UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DIEGO HENRIQUE PRESNER

ELSON LUIZ DOS SANTOS JUNIOR

ESTUDO SOBRE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO APLICANDO A METODOLOGIA *EXTREME PROGRAMMING* EM UMA APLICAÇÃO *WEB*

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2014

DIEGO HENRIQUE PRESNER

ELSON LUIZ DOS SANTOS JUNIOR

ESTUDO SOBRE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO APLICANDO A METODOLOGIA *EXTREME PROGRAMMING* EM UMA APLICAÇÃO *WEB*

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Willian Massami Watanabe

PONTA GROSSA

2014

TERMO DE APROVAÇÃO

ESTUDO SOBRE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO APLICANDO A METODOLOGIA EXTREME PROGRAMMING EM UMA APLICAÇÃO WEB

por

DIEGO HENRIQUE PRESNER

ELSON LUIZ DOS SANTOS JUNIOR

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 04 de junho de 2014 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Willian Massami Watanabe
Prof.º Orientador

_____
Richard Ribeiro
Membro titular

_____
Vinícius Camargo Andrade
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Dedicamos este trabalho aos nossos pais que nos ajudaram sempre, proporcionando oportunidades para a construção de um futuro melhor.

AGRADECIMENTOS

DIEGO HENRIQUE PRESNER

Agradeço primeiramente aos meus pais, pela dedicação, amor e carinho que tiveram comigo durante toda a minha vida, me apoiando sempre nos momentos difíceis e ajudando nas decisões importantes da minha vida. Amo muito vocês!!!

Ao Professor Willian, por ter compartilhado seus conhecimentos conosco, dando sugestões e todo o apoio necessário para a elaboração deste TCC.

A minha namorada Karyn, pelo amor e pelo apoio que teve comigo durante a minha jornada acadêmica, e por sempre me ajudar a ser cada vez melhor. Amo muito você!

A todos meus familiares, que contribuíram positivamente ao longo dessa caminhada.

A todos meus amigos, que diretamente ou indiretamente contribuíram de forma positiva ao longo dessa caminhada.

A mim mesmo, pela iniciativa e dedicação!

AGRADECIMENTOS

ELSON LUIZ DOS SANTOS JUNIOR

Agradeço primeiramente aos meus pais, que em todos os momentos estiveram ao meu lado, me apoiando, cobrando quando necessário, estimulando nos momentos difíceis, ajudando em decisões importantes saiba que são vocês os grandes responsáveis pela pessoa que sou hoje.

Ao Professor Willian, por ter aceitado nos orientar, compartilhando seu conhecimento conosco e nos dando as diretivas e todo apoio necessárias para o desenvolvimento desse Trabalho.

A toda minha família, que tem sido muito importante ao longo dessa caminhada.

A minha namorada Helo, pela compreensão em não poder estar em alguns momentos junto com ela, devido à dedicação necessária ao decorrer do curso, mas que esteve ao meu lado durante essa caminhada.

A todos meus amigos que diretamente ou indiretamente compartilharam comigo diversos momentos que me trouxeram até aqui.

Obrigado a todos!

RESUMO

PRESNER, Diego Henrique**;** SANTOS J. , Elson Luiz. **ESTUDO SOBRE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO APLICANDO A METODOLOGIA EXTREME PROGRAMMING EM UMA APLICAÇÃO WEB. 2014.60 f.**

Trabalho de Conclusão de Curso de Tecnologia e Análise de Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2014.

As metodologias ágeis de desenvolvimento proporcionam respostas rápidas aos ambientes de negócios mutantes, pois possuem um conjunto de práticas que as tornam mais leves em relação às metodologias tradicionais e de fácil adaptabilidade durante o desenvolvimento de um projeto. Este trabalho tem o objetivo de estudar as metodologias ágeis de desenvolvimento, propondo a aplicação da metodologia XP no desenvolvimento de uma aplicação web. Primeiramente realizou-se uma pesquisa bibliográfica detalhando os valores, processos e práticas utilizadas pelos métodos ágeis de desenvolvimento e a metodologia XP. Em seguida foi desenvolvida a aplicação utilizando processos e práticas apoiados pela metodologia XP, como a elaboração de histórias de usuário e o desenvolvimento orientado à testes (TDD). Assim foram identificados resultados durante o desenvolvimento do projeto e posteriormente puderam ser levantadas as vantagens e desvantagens da aplicação deste tipo de metodologia no desenvolvimento de uma aplicação web.

Palavras-chave: Metodologias Ágeis de Desenvolvimento. *eXtreme Programming.* Histórias de Usuário. TDD. ATDD.

ABSTRACT

PRESNER, Diego Henrique**;** SANTOS J. , Elson Luiz. **STUDY ABOUT METHODOLOGIES AGILE OF DEVELOPMENT APPLYING METHODOLOGY EXTREME PROGRAMMING IN A WEB APPLICATION. 2014.60 f.**

Conclusion Work Course Technology Analysis and Systems Development - Federal Technological University of Paraná. Ponta Grossa, 2014.

Agile development methodologies provide quick responses to changing business environments, because they have a set of practices that make them lighter compared to traditional methodologies and easy adaptability during the development of a project. This work aims to study the agile development methodologies, proposing the application of XP methodology in the developing of a web application. First we carried out a literature review detailing the values, processes and practices used by agile development methodology and XP. Then the application was developed using methods and practices supported by the XP methodology, such as the development of user stories and Test Driven Development (TDD). Thus results were identified during the development of the project .The advantages and drawbacks of this type of methodology in the development of a web application could be subsequently located.

Keywords: Agile Development. eXtreme Programming. User Stories. TDD. ATDD.

LISTA DE ILUSTRAÇÕES

Figura 1: Custo de alterações usando processos ágeis e processos tradicionais....20

Figura 2: Práticas da *XP*....23

Figura 3: Ciclo de um release em XP....25

Figura 4: Cartão CRC....26

Figura 5: Ciclo do Modelo CRC....27

Figura 6: Ciclo de um Desenvolvimento Orientado à Testes (TDD)....30

Figura 7: Fluxograma de desenvolvimento do sistema de publicação de conteúdo....35

Figura 8: Fluxograma MVC....39

Figura 9: Comando utilizado para a execução do *selenium-server*....40

Figura 10: S*elenium-server em execução*....40

Figura 11: Teste de aceitação escrito utilizando linguagem natural....41

Figura 12: Repositório criado no *GitHub*....42

Figura 13: *View* novo_usuario.php....43

Figura 14: *Controller* cadastro.php....44

Figura 15: *Model* inserir.php....45

Figura 16: *Interface* - Página Principal / *Login*....46

Figura 17: Falha no teste de *Login*....47

Figura 18: Teste de *Login* aprovado com sucesso....48

Figura 19: Interface - Cadastro de usuário....49

Figura 20: Teste de cadastro aprovado com sucesso....50

Figura 21: Interface - Perfil de usuário....51

Figura 22: Teste de perfil de usuário aprovado com sucesso....52

Figura 23: Teste de publicação - Cenários 1 e 2....53

Figura 24: Teste de publicação - Cenário 3....54

LISTA DE QUADROS

Quadro 1- Princípios dos métodos ágeis.....18
Quadro 2 - Princípios de agilidade.....20
Quadro 3: Valores da XP.....22
Quadro 4 - Tarefa 1: Implementação página inicial.....36
Quadro 5 - Tarefa 2: Implementação do *login* de usuário.....36
Quadro 6 - Tarefa 3: Implementação do cadastro de usuário.....37
Quadro 7 - Tarefa 4: Implementação do perfil pessoal do usuário.....37
Quadro 8 - Tarefa 5: Implementação da publicação de conteúdo.....38

LISTA DE ABREVIAÇÕES E SIGLAS

XP – *eXtreme Programming*

FDD - *Feature Driven Development*

CRC - (Classe, responsabilidade ecolaboração)

TDD -*Test Driven Development*

ATDD - *AcceptanceTest Driven Development*

POO- Paradigma da orientação à objetos

MVC - *Model-View-Controller*

CRUD - (*Create, Read, Update e Delete*)

HTTP - *Hyper Text Transport Protocol*

ERP - *Enterprise Resource Planning*

OC - Ordem de Carregamento.

SUMÁRIO

1 INTRODUÇÃO 12

1.1 OBJETIVOS 13

1.2 JUSTIFICATIVA 14

1.3 ESTRUTURA DO TRABALHO 14

2 FUNDAMENTAÇÃO TEÓRICA 16

2.1 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO 16

2.1.1 Agilidade e Custos 18

2.2 METODOLOGIA XP 22

2.2.1 Planejamento 24

2.2.2 Projeto 26

2.2.3 Codificação 27

2.2.4 Testes 29

2.3 TRABALHOS CORRELATOS 32

3 METODOLOGIA E DESENVOLVIMENTO 35

3.1 FERRAMENTAS UTILIZADAS 40

4 RESULTADOS 46

4.1 TESTES PRIMEIRO RELEASE 46

4.2 TESTES SEGUNDO RELEASE 49

4.3 TESTES TERCEIRO RELEASE 51

5 CONCLUSÃO 56

5.1 TRABALHOS FUTUROS 56

6 REFERENCIAS 58

1 INTRODUÇÃO

Atualmente a tecnologia está presente em diversas áreas de atuações profissionais sendo utilizada para facilitar a execução de diversos tipos de atividades. Como consequência do avanço tecnológico, surgem novas necessidades em diversas áreas de mercado em que as pessoas precisam de uma solução para resolver problemas ou até mesmo para facilitar a execução de seus trabalhos, as quais podem ser dadas através de um *software* desenvolvido para atender tais necessidades.

Segundo Pressman (2006), o *software* não é fabricado no sentido clássico, mas sim desenvolvido, passando por um processo de engenharia, pois é um elemento que possui características lógicas e não físicas como o *hardware*.

A engenharia de *software* é composta por diversas metodologias, algumas sendo consideradas tradicionais e outras inovadoras, como por exemplo, a Programação Extrema (XP, em inglês *eXtreme Programming*).

Pressman (2011) descreve que a metodologia XP é o método com maior índice de utilizações das metodologias ágeis de desenvolvimento, os quais priorizam a satisfação do cliente e a entrega incremental, possuindo os objetivos de desenvolver sistemas rápidos e corretos, priorizando principalmente o desenvolvimento do *software* sobre a análise e o projeto do sistema.

Por esta metodologia gerar conflitos entre os engenheiros de *software* mais conservadores e os mais radicais, e por prometer redução de custos durante o desenvolvimento e a entrega dos incrementos dentro dos prazos determinados, este trabalho teve como objetivo apresentar uma visão aprofundada e prática sobre o assunto para melhor analisar a XP, construindo os aspectos positivos e negativos de seus processos no desenvolvimento de uma aplicação *web*.

1.1 OBJETIVOS

Para melhor delineamento do trabalho, foram definidos os seguintes objetivos:

1.1.1 Objetivo Geral

Estudar e desenvolver uma aplicação utilizando a metodologia XP;

1.1.2 Objetivos Específicos

- Estudar a metodologia ágil XP e como são conduzidos seus processos para o desenvolvimento;
- Apresentar definições da metodologia;
- Levantar uma determinada situação empresarial;
- Aplicar os processos do XP sobre essa determinada situação e desenvolver a aplicação;
- Descrever vantagens e dificuldades durante o estudo.

1.2 JUSTIFICATIVA

O mercado atual vive em constante transformação, e elas ocorrem de forma rápida e algumas vezes não são determinadas antecipadamente. E quando se trata de software isso também acaba ocorrendo com frequência por estarem diretamente integrados com os processos diários de um negócio ou uma empresa. Essas adaptações devem ser realizadas com rapidez e sem interromper os outros processos.

Com esse intuito foram propostas metodologias ágeis para o desenvolvimento, por exemplo, a metodologia XP, que segundo Pressman (2006), é a abordagem com maior índice de utilização para desenvolvimento de *software*.

Partindo deste pressuposto, o presente trabalho propõe o estudo dos conceitos e métodos da metodologia XP no desenvolvimento de uma aplicação *web*.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está dividido em 5 capítulos, sendo o primeiro destinado à descrição dos objetivos e a motivação dos mesmos.

O segundo capítulo apresenta a fundamentação teórica, onde são encontradas informações e conceitos sobre as metodologias ágeis de desenvolvimento e metodologia XP.

O terceiro capítulo aborda as características específicas e métodos envolvidos nos processos de planejamento, projeto, codificação e testes, que são compreendidos pela metodologia XP, assim como, a descrição da implementação da aplicação e utilização dos sistemas de gerenciamento de configurações (*GIT*) e de desenvolvimento de testes (*Pyccuracy*).

No quarto capítulo são apresentados os resultados obtidos com o desenvolvimento deste trabalho, discutindo as vantagens e desvantagens da aplicação da metodologia XP encontradas no desenvolvimento da aplicação *web*.

E, por fim, o quinto capítulo apresenta a conclusão do trabalho e indicações possíveis para trabalhos que poderão ser desenvolvidos futuramente.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os conceitos relacionados às Metodologias Ágeis de Desenvolvimento e a Metodologia XP. A seção 2.1 apresenta as metodologias ágeis de desenvolvimento. A seção 2.2 discorre sobre a metodologia XP. E, por fim, a seção 2.3 exibe os trabalhos correlatos.

2.1 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO

As metodologias ágeis de desenvolvimento, também conhecidas por métodos ágeis, são um conjunto de processos projetados para o desenvolvimento rápido de um *software*. Estes processos costumam ser iterativos, intercalando as atividades de especificação, projeto, desenvolvimento e teste (Sommerville, 2007).

Os métodos ágeis de desenvolvimento de *software* surgiram quando dezessete especialistas elaboraram um conjunto de regras que contemplam as melhores maneiras de se desenvolver o *software*, assim formando a *Agile Alliance* e estabelecendo o *Agile Manifesto*, no ano de 2001 (Beck, 2001). Estes especialistas são os criadores de métodos como:

- ***eXtreme Programming* (XP):** é uma metodologia ágil para equipes pequenas e médias e que irão desenvolver *software* com requisitos vagos e em constante mudança;
- ***Scrum*:** processo de desenvolvimento iterativo e incremental para o gerenciamento de projetos e desenvolvimento de *software* ágil;
- ***Feature Driven Development* (FDD):** é uma metodologia ágil de desenvolvimento que combina as práticas de gerenciamento ágil com abordagens à engenharia de *software* orientada a objetos;

Pressman (2006) afirma que a metodologia XP, é a metodologia ágil de desenvolvimento mais utilizada pelos desenvolvedores de *software*.

Apesar das metodologias ágeis possuírem métodos diferentes, fundamentalmente pregam o desenvolvimento leve, iterativo e incremental (Ramos, 2013).

Kent Beck (2001) e os outros desenvolvedores de software assinaram o *Agile Manifesto*, propondo melhores modos de desenvolvimento de *software*, procurando valorizar os seguintes itens:

Indivíduos e iterações em vez de processos e ferramentas;

Software funcional em vez de documentação abrangente;

Colaboração do cliente em vez de negociações por contrato;

Resposta a modificações em vez de seguir um plano;

Beck ressalta que os itens que se encontram à direita, continuam sendo importantes, mas os itens que estão à esquerda são os que possuem maior importância durante o desenvolvimento do *software* (Pressman, 2011).

Na década de 80 e início da década de 90, havia uma visão geral que a melhor maneira de garantir a qualidade de um *software*, era por meio de um minucioso planejamento de projeto utilizando métodos de análise e desenvolvimento apoiado por ferramentas *CASE* e gerenciados por processos rígidos de desenvolvimento de *software*, técnicas que tornavam tais metodologias inadequadas para o desenvolvimento de pequenos projetos (Sommerville, 2007).

Segundo Sommerville (2007), os desenvolvedores propuseram métodos ágeis de desenvolvimento, devido à insatisfação que possuíam em utilizar as metodologias inadequadas no desenvolvimento de pequenos projetos, pois o tempo gasto com o desenvolvimento era menor que o tempo gasto com a elaboração do projeto e documentação do sistema.

Podem existir diferentes abordagens, porém, existem algumas características que são fundamentais em todas elas:

- Os processos de especificação, projeto e implementação são executados paralelamente. Os requisitos especificados são os considerados mais importantes e a documentação de projeto é reduzida ou gerada automaticamente com o auxilio de um ambiente de programação utilizado para o desenvolvimento do sistema;

- O desenvolvimento do sistema é dividido em incrementos, estes sendo avaliados pelos *stackholders* e clientes, os quais podem propor alterações que devem ser acrescentadas no desenvolvimento de funções posteriores;

- O uso de ferramentas de desenvolvimento interativas para criar a interface com o usuário, as quais possibilitam a criação rápida do projeto de interface através de desenhos e inserção de ícones;

Sommerville (2007) descreve os princípios que são compartilhados pelos métodos ágeis, os quais são mostrados no Quadro 1 abaixo:

Princípio	Descrição
Envolvimento do Cliente	Clientes devem ser profundamente envolvidos no desenvolvimento do sistema. Possuem a função de fornecer e especificar novos requisitos e avaliar as iterações do sistema.
Entrega incremental	O *software* é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada um deles.
Pessoa, não processo	Os membros da equipe devem trabalhar sem auxílio de processos prescritivos, desenvolvendo e ganhando reconhecimento por suas habilidades.
Aceite as mudanças	Projetar o sistema de forma que seja fácil incluir novas mudanças.
Mantenha a simplicidade	Procurar manter a simplicidade na hora de desenvolver o *software* e sempre que possível eliminar as complexidades.

Quadro 1- Princípios dos métodos ágeis.

Fonte: Sommerville (2007, p.263). Adaptado pelos autores.

2.1.1 Agilidade e Custos

Em um ambiente empresarial, a agilidade é definida como capacidade de responder às mudanças em um enquadramento de dificuldades, se adaptando da melhor forma possível a essas mudanças impostas. As empresas devem ser ágeis para reagirem às mudanças e se ajustarem com rapidez a essas circunstâncias, e

para isso é fundamental que existam objetivos bem definidos e de conhecimento de todos os membros da organização (Alves, 2012).

No contexto da engenharia de software, Jacobson (2002) apresenta a seguinte discussão sobre agilidade:

> Atualmente, agilidade tornou-se a palavra da moda quando se descreve um moderno processo de *software*. Todo mundo é ágil. Uma equipe ágil é aquela rápida e capaz de responder apropriadamente a mudanças. Mudanças têm muito a ver com desenvolvimento de *software*. Mudanças no *software* que está sendo criado, mudanças nos membros da equipe, mudanças devido a novas tecnologias, mudanças de todos os tipos que poderão ter um impacto no produto que está em construção ou no projeto que cria o produto. Suporte para mudanças deve ser incorporado em tudo o que fazemos em *software*, algo que abraçamos porque é o coração e a alma do *software*. Uma equipe ágil reconhece que o *software* é desenvolvido por indivíduos trabalhando em equipes e que as habilidades dessas pessoas, suas capacidades em colaborar estão no cerne do sucesso do projeto. (Jacobson, 2002)

Segundo Pressman (2011), a agilidade não é apenas uma resposta às mudanças que ocorrem durante o desenvolvimento de um projeto, ela incentiva a melhora da comunicação, ou seja, tornando-a mais fácil e abrangente entre os membros da equipe; enfatiza a entrega rápida do incremento útil do sistema; diminui a importância de artefatos intermediários; acolhe o cliente como um membro da equipe eliminando as possíveis barreiras existentes entre ele e os outros membros da equipe; reconhece os limites existentes que um planejamento pode ter e que o plano de projeto deve ser flexível.

As mudanças em um projeto utilizando processos convencionais acarretam alterações nos custos de desenvolvimento do mesmo, os quais podem crescer rapidamente trazendo efeitos colaterais indesejados. Beck (2000) argumenta que um processo ágil bem elaborado "achata" o custo da curva de mudança, permitindo que a equipe responsável pelo projeto assimile tais mudanças sem um impacto significativo nos custos. A linha verde na Figura 1 demonstra isso.

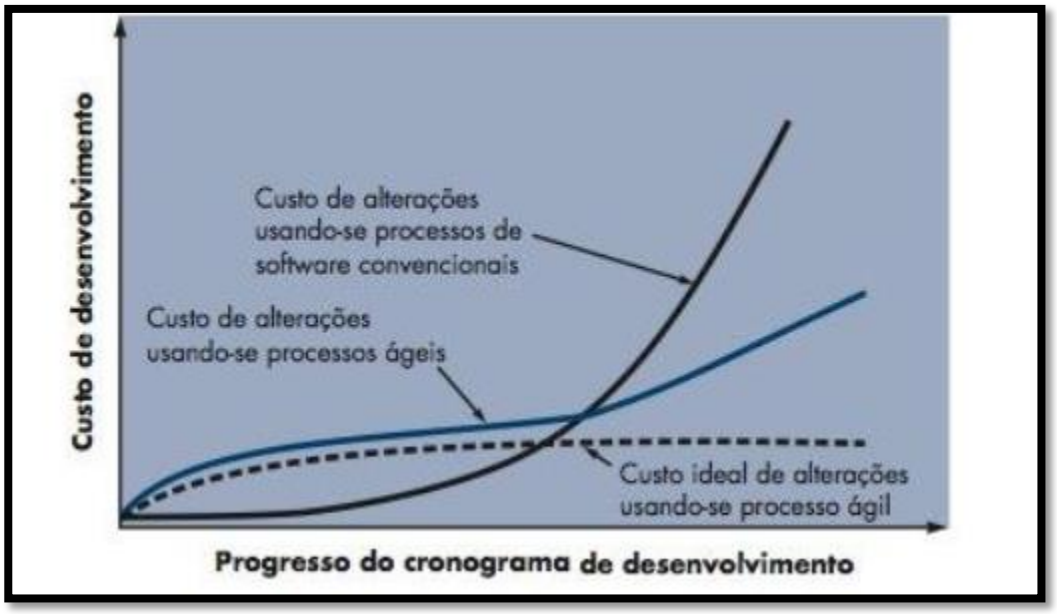

Figura 1: Custo de alterações usando processos ágeis e processos tradicionais.

Fonte: Pressman (2011, p.83).

Um processo ágil deve ser capaz de administrar a imprevisibilidade, sendo capaz de se adaptar às diversas mudanças que podem ocorrer durante o desenvolvimento do projeto, e essa adaptação deve ser feita de forma incremental, ou seja, a equipe de desenvolvimento necessita do *feedback* do cliente para que as devidas adaptações sejam realizadas corretamente (Pressman, 2011).

Os *feedbacks* podem ser dados através de partes funcionais do sistema (incrementos), as quais são entregues em curtos períodos de tempo ao cliente, possibilitando que as adaptações sejam feitas seguindo o mesmo ritmo das mudanças que podem surgir no decorrer do desenvolvimento (Pressman, 2011).

A *Agile Alliance* propôs doze princípios de agilidade que devem ser utilizados no desenvolvimento de software, os quais são citados no Quadro 2 abaixo:

Princípios de agilidade
(continua)
1. A maior prioridade é satisfazer o cliente por meio de entrega adiantada e contínua do software valioso.
2. Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento. Os

(conclusão)
processos ágeis se aproveitam das mudanças como uma vantagem competitiva na relação com o cliente.
3. Entregue o software em funcionamento frequentemente, de algumas semanas para alguns meses, dando preferência a intervalos mais curtos.
4. O pessoal comercial e os desenvolvedores devem trabalhar em conjunto diariamente ao longo de todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e apoio necessários e confie neles para ter o trabalho feito.
6. O método mais eficiente e efetivo de transmitir informações para e dentro de uma equipe de desenvolvimento é uma conversa aberta, de forma presencial.
7. Software em funcionamento é a principal medida de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os proponentes, desenvolvedores e usuários devem estar capacitados para manter um ritmo constante indefinidamente.
9. Atenção contínua para com a excelência técnica e para com bons projetos aumenta a agilidade.
10. Simplicidade - a arte de maximizar o volume de trabalho não efetuado - é essencial.
11. As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto-organizam.
12. Com intervalos regulares, a equipe se avalia para ver como tornar-se mais eficiente, então sintoniza e ajusta seu comportamento de acordo.

Quadro 2 - Princípios de agilidade

Fonte: Pressman (2011, p.84). Adaptado pelos autores.

2.2 METODOLOGIA XP

A XP é uma abordagem amplamente utilizada, que segue os princípios propostos pelos métodos ágeis. A XP é uma metodologia que ajuda a desenvolver sistemas de melhor qualidade em um menor período de tempo, reduzindo os custos e transtornos causados às pessoas envolvidas no projeto (Castro, 2006).

Beck (2004) define um conjunto de cinco valores que embasam o trabalho realizado como parte da XP, os quais são mostrados no Quadro3 abaixo:

Valores	Descrição
Comunicação	- Colaboração estreita da equipe; - Estabelecimento de metáforas;
Simplicidade	- Projetar para as necessidades imediatas; - Facilidade de implementação;
Feedback	- Principais fontes: Software implementado, cliente e equipe de desenvolvimento; - Levantamento de novas necessidades surgidas no decorrer do projeto;
Coragem	- Disciplina para aprender a projetar para o "hoje"; - Reconhecimento de que as necessidades futuras podem mudar drasticamente;
Respeito	- Respeito entre todos os membros envolvidos no projeto e com o próprio software; - Entrega com sucesso de incrementos de software aumenta o respeito da equipe pelo processo XP;

Quadro3 - Valores da XP.

Fonte: Beck (2004) e Pressman (2011, p.87). Adaptado pelos autores.

Recomenda-se o uso da metodologia XP no desenvolvimento de projetos em que os requisitos sejam incertos e que possam mudar constantemente e em uma equipe de desenvolvimento de no máximo doze pessoas, possibilitando o desenvolvimento iterativo e a utilização do paradigma da orientação a objetos (POO) na construção do sistema (Ramos, 2013).

A XP é composta por um conjunto de regras e práticas constantes, possuindo quatro atividades metodológicas (Figura 2), as quais serão explicadas nas subseções a seguir.

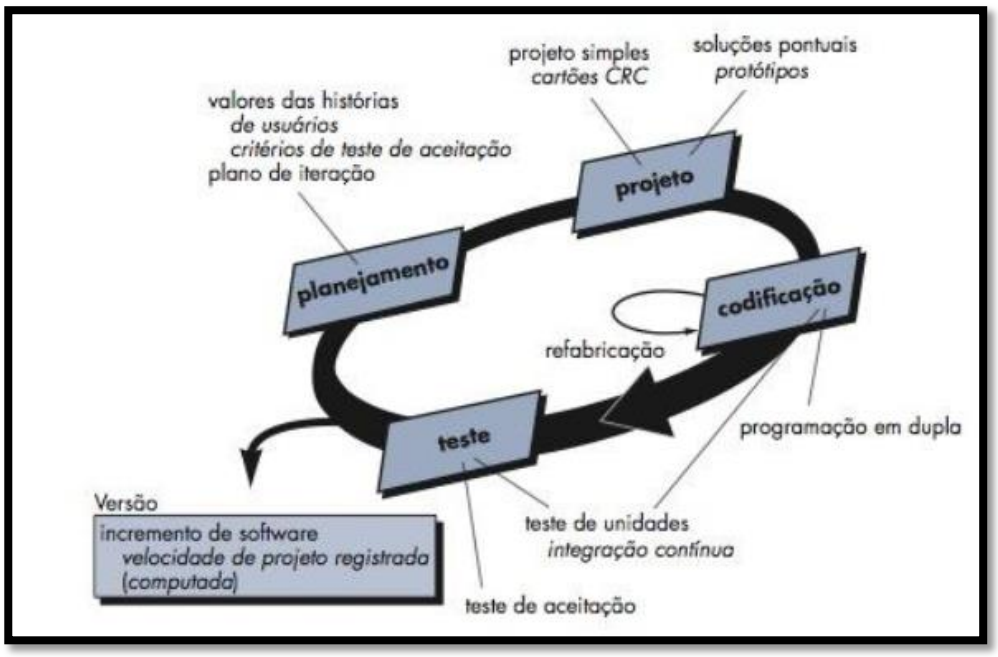

Figura 2: Práticas da XP.

Fonte: Pressman (2011, p.88)

2.2.1 Planejamento

A etapa de planejamento consiste nas atividades que serão utilizadas para o levantamento dos requisitos do software. Ela se inicia com a atividade de "ouvir" o cliente, possibilitando o levantamento dos requisitos do sistema e levando os membros da equipe ao entendimento sobre qual ambiente de negócios o *software* que será desenvolvido vai atuar (Pressman, 2011).

Para que essa atividade ocorra de forma eficaz, é necessário que o cliente seja envolvido intimamente no desenvolvimento das atividades, tornando-se um membro da equipe de desenvolvimento sempre presente (Sommerville, 2007).

Após "ouvir" o cliente, a equipe de desenvolvimento dá inicio à criação das histórias de usuário, onde são descritos os resultados, as características e as funcionalidades requisitadas para o *software* a ser construído. O cliente é o responsável pela criação de cada história, as quais ele atribuí um grau de prioridade, ou seja, a história que receber o grau de prioridade maior é a que deve ser elaborada primeiro (Pressman, 2011). Normalmente as histórias que recebem a maior prioridade são as que podem ser usadas imediatamente para proporcionar apoio útil ao negócio (Sommerville, 2007).

A equipe XP irá analisar cada história e atribuir os custos de desenvolvimento em cada uma, os quais são medidos em semanas de desenvolvimento e se uma história requerer mais de três semanas, será solicitado ao cliente que ele a divida em histórias menores, as quais receberão valores e custos novamente (Pressman, 2011).

O cliente e os desenvolvedores, juntamente, agrupam as histórias e decidem qual será a próxima história, ou incremento de software, a ser desenvolvida pela equipe XP. Eles devem chegar a uma concordância mútua, podendo assim definir quais serão as histórias incluídas, datas de entrega e outras questões levantadas durante o desenvolvimento do projeto.

A equipe XP ordena as histórias em uma destas três formas:

1- Todas serão implementadas imediatamente;

2- As histórias de maior valor serão deslocadas para cima do cronograma e implementadas primeiro;

3- As histórias de maior risco serão deslocadas para cima no cronograma e implementadas primeiro;

A metodologia XP permite que novas histórias sejam escritas a qualquer momento e conforme o projeto prossegue, o cliente pode acrescentar novas histórias, mudar o grau de prioridade, dividi-las ou eliminá-las (Pressman, 2011).

O desenvolvimento incremental é uma das práticas apoiadas pela metodologia XP, onde são liberados pequenos [1] *releases* do sistema frequentemente, abordando os requisitos baseados nas histórias do cliente.

A Figura 3 abaixo ilustra o ciclo de um *release* em XP para produzir um incremento do sistema:

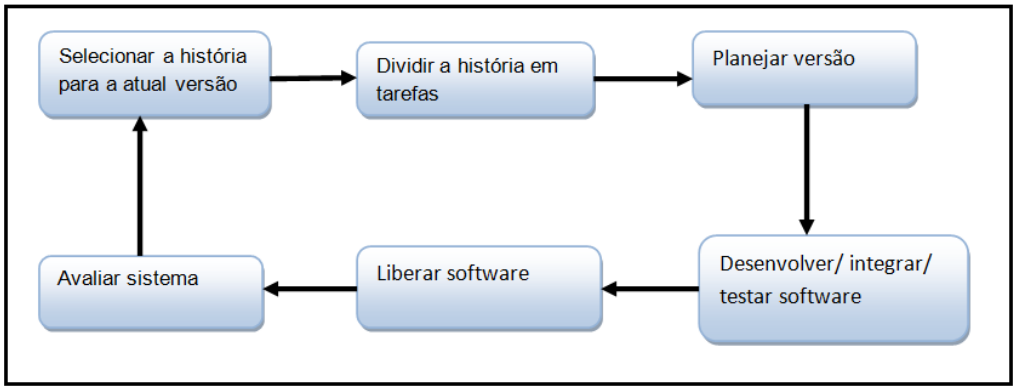

Figura 3: Ciclo de um release em XP.

Fonte: Sommerville (2007, p.264). Adaptado pelos autores.

[1] Release: palavra em inglês, quando usada como substantivo significa liberação ou lançamento.

2.2.2 Projeto

A metodologia XP segue rigorosamente o princípio KISS (*keep it simple, stupid*), onde o projeto é desenvolvido com foco na simplicidade, evitando uma representação complexa do mesmo. Um sistema planejado de forma simples pode ser rapidamente adaptado às mudanças, proporcionando redução de custos e tempo durante seu desenvolvimento, sem deixar de possuir recursos para atender às necessidades do cliente (Pressman, 2011).

A XP encoraja o uso da refatoração, a qual pode ser definida como um processo de alteração do software que não afeta o comportamento externo do código, mas aprimora a sua estrutura interna. Utilizando a refatoração é possível simplificar o projeto interno e diminuir as chances de introduzir bugs no sistema, ou seja, ao se refatorar o código equipe de desenvolvimento estará aperfeiçoando o projeto de codificação após esta ter sido feita (Fowler, 2000).

São utilizados na XP como mecanismo para entender o software em uma perspectiva orientada a objetos os cartões CRC (classe-responsabilidade-colaborador). São apresentadas no cartão CRC o nome da classe, os atributos e métodos (responsabilidades), e as associações (colaboradores). A figura 4 ilustra um exemplo de cartão CRC:

	Carro	
	Responsabilidade	**Colaboração**
atributos	Possuir uma cor	Proprietário
atributos	Possuir placa	Condutor
atributos	Possuir modelo	
métodos	Acelerar	
métodos	Frear	

Figura 4: Cartão CRC.

Fonte: Beck e Cunningham (1989).Adaptado pelos autores.

Estes cartões são utilizados para identificar e organizar as classes orientadas a objetos que são relevantes para o incremento do software, sendo considerado o único artefato produzido como parte do processo XP (Pressman, 2011). O diagrama a seguir (Figura 5), apresenta o ciclo de funcionamento do modelo CRC:

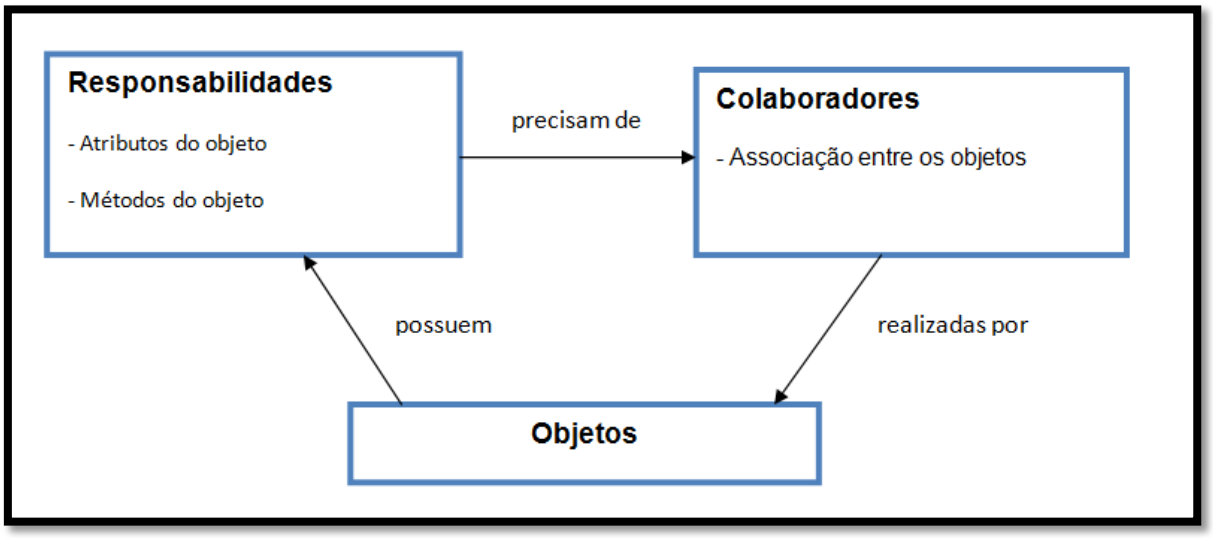

Figura 5: Ciclo do Modelo CRC.

Fonte: Beck e Cunningham (1989). Adaptado pelos autores.

A fase de projeto na XP é vista como algo transitório por produzir poucos artefatos, se algum, além dos cartões CRC, e deve sofrer modificações contínuas conforme o desenvolvimento prossegue. A refabricação é responsável por controlar tais modificações, sugerindo pequenas mudanças de projeto, as quais são "capazes de melhorá-lo radicalmente" (Fowler, 2000).

2.2.3 Codificação

A programação em pares é uma das técnicas inovadoras utilizadas pela metodologia XP. Esta técnica consiste no trabalho em duplas, em uma mesma estação de trabalho para criar o código para uma história, possibilitando a resolução de problemas em tempo real e garantia de qualidade, pois o código é revisto à medida que é implementado. Cada um dos programadores recebe um papel diferente, enquanto um dos programadores digita o código, o outro analisa o que foi

construído, buscando erros e possíveis melhorias. Beck (2004) menciona que "é mais provável a ocorrência de erros, reprojeto e o desrespeito às práticas quando as pessoas programam sozinhas, principalmente sob pressão".

O desenvolvimento não envolve sempre a mesma dupla de programadores, pois o intuito da programação em pares é que os pares sejam criados dinamicamente, possibilitando aos membros trabalharem com outros diferentes em uma programação em pares no decorrer do projeto, aumentando a união e disseminação das experiências da equipe de desenvolvimento (Sommerville, 2007).

A prática da programação em pares contribui para a comunicação da equipe, a qual é um dos valores fundamentais da XP, pois os pares estarão se comunicando constantemente com o objetivo de encontrar a melhor forma possível para resolver um determinado problema que possa aparecer no decorrer do projeto (Santos, 2013). Pressman (2006) menciona que “isso fornece um mecanismo de solução de problemas em tempo real e de garantia de qualidade em tempo real. Também mantém os desenvolvedores focados no problema em mãos”.

O código produzido em um projeto XP é de propriedade coletiva, qualquer membro da equipe tem a liberdade para editar e melhorar qualquer trecho de código do projeto. A propriedade coletiva permite que todos os membros da equipe conheçam o código do sistema e possam fazer modificações caso sejam necessárias. Os membros da equipe possuem a liberdade para trabalhar desta forma, pois o código produzido é orientado a testes, o que significa que cada classe e cada método possuem seus testes automatizados, garantindo o funcionamento do sistema (Santos, 2013). Quanto a isso, Beck (2004) cita que "escrevendo e executando testes diminuem-se as chances de estragar algo acidentalmente". O uso da propriedade coletiva minimiza o impacto causado à equipe caso um membro da equipe venha a faltar, pois os demais estarão habituados com o código e poderão realizar as alterações necessárias (Beck, 2004).

É importante que todo o código produzido em um projeto XP siga um padrão, seguindo um conjunto de regras, como nomenclatura de classes, métodos, variáveis e formatação do código.

> Se você terá todos esses programadores trabalhando em tantas partes diferentes do sistema, trocando de duplas várias vezes ao dia e refatorando os códigos uns dos outros constantemente, você não pode ter conjuntos diferentes de práticas de codificação. Com um pouco de prática, será impossível dizer qual pessoa do time escreveu que código. (BECK, 2004 p.72)

A adoção de um padrão de codificação mantém o código coletivo, ajuda na comunicação da equipe de projeto facilitando a programação em pares e proporciona ganho de tempo na produção de código.

> Você pode trabalhar com velocidade total porque não precisa reformatar o código durante o trabalho, nem precisa estar constantemente alternando deum estilo para outro ou tentando entender a formação e o estilo, bem como o próprio código. (ASTELS, 2002 p. 10)

2.2.4 Testes

Uma das principais diferenças da XP em relação às metodologias convencionais é a maneira de como é testada a aplicação. Na XP, a ênfase dada à fase de testes é maior, pois o desenvolvimento iterativo não gera uma documentação detalhada do sistema, impossibilitando que uma equipe externa desenvolva os testes do software (Sommerville, 2007).

Segundo Sommerville (2007) as principais características de teste na XP são:

Desenvolvimento *test-first;*

Desenvolvimento incremental de teste a partir de cenários;

Envolvimento do usuário no desenvolvimento e validação de testes;

O uso de ferramentas de teste automatizadas;

A XP possui como uma de suas importantes inovações o desenvolvimento de *test-first*, a qual busca definir a interface tanto como especificar as funcionalidades que serão desenvolvidas, reduzindo a probabilidade de erros em novos incrementos do software. O desenvolvimento *test-first* propõe que os testes sejam escritos antes

do código, sendo um componente executável, possibilitando sua execução logo após uma tarefa ser implementada (Sommerville, 2007).

A XP utiliza a técnica de Desenvolvimento Orientado a Testes (TDD, em inglês *Test Driven Development*) no decorrer de todo o projeto, que é conhecida por criar um mecanismo de testes automatizados antes de desenvolver as funcionalidades de cada história envolvida no sistema.

O fluxograma a seguir (Figura 6) mostra como é o ciclo de um TDD:

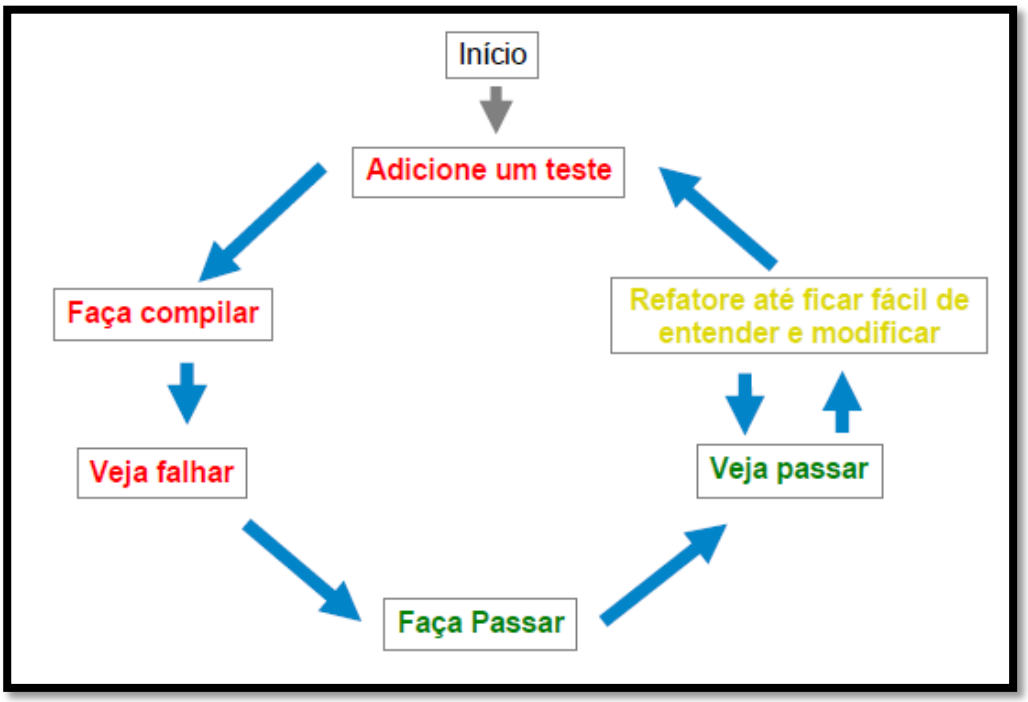

Figura 6: Ciclo de um Desenvolvimento Orientado à Testes (TDD).
Fonte: Castro (2006, p. 74).

Teles (2005) afirma que “a única forma de se mover rapidamente e com confiança é tendo uma rede de testes, tanto de unidade, quanto de aceitação”.

Os testes de unidade são responsáveis por monitorar automaticamente métodos de classes que são utilizadas para implementar as funcionalidades do sistema. A equipe de desenvolvimento divide cada cenário em tarefas, projetando

um teste unitário para cada uma delas, o qual irá verificar a implementação descrita na tarefa selecionada (Sommerville, 2007).

Os testes de aceitação, também denominados testes de cliente, são especificados pelo cliente mantendo o foco nos requisitos e funcionalidades totais do sistema. Eles são criados a partir das histórias de usuários e implementados como parte do software (Pressman, 2011).

Esses testes ajudam na documentação e monitoração do sistema em alto nível, proporcionando que as funcionalidades desenvolvidas sejam revistas pelo cliente, ou seja, a cada nova funcionalidade completa, o cliente realiza um teste de aceitação, verificando se o sistema está sendo desenvolvido como o esperado e do ponto de vista do usuário final.

Teles (2006) menciona o seguinte a respeito dos testes de aceitação:

> Podemos entender o teste de aceitação como sendo um roteiro que tem vinculado a ele um conjunto de respostas esperadas. Dada uma estória, é importante escrever um passo a passo que o usuário deverá executar. Para cada ação, existe uma resposta esperada que esta escrita no passo a passo. O usuário deve executar as ações e comparar os resultados com aqueles esperados. Se o sistema apresenta um resultado diferente é porque existe um erro nele ou o próprio teste está incorreto. (TELES, 2006 p. 138)

Os testes, tanto de unidade quanto de aceitação, garantem que cada nova funcionalidade desenvolvida esteja correta e que o sistema está sendo desenvolvido de acordo com as exigências do cliente.

2.3 TRABALHOS CORRELATOS

Ramos (2013) atuava como desenvolvedor e responsável pelo projeto dentro de uma empresa de armazenamento e devolução de combustível, a UNIBRASPE. A empresa possuía um ERP que abordava todos os departamentos, porém o controle de estoque de terceiros e faturamento não atendia as necessidades da empresa e precisava ser desenvolvido em separado. Sendo assim a empresa optou por deixar o seu próprio departamento de informática responsável pelo projeto. Era um departamento recém-formado e não contava com muitos profissionais.

A equipe desenvolvedora foi transferida para a mesma sala que os responsáveis pela supervisão, administração e faturamento após a decisão da utilização da metodologia XP.

Para armazenar os produtos, a distribuidora precisa emitir uma nota fiscal de armazenagem, já para retirar o combustível a distribuidora emite uma ordem de carregamento (OC). Esta OC indica se a distribuidora possui estoque suficiente, dando continuidade ao procedimento de carregamento. Na saída do veículo, o sistema deve emitir uma nota fiscal de devolução de armazenagem, encerrando assim o processo.

Na primeira reunião foram definidas algumas rotinas para chegar à emissão de uma nota fiscal, como por exemplo:

1. Cadastro de clientes;

2. Cadastro de motoristas;

3. Cadastro de veículos;

4. Cadastro de produtos;

Após o desenvolvimento dessas rotinas, foi entregue a primeira versão do sistema aos usuários que iniciaram imediatamente o uso do sistema, apontando melhorias e solicitando novos recursos.

Dentre os benefícios alcançados com o uso da metodologia XP, o autor destaca alguns pontos importantes como:

- A proximidade com o cliente trouxe vários benefícios, obtendo um melhor entendimento das necessidades e dificuldades enfrentadas diariamente, com isso surgia a proposta de soluções e mudanças;

- Satisfação dos usuários que recebiam constantemente uma nova versão do software, contendo as funcionalidades priorizadas pelos mesmos e que agregavam maior valor ao negócio;

- Tranquilidade por parte do cliente, sabendo que os desenvolvedores aceitavam as constantes mudanças no projeto uma vez que sua ferramenta se adaptava rapidamente as mudanças e requisitos do mercado;

- Agilidade da equipe de desenvolvimento em responder as solicitações dos usuários com um ambiente favorável para todos.

A equipe de desenvolvimento não tinha experiência em TDD e optaram por fazer testes funcionais, alguns testes automatizados foram criados, principalmente para validação de rotinas críticas, análise automática de logs ou registros em bancos de dados.

Silva et al (2004), desenvolveram uma ferramenta para modelagem de requisitos através da descrição dos termos do Léxico e de Cenários, como se trata de um curso, primeiramente os alunos precisaram aprender essas duas técnicas para desenvolver. Consequentemente utilizaram a metodologia XP para o desenvolvimento.

A aplicação desenvolvida oferece as funcionalidades de edição de léxico e cenário, gerenciamento de projetos e usuários. Para desenvolvê-la os alunos tiveram a necessidade de tratar as diversas propriedades que cercam o projeto, visto que os alunos têm que verificar e validar a aplicação que estão construindo. Os autores afirmam que a verificação ajudou a planejar e editar corretamente, a validação ajudou a levantar os requisitos de edição de maneira a oferecer um maior apoio a estas atividades. Além disto, foi necessário elaborar uma base de testes para validar a aplicação.

Os testes foram realizados constantemente entre os pares de programação e a cada versão foram realizados também pelo cliente. Nesse projeto em questão o cliente é o professor, este avaliou os limites do software e se estava dentro do nível de aceitação.

Os alunos se tornaram mais participativos durante o aprendizado na disciplina e críticos para distinguir os diversos processos de desenvolvimento existentes.

O desenvolvimento ocorreu através de pequenas iterações. A obtenção dos dados, modelagem e análise dos requisitos ocorreu em diversos momentos visto que o cliente estava sempre disponível. O projeto foi realizado através do detalhamento dos cenários e alguns rascunhos da arquitetura de partes críticas do *software* e do banco de dados. E, por fim, a implementação que é a atividade central do processo XP.

Muitas dificuldades encontradas pelos alunos para desenvolver o software sob as condições impostas os ajudaram a identificar falhas e a discutir o que seria mais apropriado em tais situações e a perceber o porquê dos diversos aspectos estudados pela disciplina de engenharia de *software*. Foram destacadas também as dificuldades como:

- A infraestrutura obrigou os alunos a montarem seus próprios servidores e adaptar uma forma de controle de versões para a aplicação e as modificações realizadas;

- Iterações curtas dificultam o desenvolvimento de processos mais complexos que requerem um pouco mais de experiência;

- Falta de documentação por parte dos envolvidos no projeto, causando dificuldades quando é necessário um membro consultar uma rotina desenvolvida por outro membro sem a documentação adequada.

3 METODOLOGIA E DESENVOLVIMENTO

Neste trabalho foi aplicada a metodologia de desenvolvimento ágil XP, para o desenvolvimento da aplicação web, com o intuito de levantar as vantagens e desvantagens de forma aplicada no desenvolvimento de software.

O sistema desenvolvido neste trabalho consiste em uma aplicação utilizada para a publicação de conteúdo, onde os usuários poderão possuir suas contas pessoais e publicar textos em suas respectivas páginas.

As atividades compreendidas neste trabalho (Figura 7) são aplicadas em cada um dos *releases* do sistema desenvolvido. As etapas aplicadas em cada *release* são: (a) criação das histórias de usuário; (b) divisão das histórias em tarefas; (c) desenvolvimento dos testes; (d) codificação da aplicação; (e) execução dos testes; e (f) refatoração da aplicação.

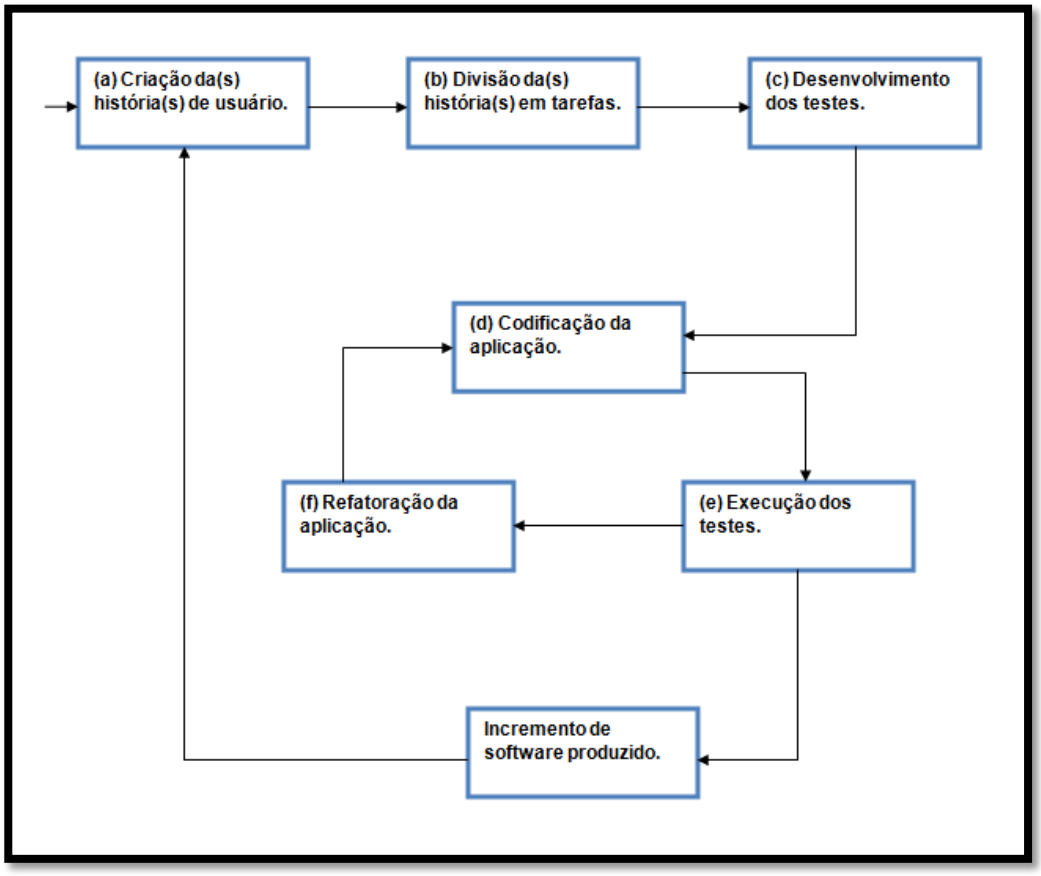

Figura 7: Fluxograma de desenvolvimento do sistema de publicação de conteúdo.

Fonte: Autoria própria.

O primeiro passo a ser dado para o desenvolvimento da aplicação foi a criação das histórias de usuário, pois foi através delas que os requisitos e funcionalidades do sistema foram extraídos e especificados para o desenvolvimento da aplicação. Após as histórias serem elaboradas, elas foram divididas em tarefas, as quais correspondem aos requisitos e funcionalidades que foram impostas a cada *release* do sistema.

As tarefas um e dois são referentes ao primeiro *release* do sistema, e estão representadas nos quadros 4 e 5 respectivamente.

Tarefa 1: Implementar página inicial.
Ao acessar o sistema, o usuário irá visualizar a página inicial. Serão apresentadas nessa página informações referentes ao sistema como o nome, logo e a versão do sistema. A página de apresentação deverá apresentar um formulário para realizar o *login* e um link para acesso ao formulário de cadastro de usuário.

Quadro 4 - Tarefa 1: Implementação página inicial.

Fonte: Autoria própria.

Tarefa 2: Implementar o *login* de usuário.
Após acessar a página inicial, o usuário poderá inserir seu *login* e senha para logar-se à aplicação e clicar no botão *login*. Para isso, o usuário deverá inserir o *login* e senha corretos, os quais foram cadastrados por ele no formulário de cadastro de usuário. O formulário de *login* deverá validar a unicidade dos campos e verificar a existência dos dados inseridos. Se o *login* e a senha estiverem corretos o usuário conseguirá logar com sucesso e será redirecionado ao seu perfil, caso contrário, o sistema retornará uma mensagem de erro informando ao usuário que os dados inseridos estão incorretos, pedindo-lhe que sejam inseridos novamente.

Quadro 5 - Tarefa 2: Implementação do *login* de usuário.

Fonte: Autoria própria.

A tarefa três no quadro 6 corresponde ao segundo *release* do sistema, e é apresentada a seguir:

Tarefa 3: Implementar cadastro de usuários.
A página de apresentação deverá conter um link que permitirá que o usuário realize o seu cadastro. Após clicar no link o usuário será redirecionado a um formulário de cadastro, no qual deverão ser preenchidos todos os campos. Serão pedidas no formulário, informações como o seu nome completo, sexo, data de nascimento, e-mail, cidade e estado. O usuário também deverá inserir o *login* e a senha que serão utilizados para realizar o *login* na aplicação. O formulário de cadastro deverá validar os campos e verificar a existência dos dados inseridos. Após preencher o formulário, o usuário receberá uma mensagem informando-lhe que realizou o seu cadastro com sucesso e será redirecionado para a página inicial onde poderá realizar o seu *login* e acessar o seu perfil.

Quadro 6 - Tarefa 3: Implementação do cadastro de usuário.

Fonte: Autoria própria.

As tarefas quatro (Quadro 7) e cinco (Quadro 8) correspondem ao terceiro e último *release* do sistema, e são apresentadas a seguir:

(continua) **Tarefa 4:** Implementar perfil pessoal.
Após o usuário ter realizado o seu *login*, ele terá acesso ao seu perfil pessoal. A página deverá apresentar algumas informações do usuário como o nome completo, sexo, cidade e estado, as quais deverão corresponder com as que foram inseridas no cadastro. Será possível visualizar todos os textos publicados pelo usuário, os quais aparecerão em uma janela com uma barra de rolagem que permitirá que as postagens mais antigas sejam visualizadas.

(conclusão)
No topo da página será visualizado um link para o *logout* do usuário, o qual permitirá que ele saia de seu perfil.

Quadro 7 - Tarefa 4: Implementação do perfil pessoal do usuário.

Fonte: Autoria própria.

Tarefa 5: Implementar publicação de conteúdo.
Após o usuário acessar o seu perfil pessoal, ele poderá publicar textos, os quais serão visualizados por ele e pelos demais usuários cadastrados no sistema. Para isso ele deverá escrever seu texto na área de texto que é apresentada logo abaixo do quadro que mostra suas publicações e apertar o botão enviar e em seguida verá seu conteúdo publicado na página.

Quadro 8 - Tarefa 5: Implementação da publicação de conteúdo.

Fonte: Autoria própria.

Neste trabalho os testes foram desenvolvidos utilizando o conceito de Desenvolvimento Orientado a Testes de Aceitação (ATDD), que assim como o Desenvolvimento Orientado a Testes (TDD) baseia-se na criação de testes antes do código, sendo que no ATDD os testes de aceitação são criados com o objetivo de testar o comportamento desejado para a aplicação, em comparação ao uso de testes de unidade no TDD. São criados um ou mais testes de aceitação para cada funcionalidade do sistema, os quais são discutidos e criados com o apoio do cliente, especificando e definindo os requisitos e funcionalidades da aplicação (Ferreira e Silva, 2012).

A aplicação foi desenvolvida seguindo o padrão de codificação MVC (*Model-View-Controller*) o qual utiliza uma abordagem orientada a objetos e separa o código em camadas.

A visão (*view)* é a camada responsável pela apresentação do sistema, a qual proporciona a iteração entre o usuário e a aplicação. O controle (*controller*) é a

camada intermediária que controla o fluxo do programa, é nela que se encontra toda a lógica de uso do protocolo HTTP (*Hyper Text Transport Protocol*), e a *model* é a camada responsável por armazenar e mostrar as informações contidas no banco de dados do sistema e a realização de operações CRUD (*Create, Read, Update e Delete*).

O fluxograma a seguir (Figura 12) mostra como é realizada a comunicação entre as camadas no padrão MVC:

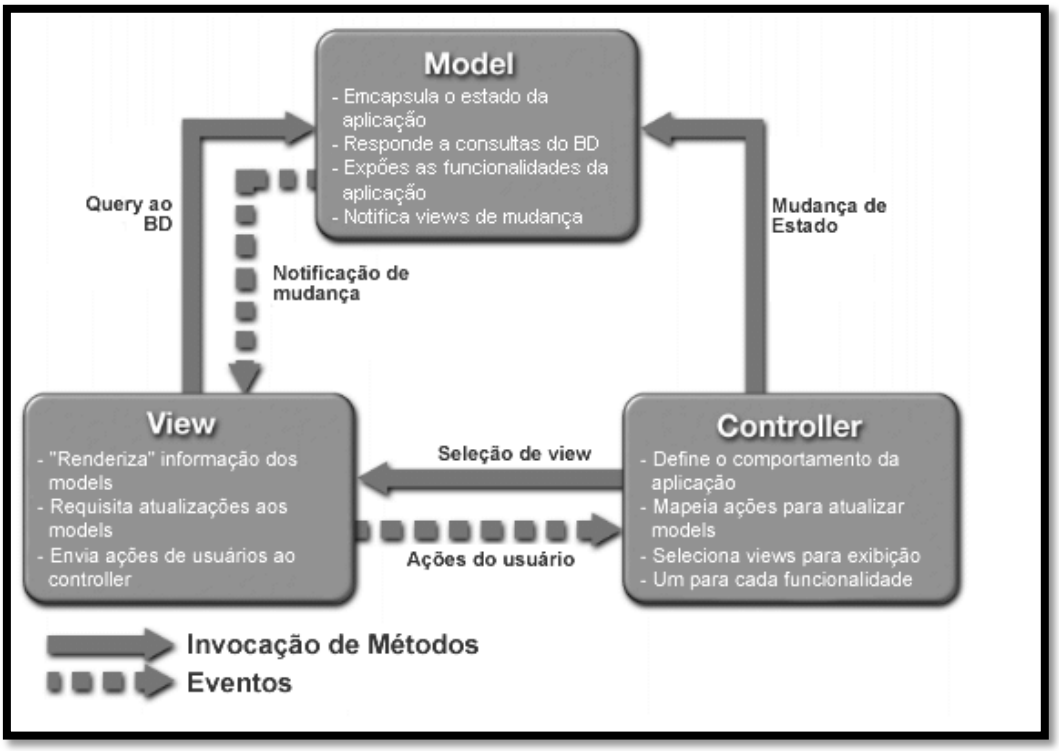

Figura 8: Fluxograma MVC.

Fonte: Zemel (2009).

Todo o ciclo é definido pela *controller*, ela é a camada responsável pela seleção de *views* a serem exibidas, de acordo com determinada requisição HTTP, também é responsável pela chamada de consultas ou atualizações junto à camada *model*, denominado mudança de estados, a qual executa as suas determinadas ações ao banco.

3.1 FERRAMENTAS UTILIZADAS

Após a divisão e descrição das tarefas, foram desenvolvidos os testes de aceitação para cada um dos *releases* mostrados anteriormente. Para o desenvolvimento e execução dos testes foram utilizadas duas ferramentas:

Selenium: é uma ferramenta desenvolvida para testar aplicações web pelo navegador de forma automatizada. *Selenium* se refere ao *Acceptance Testing* (ou *functional testing*) que envolve executar testes em um sistema finalizado. Os testes são executados diretamente em um navegador, exatamente como o usuário faria.

O *selenium-server* deve ser executado antes dos testes de aceitação serem executados. Para executarmos o servidor utilizamos o comando mostrado na figura (Figura 9) a seguir:

```
root@vader-Dell-System-XPS-L502X:~# java -jar selenium-server-standalone-2.38.0.jar
```

Figura 9: Comando utilizado para a execução do *selenium-server*.

Fonte: Autoria própria.

Após o comando ser executado a figura 10 mostra o *selenium-server* sendo executado no sistema operacional Ubuntu 12.04:

```
root@vader-Dell-System-XPS-L502X:~# java -jar selenium-server-standalone-2.38.0.jar
Abr 24, 2014 12:56:50 PM org.openqa.grid.selenium.GridLauncher main
Informações: Launching a standalone server
12:56:51.348 INFO - Java: Oracle Corporation 24.45-b08
12:56:51.358 INFO - OS: Linux 3.8.0-38-generic amd64
12:56:51.388 INFO - v2.38.0, with Core v2.38.0. Built from revision bd32d4e
12:56:51.533 INFO - Default driver org.openqa.selenium.ie.InternetExplorerDriver registration is skipped:
ernet explorer, version=}] does not match with current platform: LINUX
12:56:51.680 INFO - Default driver org.openqa.selenium.iphone.IPhoneDriver registration is skipped: regist
urrent platform: LINUX
12:56:51.688 INFO - Default driver org.openqa.selenium.iphone.IPhoneDriver registration is skipped: regist
 current platform: LINUX
12:56:51.785 INFO - RemoteWebDriver instances should connect to: http://127.0.0.1:4444/wd/hub
12:56:51.786 INFO - Version Jetty/5.1.x
12:56:51.787 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
12:56:51.787 INFO - Started HttpContext[/selenium-server,/selenium-server]
12:56:51.787 INFO - Started HttpContext[/,/]
12:56:51.891 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@fa09c82
12:56:51.891 INFO - Started HttpContext[/wd,/wd]
12:56:51.915 INFO - Started SocketListener on 0.0.0.0:4444
12:56:51.916 INFO - Started org.openqa.jetty.jetty.Server@44ee06be
```

Figura 10: S*elenium-server em execução*.

Fonte: Autoria própria.

Pyccuracy: é uma ferramenta escrita em *Python* que tem como objetivo tornar mais fácil o desenvolvimento de testes de aceitação automatizados. Além disso, melhora a legibilidade dos testes usando uma linguagem natural estruturada e um mecanismo simples para compreender essa linguagem, para que desenvolvedores e clientes possam colaborar e entender o que fazem os testes.

A Figura 11 apresenta um teste de aceitação elaborado durante o desenvolvimento deste trabalho, com o intuito de testar a funcionalidade de cadastro de usuário.

```
Como um usuário sem cadastro
Eu quero me cadastrar na aplicação
Para que eu possa postar conteúdos

Cenário 1 - Cadastro na aplicação

Dado que
        Eu navego para "http://localhost/TCC/principal/novo_usuario"
        E eu espero por 3 segundos
Quando
        #Eu clico no link "Cadastre-se"
        Eu vejo o título "Cadastro de usuario"
        Eu preencho a caixa de texto "nome" com "Joao"
        E eu espero por 3 segundos
        E eu marco a radio "rb_masculino"
        E eu espero por 3 segundos
        E eu preencho a caixa de texto "login" com "joao"
        E eu espero por 3 segundos
        E eu preencho a caixa de texto "email" com "joao@gmail.com"
        E eu espero por 3 segundos
        E eu preencho a caixa de texto "senha" com "1234"
        E eu espero por 3 segundos
        E eu preencho a caixa de texto "cidade" com "Ponta Grossa"
        E eu espero por 3 segundos
        E eu seleciono o item com texto "PR" na select "uf"
        E eu espero por 3 segundos
        E eu seleciono o item com texto "1" na select "dia"
        E eu espero por 3 segundos
        E eu seleciono o item com texto "Janeiro" na select "mes"
        E eu espero por 3 segundos
        E eu seleciono o item com texto "1988" na select "ano"
        E eu espero por 3 segundos
Então
        Eu clico no botão "btn_cadastro"
        E eu espero por 5 segundos
        E eu vejo o título "btn_login"
```

Figura 11: Teste de aceitação escrito utilizando linguagem natural.

Fonte: Autoria própria.

Após o desenvolvimento dos testes, utilizaram-se as seguintes ferramentas para a codificação de cada um dos *releases* da aplicação:

XAMPP: é um servidor que consiste principalmente na base de dados MySQL, o servidor web Apache e os interpretadores para linguagens de script: PHP e Perl.

NetBeans: é um ambiente de desenvolvimento integrado (IDE) gratuito para desenvolvedores de software nas linguagens Java, C, C++, PHP, Groovy, Ruby, entre outras. Ele pode ser utilizado em muitas plataformas como Windows, Linux, Solaris e MacOS, oferecendo diversas ferramentas necessárias para a criação de aplicativos.

GitHub: é um serviço de *Web Hosting* compartilhado para projetos que usam o controle de versão Git. Este site possui funcionalidades de uma rede social como *feeds*, *followers*, *wiki* e um gráfico que mostra como os desenvolvedores trabalham as versões de seus repositórios.

A Figura 12 mostra o repositório criado para o compartilhamento da aplicação desenvolvida:

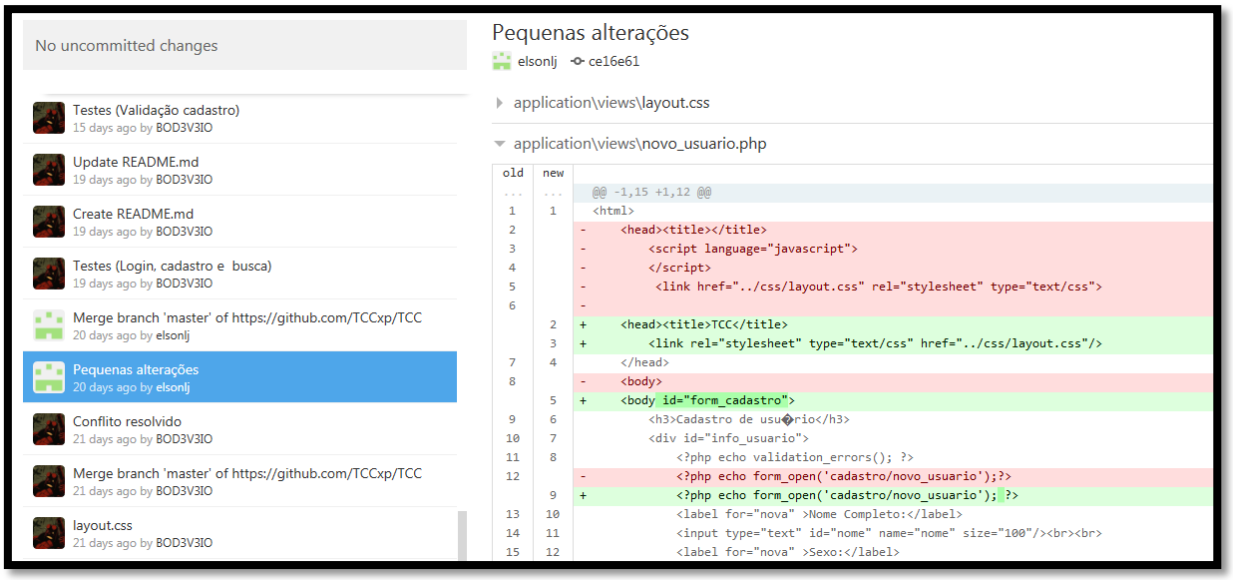

Figura 12: Repositório criado no *GitHub*.

Fonte: Autoria própria.

CodeIgniter: é um *framework* para desenvolvimento de aplicações PHP, que possui ferramentas que permitem desenvolver aplicações muito mais rápido do que poderíamos fazer sem a utilização de um *framework*. Ele contém um excelente

conjunto de bibliotecas para tarefas comuns necessárias, interfaces simples e uma estrutura lógica para acesso a estas bibliotecas.

O *CodeIgniter* utiliza a abordagem MVC (*Model-View-Controller*), permitindo a separação entre a lógica e a apresentação da aplicação.

O trecho de código apresentado na figura 13 mostra a *view* **novo_usuario.php**, onde foi criado o formulário de cadastro desenvolvido para este trabalho:

```
<div id="templatemo_middle">

    <h3>Cadastro de usuário</h3>
    <div id="form_cadastro">
        <?php echo validation_errors(); ?>
        <?php echo form_open('cadastro/novo_usuario'); ?>
        <label for="nova" >Nome Completo:</label>
        <input type="text" id="nome" name="nome" size="70"/><br><br>
        <label for="nova" >Sexo:</label>
        <input type="radio" id="rb_masc" name="sexo" value="masculino"/>Masculino
        <input type="radio" id="rb_fem" name="sexo" value="feminino"/>Feminino<br><br>
        <label for="nova" >Login:</label>
        <input type="text" id="login" name="login" size="70"/><br><br>
        <label for="nova" >Senha:</label>
        <input type="password" id="senha" name="senha" size="30"/><br><br>
        <label for="nova" >Email:</label>
        <input type="text" id="email" name="email" size="70"/><br><br>
        <label for="nova" >Cidade:</label>
        <input type="text" id="cidade" name="cidade" size="50"/><br><br>
        <label for="nova" >Estado:</label>
        <span class="select_data" data-type="selectors" data-name ="uf">
            <select name="uf" id="uf" class="">
                <option value="0" selected="1">--</option>
                <option value="PR">PR</option>
                <option value="SP">SP</option>
                <option value="RJ">RJ</option>
            </select>
        </span><br><br>
        <label for="nova">Data de Nascimento:</label>
        <...67 linhas /><br />
        <br />
```

Figura 13: *View* novo_usuario.php.

Fonte: Autoria própria.

O trecho de código apresentado na figura 14 mostra a *controller* **cadastro.php** que faz a intermediação entre a *view* **novo_usuario.php** e a *model* **inserir.php**:

```php
class Cadastro extends CI_Controller {

    function __construct() {
        parent::__construct();
        $this->load->model('inserir', '', TRUE);
    }

    function novo_usuario() {
        $this->load->library('form_validation');
        $this->form_validation->set_rules('nome', 'Nome', 'required');
        $this->form_validation->set_rules('sexo', 'Sexo', 'required');
        $this->form_validation->set_rules('login', 'Login', 'required');
        $this->form_validation->set_rules('senha', 'Senha', 'required');
        $this->form_validation->set_rules('email', 'Email', 'required');
        $this->form_validation->set_rules('cidade', 'Cidade', 'required');
        $this->form_validation->set_rules('uf', 'Estado', 'required');
        $this->form_validation->set_rules('dia_nasc', 'Data de Nascimento', 'required');
        $this->form_validation->set_rules('mes_nasc', 'Data de Nascimento', 'required');
        $this->form_validation->set_rules('ano_nasc', 'Data de Nascimento', 'required');
        if ($this->form_validation->run() != FALSE) {
            $query = array(
                'nome' => $this->input->post('nome'),
                'sexo' => $this->input->post('sexo'),
                'login' => $this->input->post('login'),
                'senha' => $this->input->post('senha'),
                'email' => $this->input->post('email'),
                'cidade' => $this->input->post('cidade'),
                'uf' => $this->input->post('uf'),
                'data' => $this->input->post('ano_nasc,mes_nasc,dia_nasc'));
            $this->inserir->valida_novo_usuario($query);
            echo ("<script language='JavaScript'>
                    window.alert('Usuario cadastrado')
                    window.location.href='http://localhost:8888/TCC/';
                </script>");
            redirect('principal');
        }
```

Figura 14: *Controller* cadastro.php.

Fonte: Autoria própria.

Neste trecho foi utilizada a biblioteca "form_validation", como pode ser observada na Figura 14, a qual é uma biblioteca de configurações utilizadas para validação dos valores inseridos na *view* de cadastro de usuário (Figura 13).

Após as configurações serem determinadas, ela fará a verificação das informações inseridas e retornará um aviso caso elas não estejam de acordo com as regras definidas na aplicação. Exemplos de configurações dessa biblioteca são campos obrigatórios, tamanho máximo e mínimo, valores numéricos, endereço de e-mail, entre outros.

O trecho de código apresentado na figura 15 mostra a *model* **inserir.php**, que é responsável por validar o *login* e a senha do usuário cadastrado e pela inserção no banco de dados:

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
    class Inserir extends CI_Model {

        function __construct() {
            parent::__construct();
        }

        function valida_novo_usuario($dados){

            $this -> db -> select('login, email');
            $this -> db -> from('usuario');
            $this -> db -> where('login', $dados['login']);
            $this -> db -> where('email', $dados['email']);
            $query = $this -> db -> get();
            if($query -> num_rows() == 0)
            {
                return $this->novo_usuario($dados);
            }
            else
            {
                echo
                ("<script language='JavaScript'>
                    window.alert('Sucesso!')
                </script>");
                redirect('principal');
            }
        }

        function novo_usuario($dados){

            return $this->db->insert('usuario', $dados);
        }
    }
?>
```

Figura 15: *Model* inserir.php.

Fonte: Autoria própria.

4 RESULTADOS

A seguir são apresentadas as interfaces e os resultados obtidos após a execução dos testes de aceitação para cada um dos *releases* da aplicação desenvolvida para este trabalho.

4.1 TESTES PRIMEIRO RELEASE

A Figura 16 apresenta a interface desenvolvida para o primeiro *release* da aplicação, referente a pagina inicial e ao *login* de usuário:

Figura 16: Interface - Página Principal / *Login*.

Fonte: Autoria própria.

O objetivo da página principal é mostrar informações referentes à aplicação, apresentando ao usuário uma breve descrição do que se trata o sistema,

possibilitando que o mesmo realize o seu *login* na aplicação. Ela possui um *link* que direciona o usuário para a tela de cadastro que será mostrada mais adiante.

Durante a codificação do primeiro *release* da aplicação, o teste "TCC_testeLogin.acc" (Figura 17) foi executado:

Figura 17: Falha no teste de *Login*.

Fonte: Autoria própria.

Como pode ser visto figura 17, o teste "TCC_testeLogin.acc" apresentou os status "*FAILED*", indicando que houve uma falha no cenário desenvolvido para o teste. A falha se encontra na linha onde está escrito "Eu vejo o título "Perfil"", na seção "Então" do teste, a qual ocorreu devido à incompatibilidade do título da página descrito no teste com o título da página que estava escrito na aplicação (Perfil Usuário), a qual é apresentada na última linha marcada em vermelho no teste.

Após uma falha ser encontrada em um teste, este deve ser analisado e o trecho de código em que houve a falha indicada deve ser refatorado, para que na próxima execução (Figura 18) o teste possa ser aprovado.

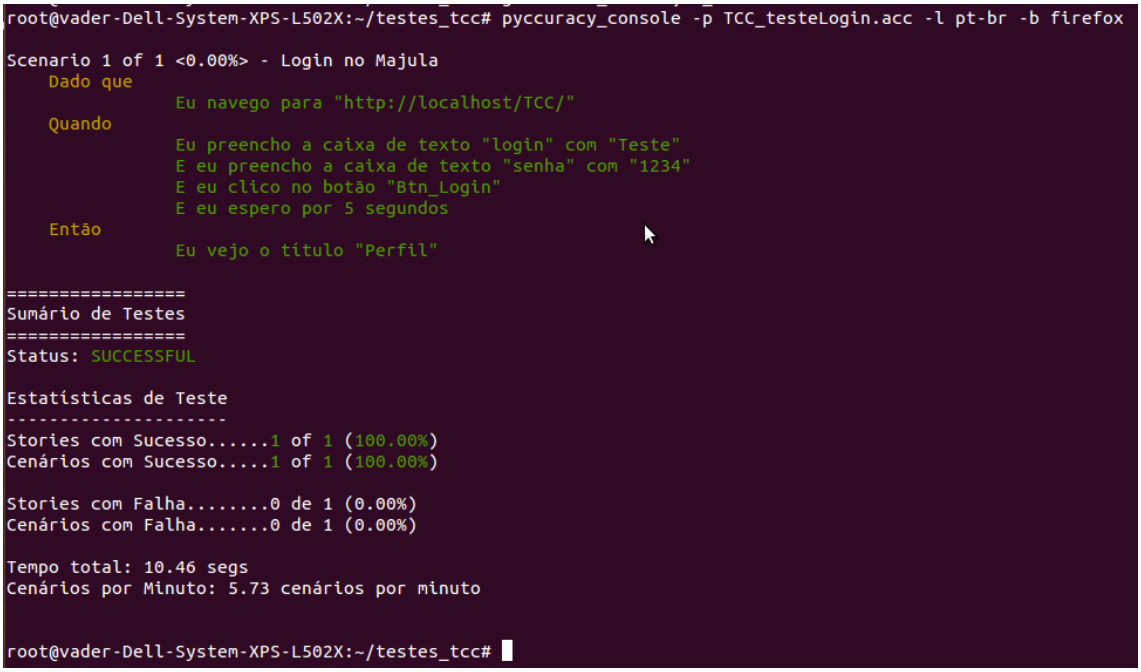

Figura 18: Teste de *Login* aprovado com sucesso.

Fonte: Autoria própria.

Como pode ser visto na figura 18 o código foi refatorado e o teste executado novamente, obtendo o status "*SUCCESSFUL*", o qual indica que o teste foi aprovado e a funcionalidade do sistema está de acordo com a que foi imposta pelo cliente na história de usuário elaborada no início da construção do primeiro *release* do sistema.

O caso apresentado no teste de aceitação do primeiro *release* do sistema aponta uma das principais vantagens do Desenvolvimento Orientado a Testes de Aceitação (ATDD), que é a identificação dos erros na aplicação e a correção dos mesmos, o que torna muito mais fácil a manutenção da aplicação, reduzindo a probabilidade de introdução de erros nos incrementos de software que serão desenvolvidos futuramente no decorrer do projeto.

Os testes de aceitação também possuem um papel importante no controle do código que foi desenvolvido, pois se um teste aprovado anteriormente sofrer uma falha em uma execução posterior, é sinal que o código da aplicação foi alterado por alguém da equipe de desenvolvimento e esta alteração não foi repassada aos demais membros.

4.2 TESTES SEGUNDO RELEASE

A Figura 19 apresenta a interface desenvolvida para o segundo *release* da aplicação, referente ao cadastro de usuários:

Figura 19: Interface - Cadastro de usuário.

Fonte: Autoria própria.

Através desta página o usuário poderá realizar o seu cadastro na aplicação, preenchendo o formulário com os seus dados (nome, sexo, *login*, senha, e-mail, cidade, estado e data de nascimento). Após a realização do cadastro ele terá uma conta válida no sistema e poderá realizar o *login* na aplicação inserindo o *login* e a senha cadastrados.

Durante a codificação do segundo *release* da aplicação, o teste "TCC_testeCadastro.acc" (Figura 20) foi executado:

```
root@vader-Dell-System-XPS-L502X:~/testes_tcc# pyccuracy_console -p TCC_testeCadastro.acc -l pt-br -b firefox

Scenario 1 of 1 <0.00%> - Cadastro no Majula
    Dado que
                #Eu clico no link "Cadastre-se"
                Eu navego para "http://localhost/TCC/principal/novo_usuario"
                E eu espero por 3 segundos
    Quando
                Eu vejo o titulo "Cadastro"
                E eu preencho a caixa de texto "nome" com "Joao"
                E eu espero por 3 segundos
                E eu marco a radio "rb_masc"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "login" com "joao"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "email" com "joao@gmail.com"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "senha" com "1234"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "cidade" com "Ponta Grossa"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "PR" na select "uf"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "1" na select "dia"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "Janeiro" na select "mes"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "1988" na select "ano"
                E eu espero por 3 segundos
    Então
                Eu clico no botão "cadastrar"
                E eu espero por 5 segundos
                E eu vejo o título "Login"

=================
Sumário de Testes
=================
Status: SUCCESSFUL

Estatísticas de Teste
---------------------
Stories com Sucesso......1 of 1 (100.00%)
Cenários com Sucesso.....1 of 1 (100.00%)

Stories com Falha........0 de 1 (0.00%)
Cenários com Falha.......0 de 1 (0.00%)

Tempo total: 44.12 segs
Cenários por Minuto: 1.36 cenários por minuto
```

Figura 20: Teste de cadastro aprovado com sucesso.

Fonte: Autoria própria.

Como pode ser visto na Figura 20 o teste de aceitação "TCC_testeCadastro.acc" recebeu o status "SUCCESSFULL", e como mostra a estatística do teste, o cenário obteve 100% de acerto. Este teste foi desenvolvido com o intuito de testar a funcionalidade do cadastro através dos componentes inseridos no formulário de cadastro (*textfields, radio buttons, buttons e selects*).

4.3 TESTES TERCEIRO RELEASE

A Figura 21 apresenta a interface desenvolvida para o terceiro e último *release* da aplicação, referente ao perfil de usuário e publicação de conteúdo:

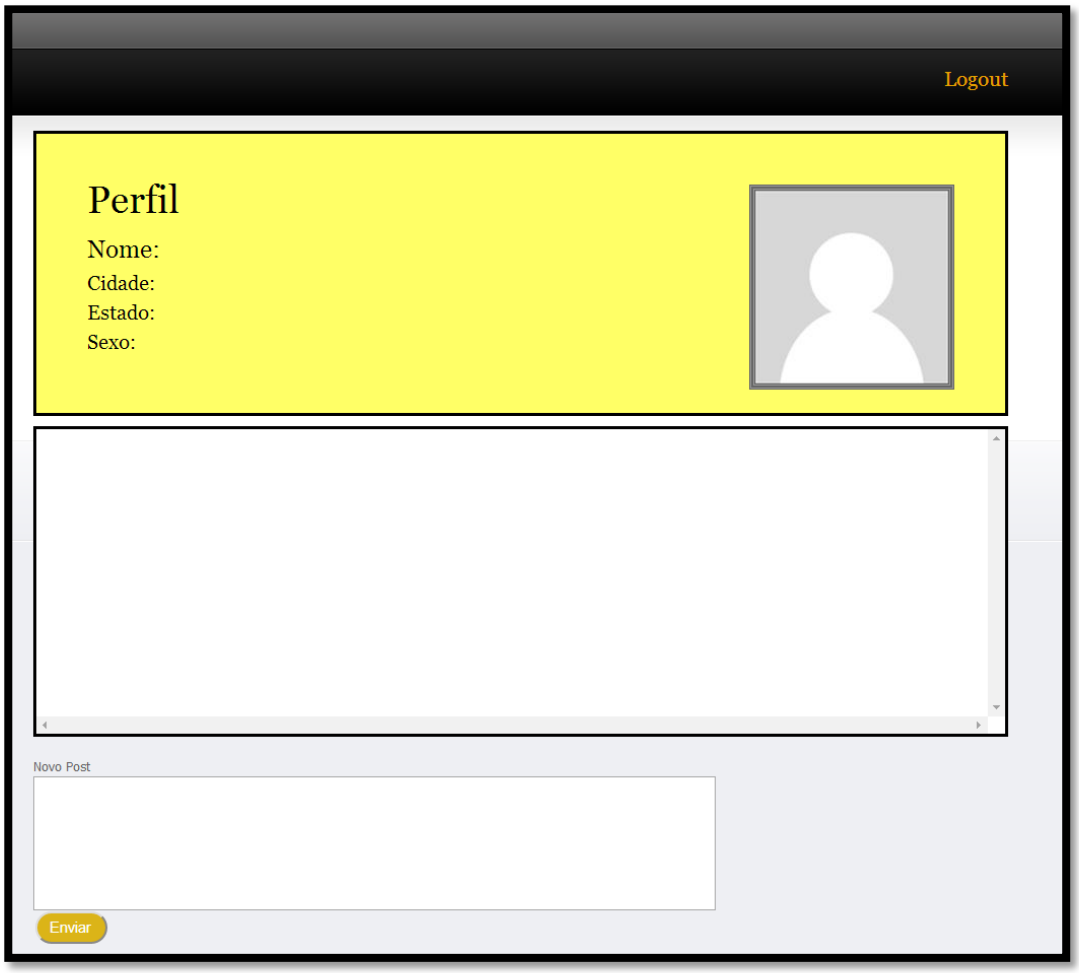

Figura 21: Interface - Perfil de usuário.

Fonte: Autoria própria.

Esta página apresentará as informações do usuário e os textos publicados por ele. O usuário poderá inserir os textos que desejar escrevendo-os na área de texto "Novo Post", e publicá-los clicando no botão "Enviar" que se encontra logo abaixo. Após publicar o seu texto, ele poderá ser visualizado na área acima da área de texto, em seu perfil, a qual possui duas barras de rolagem que auxiliam na visualização dos textos publicados.

Durante a codificação do terceiro *release* da aplicação, o teste "TCC_testePerfil.acc" (Figura 22) foi executado:

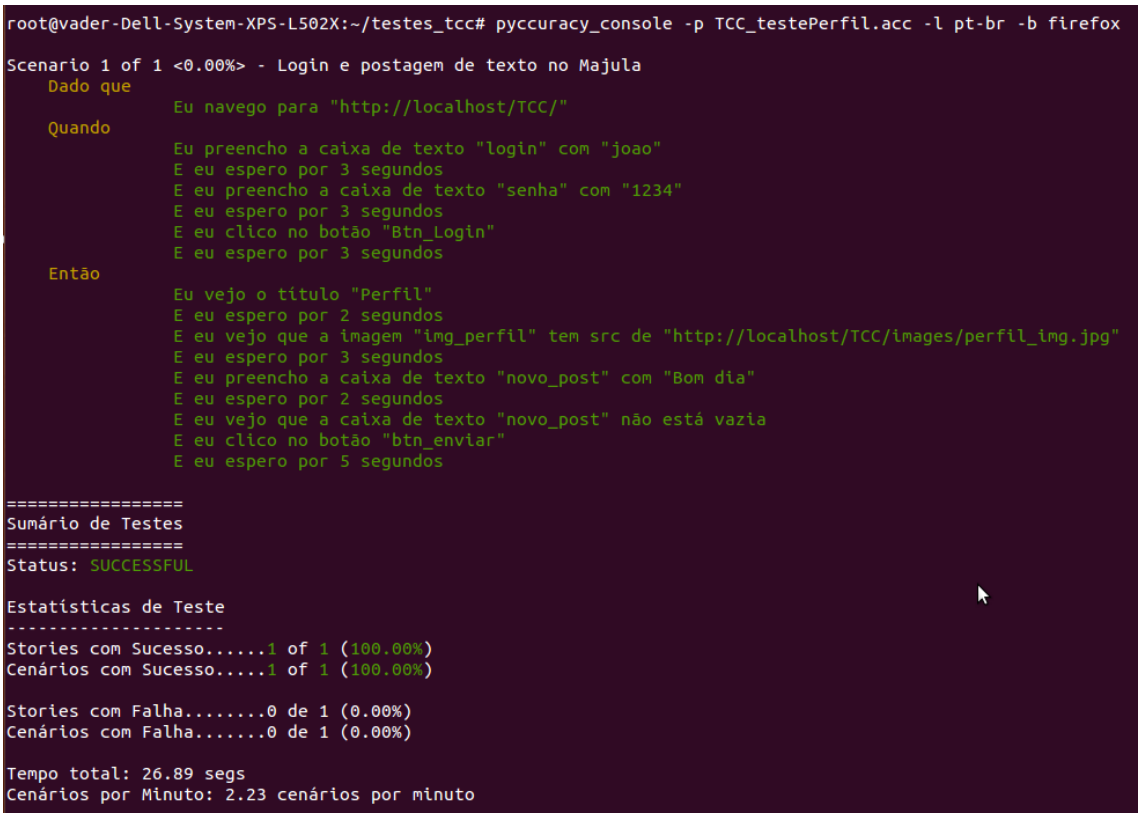

Figura 22: Teste de perfil de usuário aprovado com sucesso.

Fonte: Autoria própria.

Na figura 22 o teste de aceitação "TCC_testePerfil.acc" recebeu o status "SUCCESSFULL", e como mostra a estatística do teste, o cenário obteve 100% de acerto. Neste teste foram testadas as funcionalidades do *login* e da publicação de conteúdo em conjunto, pois para que um usuário possa realizar uma publicação é obrigatório que o mesmo esteja logado na aplicação.

Após o desenvolvimento dos testes para cada um dos três *releases* do sistema, foram elaborados testes de aceitação que testam as funcionalidades da aplicação em conjunto, para verificar a integração dos incrementos de software produzidos.

O teste TCC_testePublicacao.acc (Figura 23 e Figura 24) executa um teste utilizando cada uma das funcionalidades desenvolvidas para a aplicação:

```
root@vader-Dell-System-XPS-L502X:~/testes_tcc# pyccuracy_console -p TCC_testePub
licacao.acc -l pt-br -b firefox

Scenario 1 of 3 <0.00%> - Login sem cadastro no Majula
    Dado que
                Eu navego para "http://localhost/TCC/"
    Quando
                Eu preencho a caixa de texto "login" com "marge_sp"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "senha" com "duff1234"
                E eu espero por 3 segundos
                E eu clico no botão "Btn_Login"
                E eu espero por 3 segundos
    Então
                Eu vejo o título "Login"

Scenario 2 of 3 <33.33%> - Cadastro no Majula
    Dado que
                Eu navego para "http://localhost/TCC/principal/novo_usuario"
                E eu espero por 3 segundos
    Quando
                Eu vejo o título "Cadastro"
                Eu preencho a caixa de texto "nome" com "Marge Simpson"
                E eu espero por 3 segundos
                E eu marco a radio "rb_fem"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "login" com "marge_sp"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "email" com "m_simpson@hotmail.co
                E eu preencho a caixa de texto "email" com "m_simpson@hotmail.co
m"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "senha" com "duff1234"
                E eu espero por 3 segundos
                E eu preencho a caixa de texto "cidade" com "Springfield"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "PR" na select "uf"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "17" na select "dia"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "Dezembro" na select "mes"
                E eu espero por 3 segundos
                E eu seleciono o item com texto "1989" na select "ano"
                E eu espero por 3 segundos
    Então
                Eu clico no botão "cadastrar"
                E eu espero por 5 segundos
                E eu vejo o título "Login"
```

Figura 23: Teste de publicação - Cenários 1 e 2.

Fonte: Autoria própria.

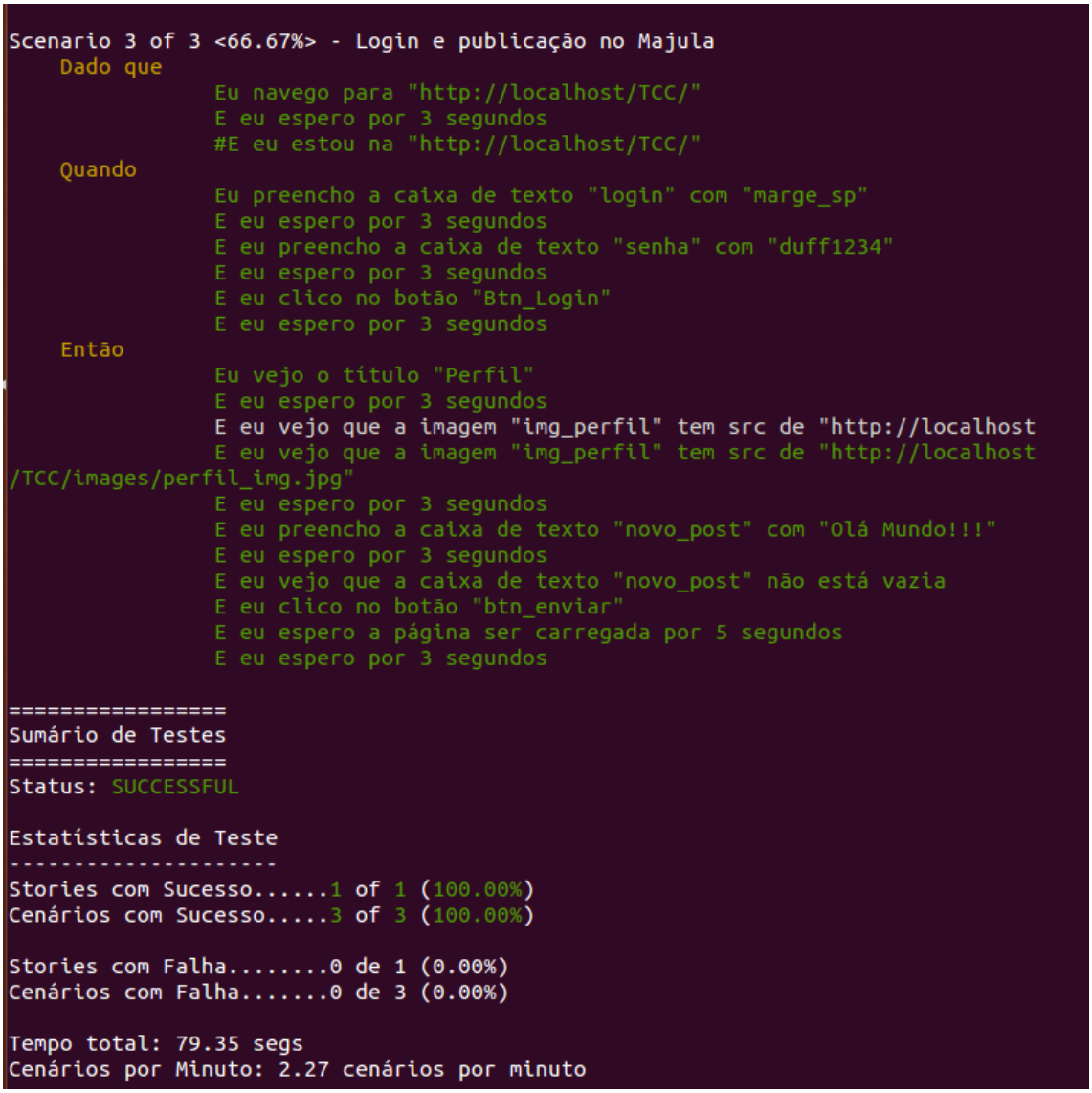

Figura 24: Teste de publicação - Cenário 3.

Fonte: Autoria própria.

O teste "TCC_testePublicacao.acc" é composto por três cenários, os quais são executados em sequência. O primeiro cenário tenta realizar o *login* de um usuário que não possui seu cadastro no sistema, o qual receberá uma mensagem informando que o seu *login* e sua senha são inválidos, não sendo redirecionado à página de perfil. Após a tentativa de *login* sem sucesso, a ação do segundo cenário é realizar o cadastro do usuário, validando o *login* e a senha que foram utilizados na tentativa de *login* apresentada anteriormente. Após a execução do segundo cenário, o usuário terá seu cadastro efetuado com sucesso, podendo assim realizar o *login* na aplicação, ação que é realizada na seção "Quando" do terceiro cenário (Figura

23), e será redirecionado para a página de seu perfil, e no mesmo cenário a próxima ação a ser executada pelo teste será a publicação do texto "Olá Mundo!!!" em seu perfil pessoal.

A Figura 24 ilustra um cenário de teste aprovado e a somatória dos cenários resulta em 100% de acerto (33.33% de acerto para cada cenário).

Com a execução do teste "TCC_testePublicacao.acc" foi possível verificar que a integração dos incrementos de software de cada um dos *releases* da aplicação foi realizada com sucesso, pois os testes apresentaram o comportamento esperado, correspondendo com as funcionalidades e requisitos especificados nas histórias de usuário elaboradas para o desenvolvimento deste trabalho.

A aplicação e todos os testes desenvolvidos para este trabalho se encontram no repositório compartilhado no *GitHub*, o qual pode ser acessado através do link:<https://github.com/TCCxp/TCC>.

5 CONCLUSÃO

Através das práticas e métodos adotados pelo uso da metodologia XP, foi possível desenvolver a aplicação, seguindo as exigências impostas para o seu desenvolvimento. Os valores transmitidos pela XP diferem bastante das metodologias convencionais, o que leva alguns autores a considerarem uma metodologia mais "humana" em relação às metodologias tradicionais. Cockburn (2002) argumenta que as metodologias convencionais muitas vezes deixam de lado a fragilidade das pessoas que constroem o software, pois os engenheiros de software não são máquinas e possuem diferenças de habilidade, criatividade e outros fatores que influenciam diretamente um projeto. Com a elaboração das histórias de usuário e com o desenvolvimento orientado a testes de aceitação (ATDD) foi possível manter o escopo do projeto, pois os testes, além de testarem as funcionalidades da aplicação, proporcionaram o controle das mudanças realizadas no código. A programação em pares possibilitou a disseminação de conhecimento e exercício da comunicação entre as pessoas envolvidas no projeto. A execução dos testes de aceitação durante o desenvolvimento deste trabalho possibilitou verificar a integração de cada incremento do software produzido, proporcionando a sincronia de cada incremento de software elaborado separadamente.

Portanto, pode-se concluir que os métodos ágeis de desenvolvimento, e especificamente a metodologia XP, são alternativas de desenvolvimento de software que podem ser utilizadas em projetos de pequeno e médio porte, onde a equipe de desenvolvimento possa ser encorajada a adquirir a disciplina necessária para aplicar as técnicas e valores propostos pela XP, podendo assim elaborar uma aplicação em um tempo menor, a qual será flexível às mudanças impostas pelo cliente.

5.1 TRABALHOS FUTUROS

Como indicação de trabalho futuro, recomenda-se o uso de outros frameworks de desenvolvimento para testes. Além disso, cita-se o desenvolvimento de testes de unidade, os quais podem ser utilizados para testar as entradas e saídas do sistema, em um nível mais baixo que os testes de aceitação elaborados neste

trabalho. Outra opção é utilizar os artefatos produzidos durante o desenvolvimento e elaborar uma documentação, incluindo os cartões CRC que foram citados anteriormente neste trabalho.

6 REFERENCIAS

ALVES, José P. **Agilidade.** Lisboa: PwC, Revista CEO, 2012. 52 p.

ASTELS, David; MILLER, Granville; NOVÁK, Miroslav.***Extreme programming:*** guia prático. 1ª Edição. Rio de Janeiro: Elsevier Brasil, 2002. 376p.

BECK, Kent; CUNNINGHAM, Ward 1989.**A Laboratory For Teaching Object-Oriented Thinking**. New Orleans: OOPSLA'89 Conference Proceedings, 1989. Disponível em:< http://c2.com/doc/oopsla89/paper.html> Acessado em 12 de maio de 2014.

BECK, Kent. ***Extreme Programming Explained:*** *Embrace Change*.1ª Edição**.**Addison-Wesley.1999. 224 p.

BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. **Manifesto para Desenvolvimento Ágil de Software.**2001. Disponível em:
< http://www.agilemanifesto.org/iso/ptbr/>. Acessado em 14 de fevereiro de 2014.

BECK, Kent. ***Extreme Programming Explained:*** *Embrace Change*.2ª Edição. Addison-Wesley, 2004. 256 p.

BECK, Kent. ***Test-Driven Development:*** *By Example*.1ª Edição. Addison-Wesley, 2002.220 p.

BECK, Kent. **Programação Extrema (XP) explicada:** acolha as mudanças. Porto Alegre: Bookman, 2004. 182 p.

BECK, Kent. **TDD**: Desenvolvimento Guiado por Testes. Porto Alegre: Bookman, 2010. 241 p.

BEZERRA, Wendy S.; CONCEIÇÂO, Carla A.. **Utilização da metodologia ágil *eXtreme Programming* (XP) como ferramenta de gestão:** um estudo de caso numa empresa do ramo de tecnologia e serviços. 2012. 16 p. Universidade Potiguar: Revista científica da escola e gestão de negócios.

CASTRO, Vinicius A. **Desenvolvimento Ágil com Programação Extrema:** Eficácia e disciplina extrema no desenvolvimento orientado a objetos de software**.** 2006. 122 p.

COCKBURN, A. ***Agile Software Development*.** Addison-Wesley, 2002.

FERREIRA FILHO, José I.; SILVA, Olissea A.. **Desenvolvimento Orientado a Testes de Aceitação.** Goiania: Pontifícia Universidade Católica (PUC-GO), 2012. 11 p.

FOWLER, Martin. ***Refactoring:*** *Improving the Design of Existing Code*.Addison-Wesley, 2000. 431 p.

GOLDMAN, Alfred et al. ***Being Extreme in the Classroom:*** *experiences Teaching XP*. J. Braz. Comp. Soc., Campinas, v. 10, n. 2, Nov. 2004. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-65002004000300002&lng=en&nrm=iso>. Acessado em20 de Outubro de 2013.

JACOBSON, Ivar. ***A Resouding 'Yes' to Agile Processes-But Also More.*** Volume 15, nº 1.2002. p. 18-24.

JOAQUIM. G.M.C - **Estudo sobre a aplicação de métodos de gerenciamento ágil de projetos para o desenvolvimento de painéis de media tensão,** São Carlos, 2011.

LARMAN, Craig; VODDE, Bas. **Acceptance Test-Driven Development with Robot Framework.**2010.Disponível em :<http://wiki.robotframework.googlecode.com/hg/publications/ATDD_with_RobotFramework.pdf>. Acessado em 28 de abril de 2014.

PRESSMAN, Roger S.. **Engenharia de Software.** 6ª Edição**.** Porto Alegre: Mcgraw-Hill, 2006. 752 p.

PRESSMAN, Roger S.. **Engenharia de Software:** Uma abordagem Profissional. 7ª Edição**.** Porto Alegre: Mcgraw-Hill, 2011. 780 p.

RAMOS, Everton A.. **Metodologias Ágeis:** *Extreme Programming*. Maringá: Universidade Estadual de Maringá, 2013. 42 p.

SANTOS,Rodrigo. **Estudo de caso sobre a utilização do e*Xtreme programming* em uma pequena empresa de desenvolvimento para web.** Santa Catarina: Universidade do Sul de Santa Catarina, 2013. 53 p.

SOMMERVILLE, Ian. **Engenharia de Software. 8ª** Edição. São Paulo: Pearson, 2007. 568 p.

TELES, Vinícius M.. ***Extreme Programming:*** Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. São Paulo: Novatec, 2006. 320 p.

TELES, Vinícius M.. **Um Estudo de Caso da Adoção das Práticas e Valores do *Extreme Programming*.** Rio de Janeiro: Dissertação (Mestrado em Informática) -

Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, 2005. 181 p. Disponível em: <http://www.improveit.com.br/xp/dissertacaoXP.pdf>. Acessado em 21 de Fevereiro de 2014.

SILVA, L.F; SAMPAIO, J.C; BREITMAN, K.K. **Ensino de Engenharia de Software: Relato de Experiências,** Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2004.

ZEMEL, Tárcio. **MVC (*Model-View-Controller*).** CodeIgniter Brasil, 30 Mar. 2009. Disponível em: <http://codeigniterbrasil.com/passos-iniciais/mvc-model-view-controller/>. Acessado em 12 de Maio de 2014.