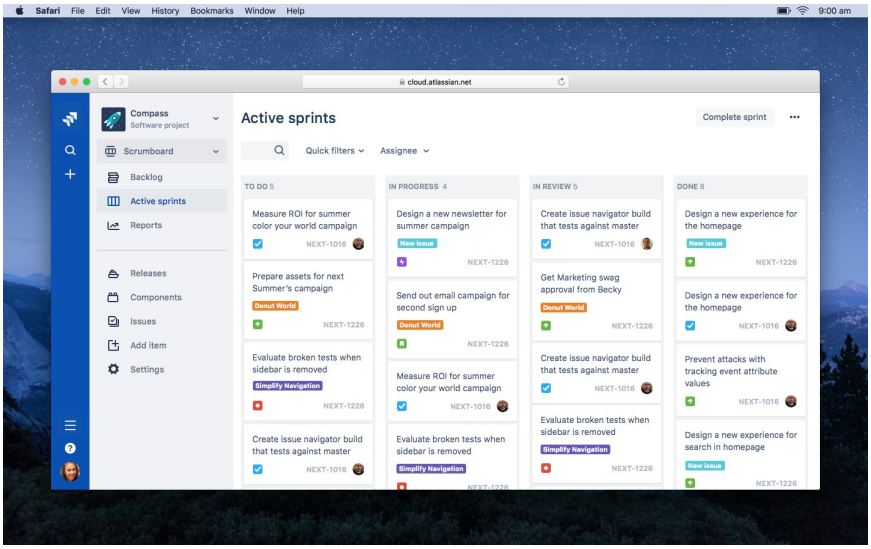
2.4 Trabalhos Correlatos

Este capítulo apresenta trabalhos relacionados ao tema desta monografia.

2.4.1 Jira Software

O JIRA é a principal ferramenta de rastreamento de bugs, rastreamento de problemas e gerenciamento de projetos. O JIRA combina rastreamento de problemas, gerenciamento ágil de projetos, fluxo de trabalho personalizável e uma estrutura de integração conectável para aumentar a velocidade de sua equipe de desenvolvimento de software. Na imagem abaixo é possível observar a interface da ferramenta (GUILLAUMEP, 2010).

Figura 1 – Sprints atividades

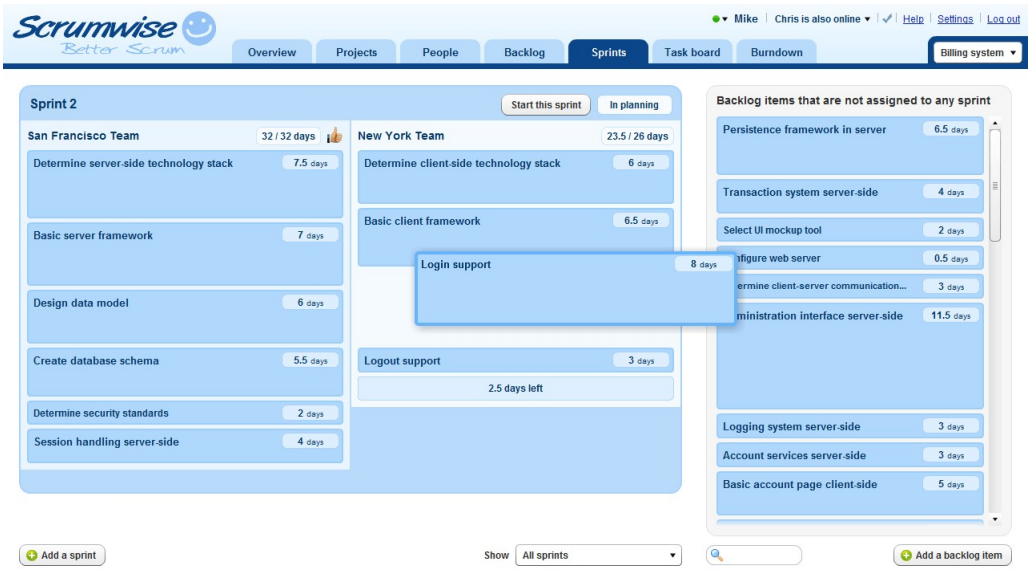


Fonte: (GUILLAUMEP, 2010)

2.4.2 Scrumwise

Scrumwise é uma das principais ferramentas para gerenciamento ágil de projetos usando *Scrum* e *Kanban*. Ele tem suporte para gerenciamento de *backlog*, planejamento de *release*, planejamento de *sprint*, quadros de tarefas, gráficos *burndown*, quadros *Kanban* e rastreamento de tempo, na figura a seguir nota-se que se trata de uma ferramenta com interface de fácil compreensão (ALTERNATIVETO, 2014).

Figura 2 – Sprints

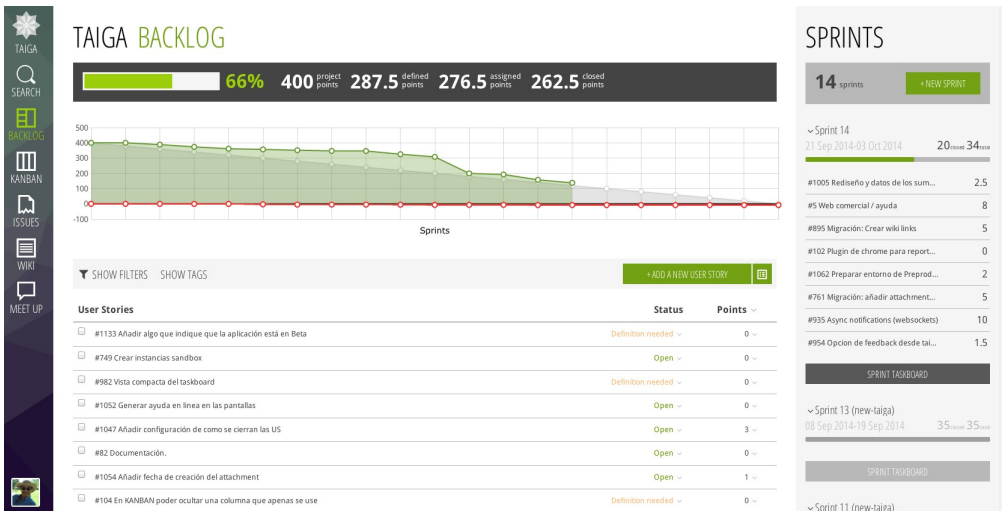


Fonte: (ALTERNATIVETO, 2014)

2.4.3 Taiga.io

Taiga é uma plataforma completa de gerenciamento de projetos para *startups* e desenvolvedores ágeis e designers que querem uma ferramenta simples e bonita que torne o trabalho realmente agradável. Taiga é grátis para projetos públicos e para um projeto privado com até 4 membros. Outros planos pagos com mais projetos e mais membros também estão disponíveis, a ferramenta possui uma interface mais elaborada, como mostra a figura abaixo (EPOSNER, 2014).

Figura 3 – Backlog



Fonte: (EPOSNER, 2014)

3 MATERIAIS E MÉTODOS

Este capítulo descreve os materiais, instrumentos e procedimentos necessários para a elaboração do projeto.

3.1 Ferramentas

O *framework open-source* JUnit será utilizado neste projeto, auxiliando no processo de criação dos testes de maneira automatizada. Apesar de simples, JUnit proporciona um ambiente completo para a realização de testes de unidade em código Java. (SOARES, 2007).

Além do *framework* citado acima, será utilizada a ferramenta de teste estrutural JaBUTi/AJ. Uma ferramenta que possui protótipo operacional implementada em meio acadêmico, mas que poderia ser utilizada perfeitamente também no meio industrial. Se trata de uma extensão da Java *Bytecode Understanding ans Testing* (JaBUTi), que pode ser usada para testar tanto programas OO (Orientados a Objetos), quanto AO (Orientados a Aspectos). (DELAMARO; JINO; MALDONADO, 2007).

3.2 Materiais e instrumentos

Os materiais e instrumentos utilizados serão os descritos nesse capítulo.

3.2.1 NetBeans IDE 8.2

O NetBeans IDE é uma ferramenta que permite desenvolver aplicações *desktop* Java, móveis e até Web, também como aplicações HTML5 juntamente com HTML, JavaScript e CSS. O IDE *(Integrated Development Environment)*, traduzido em Ambiente de Desenvolvimento Integrado também fornece um grande conjunto de ferramentas para desenvolvedores de PHP e C/C++.

Segundo Bonfim (2007). É de fácil interação e possui uma interface bastante agradável. O IDE pode ser executado em multiplataformas, como Windows, Linux, MacOS e Solaris. Oferece aos usuários e desenvolvedores ferramentas necessárias para a criação de aplicações de desktop, Web, empresariais e móveis multiplataformas

3.2.2 HeidiSQL

HeidiSQL é um software livre e tem o objetivo de ser fácil de aprender. *"Heidi"* permite que você veja e edite dados e estruturas de computadores que executam um dos sistemas de banco de dados *MariaDB*, *MySQL*, *Microsoft SQL* ou *PostgreSQL*. Inventado

em 2002 pela Ansgar, com um pico de desenvolvimento entre 2009 e 2013, o *HeidiSQL* pertence às ferramentas mais populares para *MariaDB* e *MySQL* em todo o mundo. (HEIDISQL, 2019)

3.2.3 XAMPP

O objetivo do XAMPP é construir uma distribuição fácil de instalar para desenvolvedores entrarem no mundo do Apache. Para torná-lo conveniente para os desenvolvedores, o XAMPP é configurado com todos os recursos ativados. No caso de uso comercial, por favor dê uma olhada nas licenças de produtos; do ponto de vista do XAMPP, o uso comercial também é gratuito. Há atualmente distribuições para Windows, Linux e OS X. (XAMPP, 2019)

3.2.4 Software Easy Scrum

O software utiliza o próprio Scrum no gerenciamento de projetos desenvolvidos durante aulas de Laboratório de Engenharia de Software.

Demonstra os resultados benéficos da utilização deste método no desenvolvimento de um software, e no gerenciamento da equipe como um todo. Além de proporcionar o contato dos alunos com o método Scrum, ambientando-os com o mercado de trabalho.

Uma ferramenta, onde orienta e ensina o uso do método Scrum no gerenciamento e desenvolvimento de projetos, e auxiliar no aprendizado da Engenharia de Software. (NODA; HORACIO, 2018).

3.3 Procedimentos

Segundo Filho e Rios (2003) O teste caixa preta tem o objetivo de averiguar a **funcionalidade** e os requisitos do sistema, é uma visão externa para o usuário, não se baseia em nenhum conhecimento de código, nem na lógica interna do que for testado.

Será observado toda funcionalidade do sistema que será avaliado, funções incorretas, erros de interface, erros de comportamento ou desempenho, simulando erros que o usuário pode cometer utilizando o *framework* open-source JUnit e o JaBUTi/AJ.

No teste de **usabilidade** serão convocados estudantes de análise e desenvolvimentos de sistemas para utilizar o *software*, enquanto eles utilizam será observado cada um dos discentes, para tentar obter melhoras na interface do sistema.

Os testes serão realizados na faculdade de tecnologia de ourinhos FATEC. Será escolhido um dia para reunir alunos do 6º semestre de análise e desenvolvimento de sistema. Após os testes, serão analisados os resultados.

3.3.1 BDD - *Behavior Driven Development* (Desenvolvimento Guiado por Comportamento ou Desenvolvimento Orientado a Comportamento)

O *BDD* é uma metodologia que tem o intuito de integrar regras de negócio através de testes automáticos que tem como foco o comportamento do *software*, ou seja, o *BDD* visa escrever testes automáticos baseados em cenários de negócio do sistema (COSTA, 2016).

O objetivo da metodologia é utilizando uma linguagem ubíqua, escrever testes que qualquer pessoa do projeto seja capaz de compreender sem obrigatoriamente possuir um domínio técnico (COSTA, 2016).

4 RESULTADOS E DISCUSSÕES

O objetivo dos testes de usabilidade e funcionais que foram aplicados no software "Easy Scrum" é analisar e avaliar se suas funcionalidades, desempenho e usabilidade estão adequados a uma ferramenta didática, identificar e relatar todos os erros encontrados dentro do sistema e apresentar uma sugestão de continuidade ao projeto.

4.1 Porque foi escolhido o método de avaliação heurística

As técnicas de avaliação de interface se diferem entre si em vários aspectos. É preciso entender as diferentes características de cada método, para definir qual melhor se encaixaria no seu projeto para avaliar a interface em um determinado contexto.

De forma breve testes de usabilidade, são testes aonde os principais envolvidos é a pessoa que irá utilizar, iniciando pela coleta de dados, informações que normalmente é para identificar o nível de satisfação da pessoa com o sistema. Essa coleta de dados é feita através de questionários e entrevistas. Outra forma também é observar o usuário. Para o usuário que tem dificuldades em expressar o que pensa pode-se usar a técnica de observação. Então, o avaliador vai observar o usuário utilizando o *software* da forma que ele acredita que seja apropriado, da maneira pelo qual ele considera adequado ou desejável. Essas observações podem ser registradas utilizando gravações, anotações, lembrando que para gravar, é feita uma autorização do usuário. E locais para esse tipo de observação poder ser algo simples como a própria casa da pessoa ou em um laboratório onde tem todos os equipamentos.

A técnica conhecida como avaliação heurística que vista identificar problemas de usabilidade conforme um conjunto de heurísticas ou diretrizes Nielsen (1994) este método é bastante rápido e de menor custo.

4.1.1 Elaborações do teste de usabilidade

Para organizar melhor a elaboração do teste foi realizada quatro passos:

Primeiro passo a escolha dos avaliadores. Com ajuda dos alunos da FATEC Ourinhos do 6 termo, foi feita o teste em sala de aula, separando dois grupos, Grupo 1 com dois alunos e Grupo 2 com três alunos, todos tendo uma noção básica do *Scrum* facilitando o teste, apontando os erros.

Segundo passo a avaliação Heurística.

Para aplicação dos testes foi escolhidas as dez heurísticas de Nielsen (1994), aonde propôs um conjunto básico de heurísticas.

Terceiro passo, Grau de Severidade.

Quando um problema qualquer for detectado, classifique-o em uma das dez heurísticas de Nielsen, anotando o problema na tabela correspondente e atribuindo o **grau de severidade** (0 até 4) para este problema conforme tabela 1.

Tabela 1 – Grau de severidade dos problemas de usabilidade.

Grau de severidade	**Tipo**	**Descrição**
0	Sem importância	Não afeta a operação da interface.
1	Cosmético	Não há necessidade imediata de solução.
2	Simples	Problema de baixa prioridade (pode ser reparado).
3	Grave	Problema de alta prioridade (deve ser reparado).
4	Catastrófico	Muito grave, deve ser reparado de qualquer forma.

Quarto passo, formulário para avaliação heurística.

Na tabela 2 abaixo, com base de várias pesquisas de formulários e exemplo para avaliação, foi feita um exemplo.

Tabela 2 – Formulário

Heurística violada	( ) 1. Visibilidade do estado do sistema; ( ) 2. Correspondência entre o sistema e o mundo real; ( ) 3. Controle e liberdade do usuário; ( ) 4. Consistência e padronização; ( ) 5. Prevenção de erro; ( ) 6. Ajuda para usuários para diagnosticarem e recuperarem erros; ( ) 7. Reconhecimento; ( ) 8. Flexibilidade e eficiência de uso; ( ) 9. Design estético e minimalista; ( ) 10. Ajuda e documentação;
Gravidade	( ) 0. Não concordo que isto seja um problema; ( ) 1. Problema cosmético: não precisa ser consertado a menos que haja tempo extra no projeto; ( ) 2. Problema simples: o conserto deste problema é desejável, mas deve receber baixa prioridade; ( ) 3. Problema grave: importante de ser consertado; deve receber alta prioridade; ( ) 4. Problema Catastrófico: é imperativo consertar este problema antes do lançamento do produto;

Foram feito agora alguns procedimentos para continuação do teste:

-Anota os problemas encontrados e sua localização;

-Aponta a gravidade destes problemas;

-Gera um relatório individual para cada resultado de sua avaliação e comentários adicionais, dentro da interface do *software*;

Durante cada sessão de avaliação, os grupos discutem sobre os possíveis erros, inspecionando os diversos elementos de interface e comparando-os com a lista de heurísticas de usabilidade. O teste teve um tempo de duração máximo de 2 horas realizados no mesmo dia.

4.2 Problemas encontrados

Foram encontrados 24 problemas de usabilidade. Abaixo uma lista dos problemas de usabilidade encontrados pelo método de avaliação heurística:

4.2.1 Problema na tela de Cadastro Login

Heurística violada: 1, 5 e 7.

Problema: Ao fazer cadastro de Login, o aluno nota que permite pontos no CPF, mas não é aceitável na hora de salvar.

Explicação: Ao informar o CPF, o usuário insere os pontos corretamente conforme o documento, mas quando clica para salvar ele não aceita, não dando um aviso *(feedback)* de que isso poderia ocorrer.

Gravidade: 2.

4.2.2 Problema na tela de Cadastro Login 2

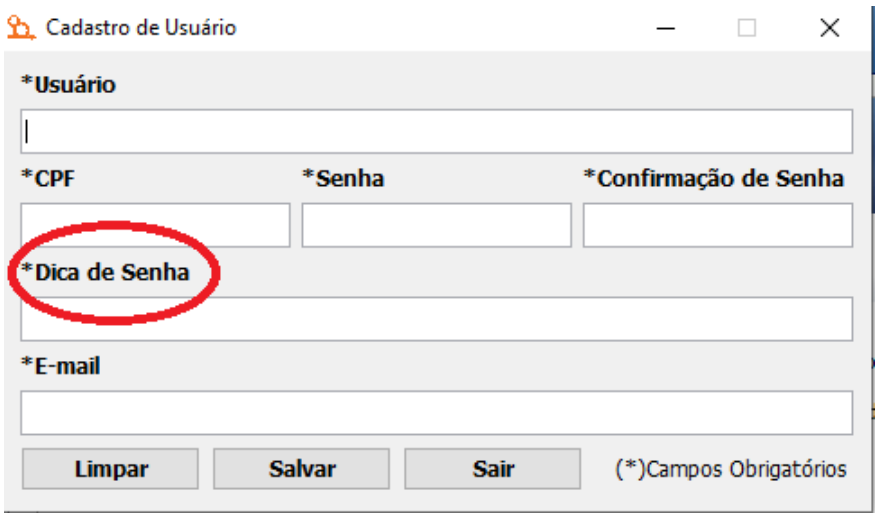
Heurística violada: 8.

Problema: Ao fazer cadastro de Login, um dos campos na tela está como obrigatório.

Explicação: O usuário preenche os campos, mas não tem a necessidade de ter como obrigação uma dica de senha. Fazendo assim o usuário ter uma “dica de senha” obrigatória para continuação do cadastro como mostra a figura 4.

Gravidade: 3.

Figura 4 – *Print* do erro (1) como mostra a figura 4.



Fonte: **(Noda e Horacio, 2019)**

4.2.3 Problema na tela de Cadastro Login 3

Heurística violada: 1,5 e 6.

Problema: Aceita CPF duplicado.

Explicação: O usuário pode perfeitamente colocar o mesmo CPF para outro tipo de nome, que o sistema aceita.

Gravidade: 4.

4.3 Problema na tela Login

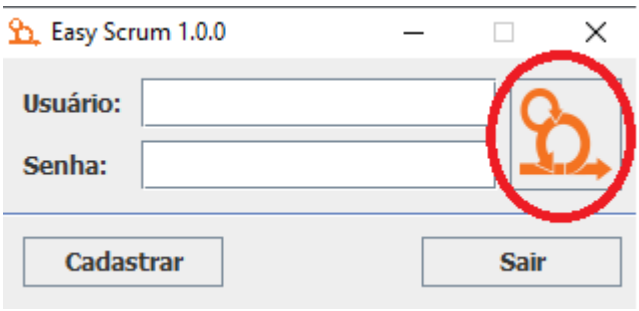
Heurística violada: 1 e 7.

Problema: Especificar botão de entrada.

Explicação: Ao se deparar com a tela de Login o usuário procura o botão de entrada, tendo uma certa dificuldade em identifica-lo, pois, como mostra a figura abaixo, há uma figura sobre ele.

Gravidade: 2

Figura 5 – *Print* do erro (2) tela de login.



Fonte: **(Noda e Horacio, 2019)**

4.4 Problema em *Product Owner*

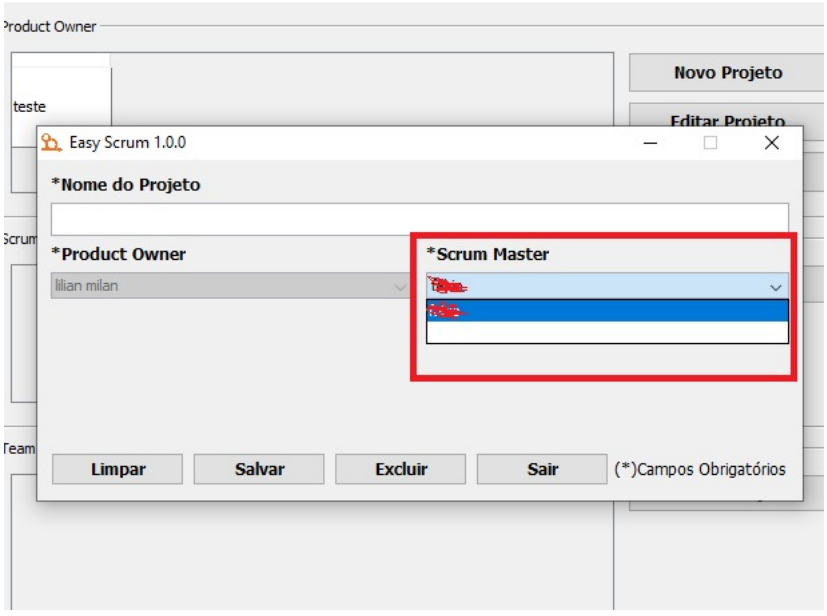
Heurística violada: 1, 3, 5, 6 e 8.

Problema: Campo em branco na hora de selecionar Scrum Master pelo *Product Owner*.

Explicação: Na hora de selecionar o *Scrum Master*, o *Product Owner* encontra um campo em branco mostrado na figura 6, se o usuário clicar nesse campo, o sistema automaticamente fecha, dando um erro irreversível.

Gravidade: 4.

Figura 6 – *Print* do erro (3) tela novo projeto.



Fonte: **(Noda e Horacio, 2019)**

4.4.1 Problema em *Product Owner 2*

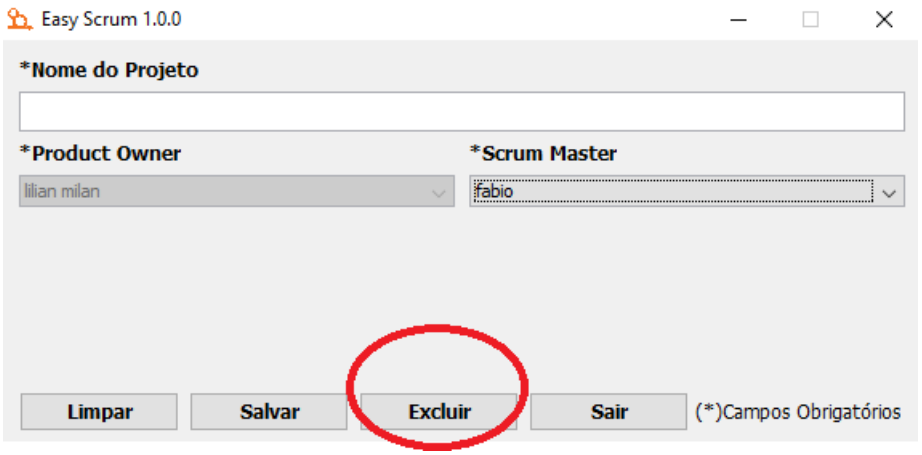
Heurística violada: 4.

Problema: Ao adicionar novo projeto o botão excluir não fica inativo.

Explicação: Ao criar um novo projeto o botão de excluir continua ativo como mostra a figura abaixo, porém, não tem como excluir algo que ainda não foi inserida no sistema.

Gravidade: 2.

Figura 7 – *Print* do erro (4) tela novo projeto.



Fonte: **(Noda e Horacio, 2019)**

4.4.2 Problema em *Product Owner 3*

Heurística violada: 4 e 5.

Problema: No botão "editar projeto" ele aceita, mas não atualiza no banco de dados.

Explicação: Ao clicar para editar projeto em seguida o botão de salvar, ele manda um *feedback* ao usuário que salvou sem problemas, porém ao verificar o banco de dados o projeto está com o nome antigo e não o que o usuário pediu para atualizar.

Gravidade: 3.

4.4.3 Problema em *Product Owner 4*

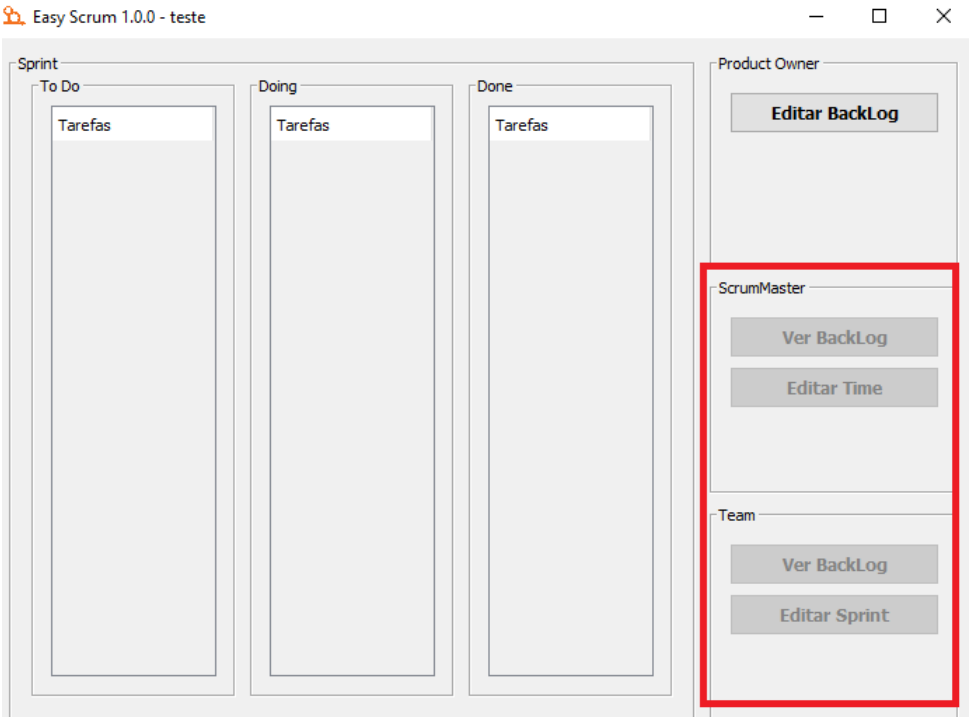
Heurística violada: 4.

Problema: Acessar projeto os botões que estão aparentemente inativos na verdade estão funcionando.

Explicação: O usuário consegue navegar perfeitamente pelos botões que aparentemente estão inativos, porém, estão apenas "apagados" como retrata a imagem a seguir.

Gravidade: 3.

Figura 8 – *Print* do erro (5) tela acessar projeto.



Fonte: **(Noda e Horacio, 2019)**

4.4.4 Problema em *Product Owner 5*

Heurística violada: 4 e 5.

Problema: Sem controle de data ao criar *sprint*.

Explicação: Ao clicar na data, o usuário pode colocar qualquer número que o sistema aceita, não tendo uma validação correta de data, isso pode ser observado na figura 9.

Gravidade: 4.

4.4.5 Problema em *Product Owner 6*

Heurística violada: 4 e 5.

Problema: Ao remover uma tarefa da sprint, a tarefa é duplicada.

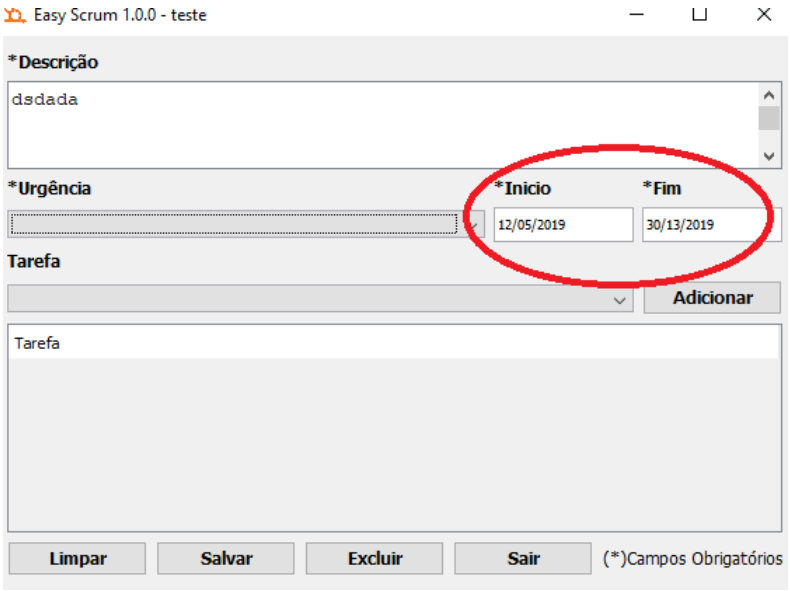
Explicação: Ao clicar em excluir, o usuário nota que a mesma tarefa foi duplicada e não excluída.

Gravidade: 4.

4.4.6 Problema em *Product Owner 7*

Heurística violada: 3 e 7.

Figura 9 – *Print* do erro (6) tela criar sprint.



Fonte: **(Noda e Horacio, 2019)**

Problema: Não existe botão para voltar a tela de login.

Explicação: O usuário pode querer mudar de login, mas não tem o botão para tal ação, o usuário tem que fechar a tela principal do sistema para abrir a tela login.

Gravidade: 2.

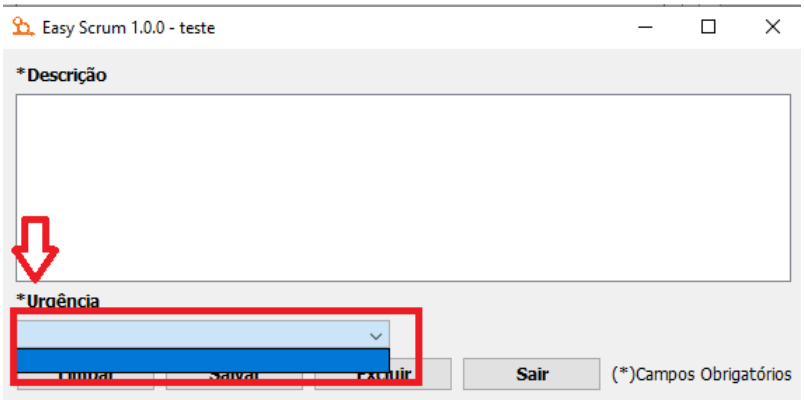
4.4.7 Problema em *Product Owner 8*

Heurística violada: 1, 3, 6 e 8. Problema: Seleção de Urgência obrigatório.

Explicação: Ao "criar tarefa" no *backlog*, o usuário precisa selecionar "urgência" para concluir a criação da tarefa, mas não há opções e nem como criar urgência, portanto não é possível preencher este campo, como é mostrado na figura 10, a seguir.

Gravidade: 4.

Figura 10 – *Print* do erro (7) tela criar tarefa.



Fonte: **(Noda e Horacio, 2019)**

4.4.8 Problema em *Product Owner 9*

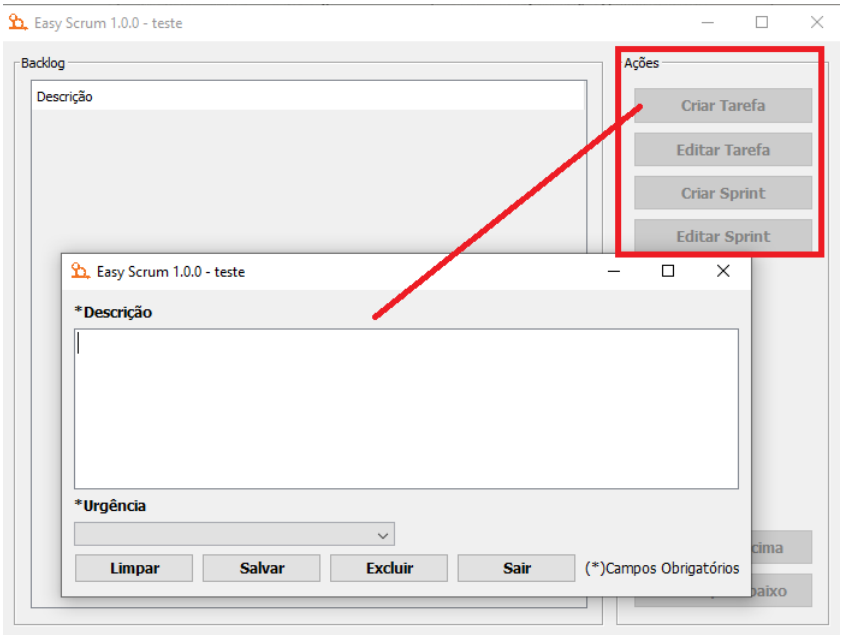
Heurística violada: 1, 3, 4, 5, 6 e 8.

Problema: Campo em branco ao criar tarefa *(sprint)* que ao clicar ocorre um erro e fecha o programa.

Explicação: Ao "criar tarefa" no *backlog*, o usuário precisa selecionar "urgência" para concluir a criação da tarefa, mas não há opções e nem como criar urgência, além dos mesmo erros anteriormente com botões inativos mas ainda assim funcionam e ao clicar no campo em branco ele fecha o sistema todo, sem o usuário conseguir reverter.

Gravidade: 4.

Figura 11 – *Print* do erro (8) tela ver backlog.



Fonte: **(Noda e Horacio, 2019)**

4.5 Problema em *Scrum Master/Team*

Heurística violada: 1, 2, 4, 5 e 8.

Problema: Tanto *Product Onwer* quanto *scrum master* e *team*, eles mexem em qualquer função.

Explicação: Problema grave, onde qualquer usuário do time pode utilizar quaisquer botões sem exceção.

Gravidade: 4.

4.6 Tabela da gravidade dos problemas

Após a realização dos testes é possível citar a gravidade de cada problema encontrado, para ser feito uma análise. Deste modo, foram encontrados 14 problemas dentre eles:

- 4 erros simples.
- 3 erros graves.
- 7 erros catastróficos.

Figura 12 – Gravidade do Problema.

Fonte: **(Autor próprio, 2019)**

A figura 12, acima, é para ter uma percepção maior da gravidade dos problemas.

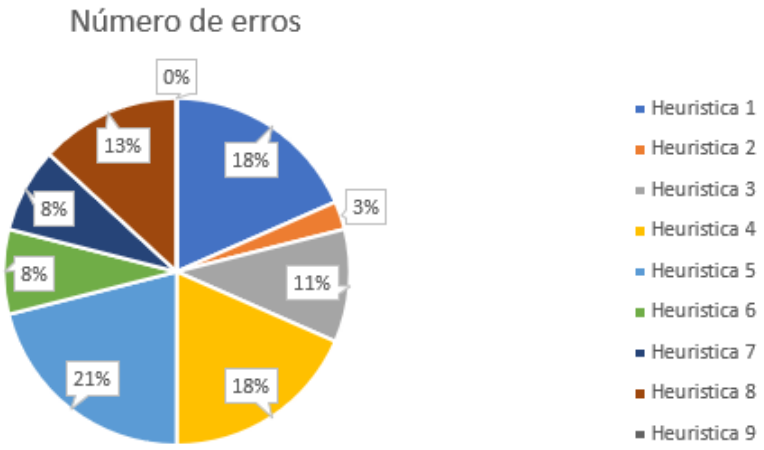
É possível notar que o grau de severidade 0 e 1 que são sem importância e cosmético estão em 0%, ou seja, não foram encontrados. Grau de severidade de número 2 simples, foram encontrados com 4 erros de 14 erros no total, tendo uma totalidade de 29%, problema com baixa prioridade, ou seja, que pode ser reparado. Com o grau de severidade de número 3, foram encontrados 3 erros, tendo uma totalidade de 21% sobre os 14 problemas. É um grau do tipo grave, problema de alta prioridade a qual deve ser reparado. E por último com o nível de severidade de número 4, encontram-se 7 erros com o total de 50%. É um grau do tipo grave, deve ser reparado de qualquer forma.

4.6.1 Heurísticas que mais obtiveram erros

Após a realização dos testes é possível observar as heurísticas encontradas nos 14 problemas de usabilidade. Abaixo, uma lista com o número da heurística e a quantidade de erros.

- Heurística 1 (7 erros).
- Heurística 2 (1 erro).
- Heurística 4 (7 erros).
- Heurística 5 (8 erros).
- Heurística 7 (3 erros).
- Heurística 8 (5 erros).

Figura 13 – Números de erros.



Fonte: **(Autor próprio, 2019)**

Na figura 13, podemos ter uma noção maior dos problemas heurísticos. Foram encontrados 31 erros de heurísticas com base em 14 problemas no total. Podemos notar que o maior número de erros com, 8 erros e 21% na tabela foi na heurística de número 5, conhecida como "Prevenção de erro".

Logo abaixo, com 18%, com uma taxa de 7 erros a heurística de número 1: "Visibilidade do estado do sistema" e heurística de número 4: "Consistência e padronização". Depois com 13% na tabela, com uma taxa de 5 erros a heurística de número 8: "Flexibilidade e eficiência de uso". Após, disso com 8% na tabela e com 3 erros a heurística de número 7 conhecida como "Reconhecimento". E por último, a heurística de número 2: "Correspondência entre o sistema e o mundo real", com 3% na tabela e com 1 erro.

4.6.2 Teste Funcional

Desde o início da interação com o sistema tivemos imensa dificuldade para compreender, de imediato não foi possível executa-lo, pois haviam erros tanto de sintaxe como de programação. Foi necessário pedir ajuda aos docentes da faculdade para efetuar melhorias

necessárias para que o software pudesse ser executado, já sendo possível constatar alguns erros funcionais.

Durante a execução do JUnit tivemos grande dificuldade, pois o JUnit é indicado quando o sistema ainda não possui front end, como o software *Easy Scrum* já possui telas, não foi possível utiliza-lo, optamos então por fazer o teste caixa-preta, analisando o material que já tínhamos do *Easy Scrum.*

Haviam trechos de código criptografados desnecessariamente, impedindo que o sistema compilasse, além disso existiam erros graves no banco de dados, trazendo necessidade de criar um novo.

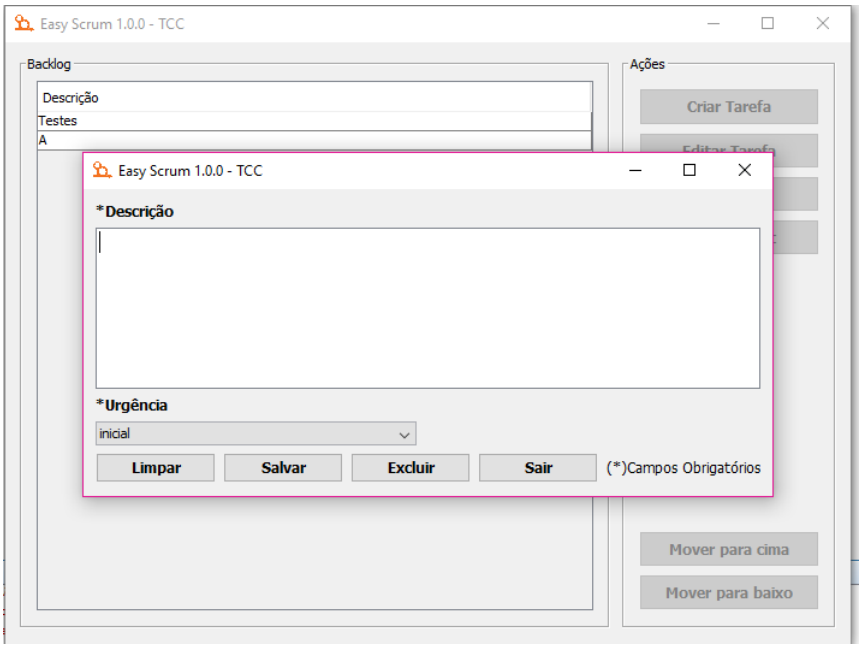
Depois de resolvidas as questões citadas acima, finalmente foi possível fazer a análise do *software Easy Scrum.*

4.6.3 Teste de Caixa-Preta

Também é chamado de teste funcional, porque é baseado nos requisitos funcionais do *software.* O foco, nesse tipo de teste, é nos requisitos da aplicação, ou seja, nas ações que deve desempenhar (COSTA, 2018).

Pode-se citar alguns exemplos de problemas que esse tipo de teste pode detectar, como data de conclusão anterior à data de início; campos de preenchimento obrigatório que não são validados; botões que não executam as ações devidas, todo tipo de falha funcional, ou seja, falhas que contrariam os requisitos do sistema (COSTA, 2018).

Figura 14 – Criar Tarefa

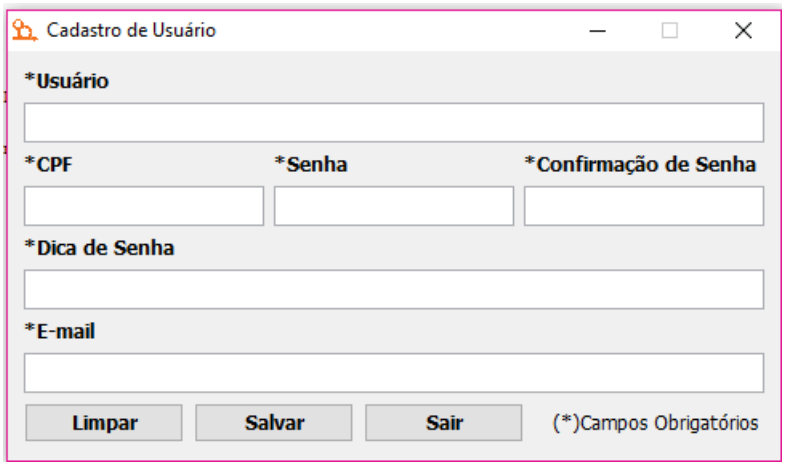


Fonte: **(Noda e Horacio, 2019)**

No dia 09 de maio de 2019, foi realizado o teste de usabilidade do sistema, com

alunos do sexto semestre do curso de Análise e Desenvolvimento de Sistemas da Fatec Ourinhos. Durante o mesmo, foi possível detectar um campo obrigatório na tela “Criar Tarefa” chamado “Urgência”, essencial para dar continuidade ao teste, porém, não havia possibilidade de cadastrar este campo, foi necessário criar na hora um dado neste campo, para prosseguir com o teste, a figura 14 acima mostra a tela após a criação da urgência.

Figura 15 – Cadastro de Usuário



Fonte: **(Noda e Horacio, 2019)**

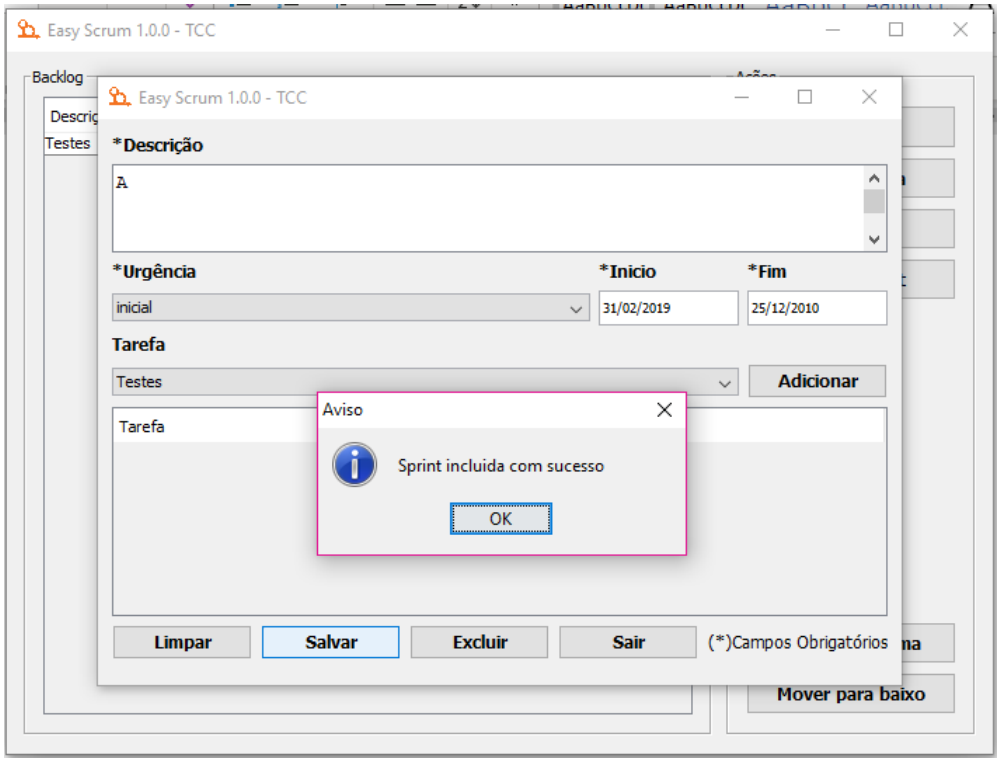
Ao efetuar o cadastro de um novo usuário, o sistema aceita somente CPFs e e-mails válidos.

O sistema aceita que mais de um usuário se cadastrem utilizando o mesmo número de CPF.

O sistema permite, mas não cadastra pontos “.” no campo CPF.

Na figura 15 acima pode-se ver a tela de cadastro em que é inserido o CPF.

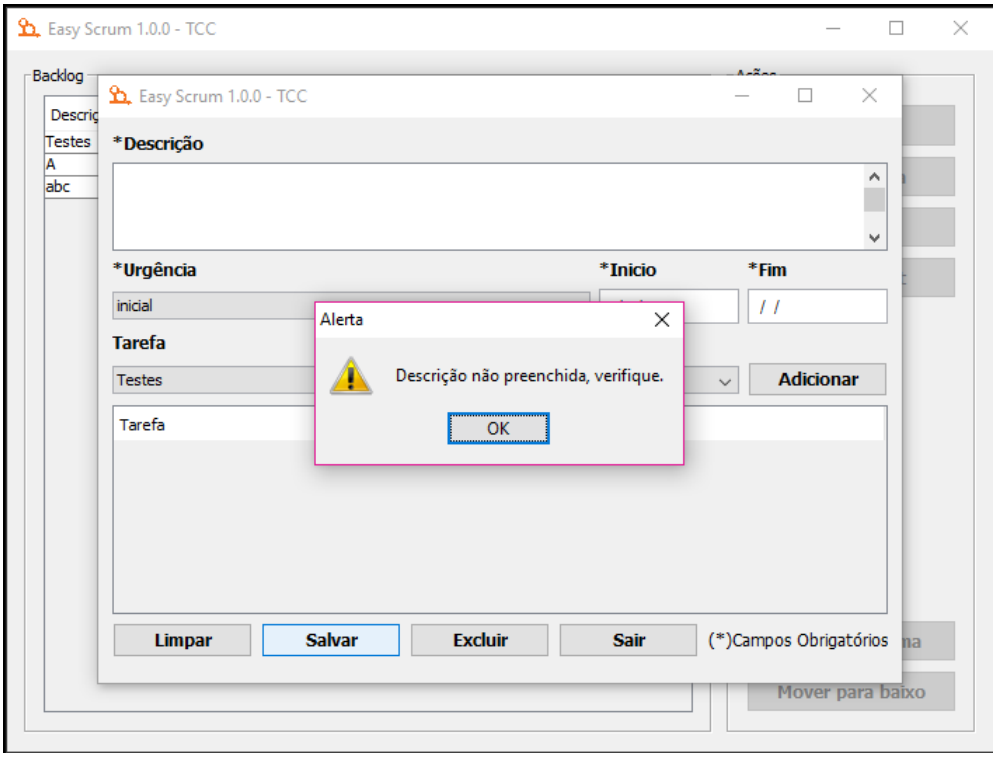
Figura 16 – Criar Sprint



Fonte: **(Noda e Horacio, 2019)**

Falta validação de datas, o sistema permite que seja criada uma *sprint* com datas inválidas, como mostrado no exemplo acima.

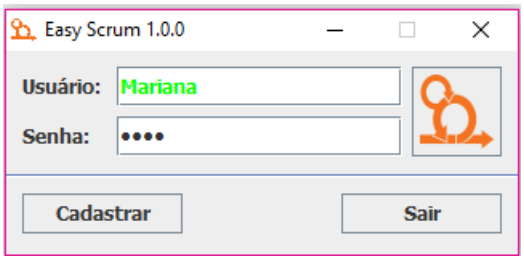
Figura 17 – Criar Sprint 2



Fonte: **(Noda e Horacio, 2019)**

Na criação de uma nova *sprint*, todos os campos precisam estar preenchidos, caso contrário, o sistema acusa mostrando o aviso representado na figura 17 acima.

Figura 18 – Login

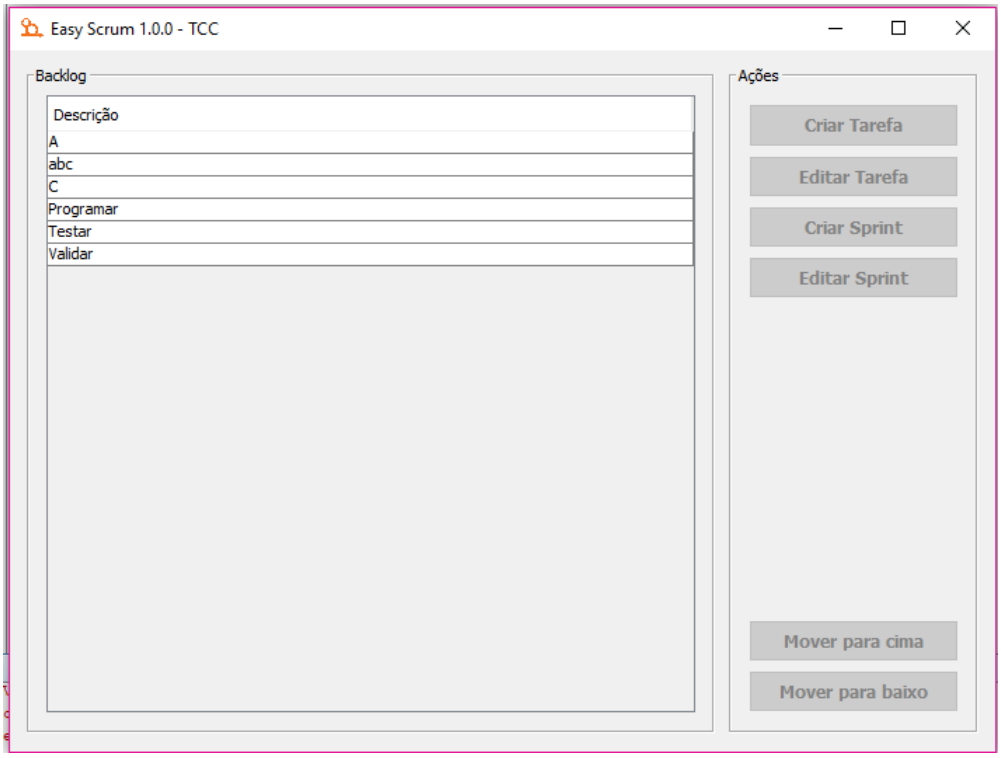


Fonte: **(Noda e Horacio, 2019)**

Ao inserir os dados para efetuar o login, a fonte fica verde quando se trata de um usuário cadastrado, e vermelho caso o usuário ainda não possua cadastro no sistema.

Falta clareza no botão para login que é representado pela figura 18.

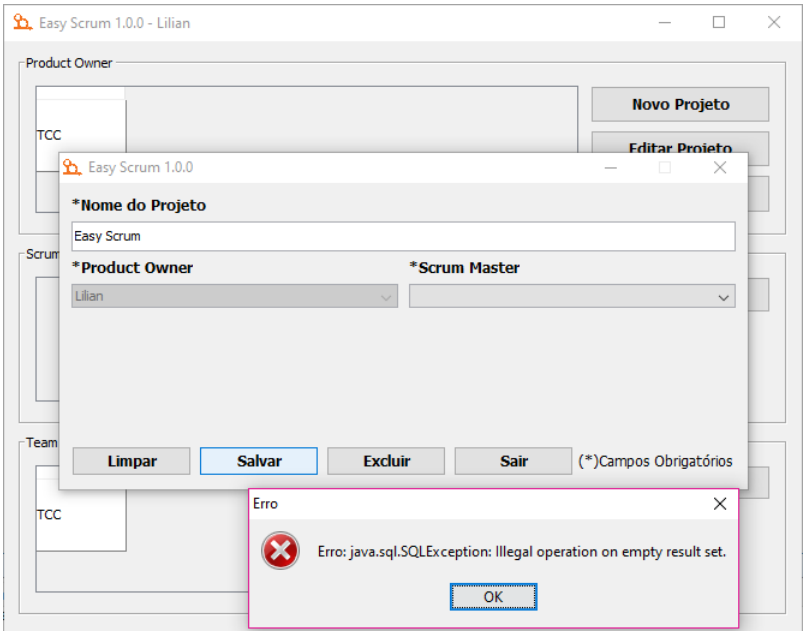
Figura 19 – Ver Backlog



Fonte: **(Noda e Horacio, 2019)**

Os botões que deveriam estar inativos, na verdade estão apenas “apagados” como mostrado na imagem 19, caso clique sobre eles, continuam funcionando, coisa que não deveria acontecer.

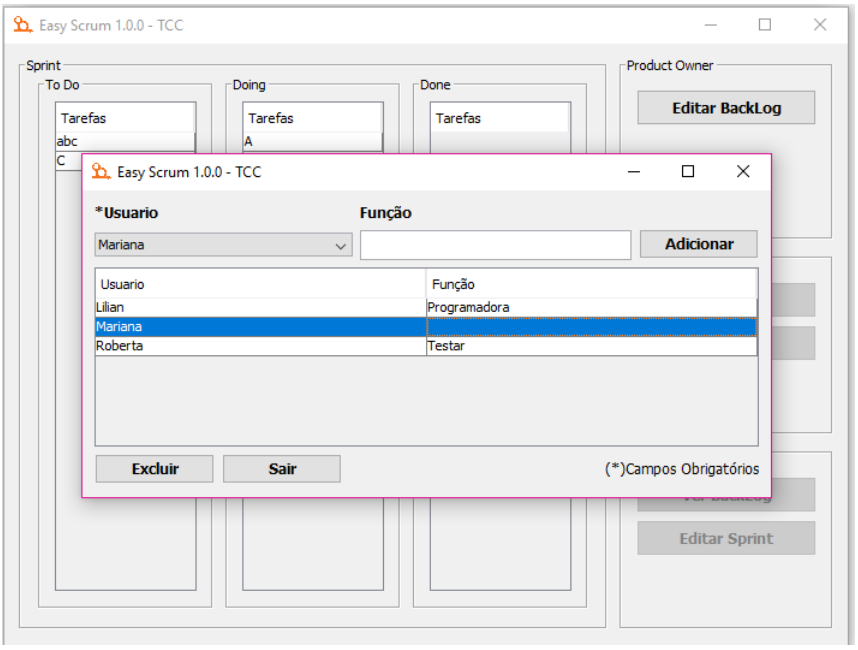
Figura 20 – Novo Projeto



Fonte: **(Noda e Horacio, 2019)**

No momento de efetuar o cadastro de um novo projeto, aparece um campo em branco selecionável no tópico *"Scrum Master"*, caso clique nele e salve, provoca um "erro fatal" mostrado na imagem 20, causando o fechamento imediato de toda a aplicação.

Figura 21 – Editar Time



Fonte: **(Noda e Horacio, 2019)**

Na edição do *time* do projeto, o sistema permite que se adicione um membro, sem que seja especificada a função que ele irá desempenhar, ou seja, com um campo vazio que pode ser observado na figura 21 acima, o que não deveria acontecer.

4.7 Conclusão dos Resultados

É possível observar com os testes realizados, que o maior número de erros sendo 8 erros que foram encontrados na heurística 5 e tem o seguinte significado segundo Prates e Barbosa (2018) (NIELSEN, 1993): "Prevenção de erro: tente evitar que o erro aconteça, informando o usuário sobre as consequências de suas ações ou, se possível, impedindo ações que levariam a uma situação de erro". Para isto o sistema deve estar preparado para se proteger contra erros, emitir mensagens de erro com qualidade e proceder à recuperação de erros ocorridos. Isso implica em detectar a ocorrência de erros na entrada de dados, informando ao usuário de forma clara e concisa seu motivo e como corrigi-lo e, quando possível, realizar a correção automática do erro".

Sobre usabilidade entende-se que é a interação entre homem-computador. A usabilidade é a parte intermediaria entre o usuário e a tecnologia, é a forma como o usuário interpreta essa tecnologia (máquina) e como essa tecnologia (máquina) responde para o usuário. Um fato primordial sobre usabilidade segundo o Krug (2008): "Não me faça

pensar", sendo uma das frases mais importantes que cita no seu livro, quer dizer que essa frase é um fator de desempate ao decidir se algo funciona ou não em um projeto web.

Mensagens de erro são, possivelmente, a coisa mais frustrante que se pode acontecer a um ser humano ao utilizar uma interface, seja ela de dispositivos móveis ou não. O erro corta a linha de raciocínio, não apresenta soluções práticas e desestimula qualquer ação que ocorria entre o usuário e o sistema. A 5ª heurística de Nielsen apresenta a prevenção de erros como um fator fundamental para a boa execução de um projeto de interface. (FROEMMING, 2011)

Além de assegurar os aspectos técnicos de um software focado para alunos, deve-se ser levado em considerações os aspectos de aprendizagem, os desenvolvedores devem assegurar que pedagogicamente seja bem utilizado. Para isso um educador deve analisar o software de aprendizagem, identificando o conceito de aprendizagem. Para ser educativo deve ser pensado como o usuário aprende como ele vai adequar a isso para construir seu conhecimento, colocando seu futuro projeto. (WASSERMANN, 2014)

5 CONCLUSÃO

O objetivo desse projeto foi identificar e analisar os problemas de usabilidade e funcionalidade do software “Easy Scrum”, para desenvolvê-lo foram necessárias pesquisas a respeito de teste de software, de usabilidade e de funcionalidade.

O principal objetivo da usabilidade é que algo funcione bem,partindo dessa ideia, o software com disponibilidade de colocar um projeto para controle não pode conter erros, porque o usuário não vai querer utilizar o software que não funcionam as ferramentas necessárias.

O objetivo foi atingido, no entanto, pudemos constatar que não há possibilidade de realizar todas as correções necessárias no sistema, durante a utilização do software foi possível encontrar diversos problemas tanto na funcionalidade quanto na usabilidade, a primeira dificuldade encontrar após o software em mãos, foi o banco de dados, onde precisamos conversar com um professor, depois a dificuldade da própria codificação, pois estava criptografada, obtemos ajuda com outro professor. Sendo assim, para continuidade do trabalho sugere-se refazê-lo, podendo haver reaproveitamento de código.

Chegamos a essa conclusão porque, 21% dos erros encontrados são nível 3 de severidade (graves), de alta prioridade. E 50% dos erros encontrados são nível 4 de severidade (catastróficos), deveriam ter sido corrigidos antes do lançamento do produto, ou seja, 71% do sistema está altamente comprometido.

REFERÊNCIAS

ALTERNATIVETO. **Scrumwise**. 2014. Disponível em: <https://alternativeto.net/software/scrumwise>. Acesso em: 11 de Maio de 2019.

BARROS, R. P. **Evolução, Avaliação e Validação do Software RoboEduc**. [S.l.]: Dissertação de Pós-graduação: Universidade Federal do Rio Grande do Norte. Rio Grande do Norte, 2011.

BASTOS, A. et al. **Base de conhecimento em teste de software**. [S.l.]: São Paulo: Martins, 2007.

BECK, K. et al. **Manifesto para o desenvolvimento ágil de software.** 2001. Disponível em: <http://www.manifestoagil.com.br>. Acesso em: 30 de Outubro de 2018.

BERTINI, S. G. E.; KIMANI, S. **Apropriating and Assessing Heuristics for Mobile Computing**. 2006. Disponível em: <http://doi.acm.org/10.1145/1133265.1133291>. Acesso em: 11 de maio de 2018.

BONFIM, J. A. de J. **Produtividade de construção de Sistemas Web com JSF**. 2007. Disponível em: <http://monografias.poli.ufrj.br/monografias/monopoli10019562.pdf>. Acesso em: 13 de Novembro de 2018.

BRASILEIRO, R. **Manifesto Ágil: o que é e qual a sua história.** 2018. Disponível em: <http://www.metodoagil.com/manifesto-agil>. Acesso em: 30 de Outubro de 2018.

COMITÊ BRASILEIRO DE COMPUTADORES E PROCESSAMENTO DE DADOS. **NBR ISO/IEC-9126-1**: Engenharia de software: Qualidade de produto. Rio de Janeiro, 2003. 21 p.

COSTA, M. **BDD – Foco no comportamento do sistema.** 2016. Disponível em: <http://www.matera.com/blog/post/bdd-validando-o-comportamento-sistema>. Acesso em: 19 de Maio de 2019.

COSTA, P. **Conceitos: testes de caixa branca e caixa preta.** 2018. Disponível em: <https://app.crowdtest.me/teste-caixa-branca-caixa-preta>. Acesso em: 19 de Maio de 2019.

DELAMARO, M.; JINO, M.; MALDONADO, J. **Introdução ao Teste de Software.** [S.l.]: Rio de Janeiro: Campus, 2007.

EPOSNER. **Taiga.io**. 2014. Disponível em: <https://alternativeto.net/software/taiga-io>. Acesso em: 11 de Maio de 2019.

FERREIRA, K. G. **Teste de Usabilidade. Belo Horizonte.** 2002. Disponível em: <https://homepages.dcc.ufmg.br/~clarindo/arquivos/disciplinas/eu/material/referencias/monografia-avaliacao-usabilidade.pdf>. Acesso em: 31 de Outubro de 2018.

FILHO, T. R. M.; RIOS, E. **Projeto & Engenharia de Software: Teste de Software.** [S.l.]: Rio de Janeiro: Alta books Ltda, 2003.

FOWLER, M.; HIGHSMITH, J. **The agile manifesto.** [S.l.]: Software Development, 2001. v. 23. 1417–1433 p.

FROEMMING. **5ª Heurística | Prevenir Erros**. 2011. Disponível em: <https://froemming.wordpress.com/2011/12/07/5a-heuristica-prevenir-erros>. Acesso em: 19 de Maio de 2019.

GONÇALVES, M. K. **Usabilidade de Software.** [S.l.]: Dissertação de Pós-Graduação: Universidade Estadual Paulista Júlio de Mesquita Filho, Bauru, 2009.

GUILLAUMEP. **Jira**. 2010. Disponível em: <https://alternativeto.net/software/jira>. Acesso em: 11 de Maio de 2019.

HEIDISQL. **O que é isso?** 2019. Disponível em: <https://www.heidisql.com>. Acesso em: 11 de Maio de 2019.

JUNIOR, C. **Quais são os principais tipos de métodos ágeis?** Project Builder, 2017. Disponível em: <https://www.projectbuilder.com.br/blog/quais-sao-os-principais-tipos-de-metodos-ageis>. Acesso em: 22 de Agosto de 2018.

KRUG, S. **Não me faça pensar: Uma abordagem de bom senso à usabilidade.** [S.l.]: Rio de Janeiro: Ed. Alta Books, 2008. 219 p.

NIELSEN, J. **Usability Engineering.** [S.l.]: Academic Press., 1993. 358 p.

NIELSEN, J. **10 Usability Heuristics for User Interface Design**. 1994. Disponível em: <https://www.nngroup.com/articles/ten-usability-heuristics/>. Acesso em: 11 de Maio de 2019.

NODA, M. D.; HORACIO, M. P. P. **Easy Scrum**. [S.l.]: Ourinhos, SP, 2018.

PEREIRA, F. **Avaliação De Usabilidade Em Bibliotecas Digitais: Um Estudo De Caso.** [S.l.]: Dissertação (Mestrado em Ciência da Informação): Escola de Ciência da Informação da Universidade Federal de Minas Gerais, Belo Horizonte, 2011. 39 p.

PRATES, R.; BARBOSA, S. **Avaliação de interfaces de usuários – Conceitos e métodos.** 2018. Disponível em: <http://www.dimap.ufrn.br/~jair/piu/artigos/avaliacao.pdf>. Acesso em: 19 de Maio de 2019.

PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional**. 7. ed. [S.l.], 2011.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. [S.l.], 2016.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum**. 2013.

SCRUM. **Lean Kanban ou Lean Startup? Diferenças entre as metodologias ágeis.** Startupi, 2014. Disponível em: <https://startupi.com.br/2014/11/scrum-lean-kanban-ou-lean-startup-diferencas-entre-as-metodologias-ageis>. Acesso em: 19 de Agosto de 2018.

SCRUMBAN. **Kanban Tool.** 2018. Disponível em: <https://kanbantool.com/pt/scrumban-kanban-e-scrum>. Acesso em: 08 de Novembro de 2018.

SIGNIFICADOS. **Significado de Kanbam**. 2015. Disponível em: <https://www.significados.com.br/kanban>. Acesso em: 18 de Outubro de 2018.

SOARES, D. C. V. **Testes de Unidade com JUnit.** 2007. Disponível em: <https://www.devmedia.com.br/testes-de-unidade-com-junit/4637>. Acesso em: 08 de Novembro de 2018.

SOARES, L. G. **Avaliação de usabilidade, por meio de índice de satisfação dos usuários, de um software gerencial.** [S.l.]: Dissertação (Mestrado em Engenharia) - Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004. 57 p.

TEIXEIRA, F. **Análise Heurística, como fazer e os benefícios para o projeto.** 2018. Disponível em: <https://brasil.uxdesign.cc/análise-heurística-o-que-é-como-fazer-e-os-benefícios-para-o-projeto-161f3d94436b>. Acesso em: 18 de Outubro de 2018.

WASSERMANN, B. R. **Dificuldades encontradas na usabilidade web em portal de educação à distância**. [S.l.]: Universidade Regional do Nordeste do Estado do Rio Grande do Sul, 2014. 70 p.

XAMPP. **Apache, MySQL, PHP, Perl. Sobre**. 2019. Disponível em: <https://www.apachefriends.org/pt_br/about.html>. Acesso em: 11 de Maio de 2019.

Apêndices

APÊNDICE A – ABERTURA DO BANCO DE DADOS *(XAMPP CONTROL PANEL E HEIDISQL)*

Primeiro Passo

O usuário deverá extrair as pastas Banco e *ProjetoEasyScrum* do *pen drive*.

Segundo Passo

Para execução do software *Easy Scrum* será necessário primeiramente abrir os softwares *XAMPP Control Panel* e o *HeidiSQL*, importar o arquivo sql que se encontra na pasta banco.

Terceiro Passo

Em seguida, é necessário clicar sobre os botões atualizar(F5) e depois executar(F9).

APÊNDICE B – ABERTURA DOS FONTES *(NETBEANS IDE 8.2)*

Primeiro Passo

Abrir a ferramenta *Netbeans IDE*, em seguida, abrir (Ctrl+Shift+O) o *ProjetoEasyScrum.*

Segundo Passo

Encontrar a classe principal que se chama *clsPrincipal.java* e executar o sistema através dela.

Terceiro Passo

Ao executar o sistema, aparece a tela de login, onde o usuário deve inserir seu usuário e senha, ou, caso ainda não possua, deve efetuar cadastro para navegar pelo *software.*