UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO
ENGENHARIA DE COMPUTAÇÃO

LUCIANO DOS REIS DE SOUSA

USO DA INTEGRAÇÃO CONTÍNUA PARA MELHORIA NA PRODUTIVIDADE E QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE EM UMA EQUIPE MULTIFUNCIONAL:

Um exemplo da equipe de desenvolvimento e manutenção de um software bancário.

São Paulo - SP
2019

LUCIANO DOS REIS DE SOUSA

USO DA INTEGRAÇÃO CONTÍNUA PARA MELHORIA NA PRODUTIVIDADE E QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE EM UMA EQUIPE MULTIFUNCIONAL.

Um exemplo da equipe de desenvolvimento e manutenção de um software bancário.

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação da Universidade Virtual de São Paulo como requisito parcial para obtenção do título de Engenheiro de Computação.

São Paulo - SP
2019

LUCIANO DOS REIS DE SOUSA

USO DA INTEGRAÇÃO CONTÍNUA PARA MELHORIA NA PRODUTIVIDADE E QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE EM UMA EQUIPE MULTIFUNCIONAL:
O exemplo da Centralizadora de TI da Caixa Econômica Federal

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação da Universidade Virtual de São Paulo como requisito parcial para obtenção do título de Engenheiro de Computação.

Aprovado em ____/____/_____

Conceito: _____

BANCA EXAMINADORA

_____

PROFESSOR PRESIDENTE

_____

PROFESSOR MEMBRO

_____

PROFESSOR MEMBRO

RESUMO

Qualidade de software, além de influenciar na percepção dos usuários quanto ao produto, pode significar redução de custos em manutenção e prevenção de prejuízos financeiros. Na CEDES – Centralizadora de Desenvolvimento de TI da CEF – CAIXA Econômica Federal, onde os serviços de tecnologia da informação são realizados por funcionários públicos e terceirizados, é importante que haja domínio sobre os processos e qualidade dos seus sistemas, para auxiliar tomada de decisões e exigir devidamente o nível de qualidade esperado. Este trabalho tem por objetivo propor uma melhoria no processo de desenvolvimento de software dos sistemas de canais da empresa - MAA, por meio da implementação de protótipo de um ambiente de Integração Contínua e conscientização dos stakeholders. Esta proposta foi elaborada a partir de um estudo de caso junto à equipe de desenvolvimento da empresa, que utiliza a abordagem ágil. Este trabalho conta com uma revisão de literatura acerca de temas correlatos à Integração Contínua, Processos de Software Qualidade de Software e Testes Automatizados. É esperado, como projeto futuro a automatização de alguns processos utilizando conceitos, ferramentas de hardware e software livre e de mercado, que foram discutidos durante a graduação.

Palavras chave: Desenvolvimento de Software, Integração Contínua, Jenkins

SUMÁRIO

1. INTRODUÇÃO

Com o advento da Transformação digital as empresas as empresas de tecnologia se se viram num cenário garantir resultados melhores devido à alta competitividade. Isto significa mudança de processos, tecnologia e uma mudança estrutural nas organizações dando um papel essencial para a tecnologia e inovação.

No caso estudado, uma equipe de desenvolvedores que estão trabalhando em um projeto juntos, utilizam como servidor de versionamento do código-fonte, como o SVN – O Apache Subversion. É comum, por engano, distração, ou até mesmo falta de treinamento que o programador envie um código que não compila para o servidor de versionamento ou até mesmo que se baixe código fonte de uma pasta desatualizada do mesmo servidor. Os outros desenvolvedores, quando forem obter o código mais recente, podem acabar obtendo o código quebrado, que havia sido enviado por engano ou até mesmo a versão base errada para desenvolver uma nova funcionalidade. Isto atrapalha o fluxo de desenvolvimento da equipe, atrasando entregas em tempo de Sprints negociadas com o dono do produto e gera erros advindos da gerência de configuração que podem causar desde pequenos erros à prejuízo financeiro quando o erro não é detectado nos testes e homologação e vai à produção.

Outro exemplo bastante comum, é quando queremos pegar a última versão de projeto para construir e colocar no servidor de produção. Quando não temos um servidor que executa nossos testes, o desenvolvedor responsável por colocar o código em produção só vai descobrir se o mesmo possui um bug ou não quando executar os testes em sua própria máquina. Pode assim acontecer que o dono do produto que fica

aguardando receber um software funcional, mas na véspera do lançamento, quando deveria ser enviado para produção que os bugs são descobertos. Tanto o dono do produto quanto o desenvolvedor responsável por esta tarefa ficarão furiosos pela falta de visibilidade do processo de entrega de software.

Um importante aspecto na avaliação destes cenários é que, o código defeituoso, seja por conta de erro de algum erro de compilação ou pelos testes não passarem foram parar no servidor de versionamento. A descoberta dos erros é feita de forma tardia. Os erros ou falhas de um código problemático aparecem bem depois do que quando foram escritos.

O trabalho descrito neste relatório foi desenvolvido sobre a ótica do DESIGN THINKING no sentido de apresentar uma proposta de solução de problema identificado em pesquisa com foco na do desejo dos envolvidos, mas que também seja economicamente viável e exequível.

Ainda nesse sentido, há um direcionamento claro para a materialização prática do protótipo que vai ao encontro dos conceitos abordados pela cultura MAKER onde a construção de projetos e protótipos reais se materializa em aprendizado efetivo e real para própria evolução do projeto.

O desafio é estabelecer uma Estratégia centrada no código com Integração entre os processos. Assim, a questão de pesquisa deste trabalho é:

“Baseado em problemas de Gerência de Configuração de Software, como a maturidade do processo de desenvolvimento pode ser melhorada com a utilização Integração Contínua na equipe MAA da empresa CAIXA ECONOMICA FEDERAL?”

2. REVISÃO BIBLIOGRÁFICA

Integração Contínua é uma prática de desenvolvimento de software onde membros da equipe incorporam mudanças ao software frequentemente, utilizando processos de compilação e testes que asseguram a integridade do projeto (FOWLER, 2016). Segundo Durvall Paul um dos primeiros autores a utilizar o termo Integração Contínua foi Grady Booch, que atribuiu como sendo “um processo realizado em intervalos regulares, que resulta em um executável que incorpora o aumento funcional do sistema”, além disso ele sugere que este processo seria uma espécie de “marco de projeto onde a gerência poderia realizar medições para controlar melhor o projeto e atacar os riscos em uma base contínua” (PAUL, 2017). Com o advento das Metodologias Ágeis diversas técnicas e práticas que permitem dinamizar o desenvolvimento de software ganharam destaque. O conceito de Integração Contínua foi aprimorado (PAUL, 2015), passando a ser recomendado não apenas em intervalos regulares, mas sempre que possível, por viabilizar um ambiente coeso e propício para redução significativa de problemas de integração, além de atenuar uma das características mais críticas de software, a invisibilidade (FOWLER, 2015). A Integração Contínua é um passo inicial para uma outra prática fortemente encorajada, a Entrega Contínua (Continuous Delivery), que se trata de uma disciplina de desenvolvimento de software onde o sistema é construído de tal forma que pode ser liberado para produção em qualquer momento (FOWLER, 2015). Os principais benefícios de entregar continuamente são reduzir risco de deploy, melhoria na confiança do progresso do projeto (aumento de visibilidade) e adiantamento do feedback do usuário (antecipação de mudanças) (FOWLER, 2015). Atualmente diversas ferramentas promovem integração contínua identificando mudanças no

repositório do projeto, verificando o código e executando um conjunto de procedimentos para verificar se a mudança é boa e não irá prejudicar alguma parte do projeto.

Algumas ferramentas bastante difundidas para integração contínua são “Cruise Control”, “Continuum”, “Team City”, “Hudson” e “Jenkins”. Entre elas o Jenkins conseguiu maior alcance na comunidade open-source, consequentemente tem vantagem na identificação e correção de bugs, implementações de melhorias, bem como no desenvolvimento de plugins compatíveis (SMART, 2011). A ferramenta Jenkins é originária do mesmo projeto da ferramenta Hudson, cuja teve sua primeira versão disponibilizada em 2005. Alguns impasses relacionados aos direitos e decisões de projeto envolvendo a empresa Oracle culminaram na “separação” dos projetos e adoção do nome Jenkins em 2011 (JENKINS, 2011). Jenkins é um software livre e open-source para integração contínua, desenvolvido em Java. Além de monitorar, integrar (através de compilação e testes) e fornecer feedback sobre os projetos, ele permite que sejam customizados diversos procedimentos adicionais através de plugins como por exemplo o “SonarQube plugin”, que permite a execução remota de inspeções de qualidade de código utilizando o Sonar (JENKINS, 2015).

3. OBJETIVOS

3.1 Objetivo Geral:

Esta pesquisa propor melhorias para o processo de desenvolvimento de software utilizando os princípios da integração contínua.

3.2 Objetivos específicos:

Pesquisar sobre Integração contínua e Testes Automatizados.

Trazer melhorias ao processo de desenvolvimento de software da equipe de desenvolvimento do sistema MAA

Pesquisar possibilidade de prototipação com baixo custo utilizando um Raspberry PI como um protótipo de servidor.

4. MATERIAL E MÉTODOS

A identificação do problema objeto da pesquisa deve ser validada conforme a metodologia do Design Thinking junto à local definindo uma vez que a proposta de solução deve atender aos desejos reais dos envolvidos no projeto e em suas manutenções. Foram distribuídos formulários de pesquisa individuais, contendo questões voltada à equipe que realiza quinzenalmente uma retrospectiva dos trabalhos feitos e metas, pontos fortes e pontos de melhoria. Para coleta de dados foram realizadas 9 entrevistas do tipo estruturas qualitativas no próprio ambiente de trabalho com o objetivo de avaliar a percepção, dores e conhecimentos dos entrevistados e os problemas relacionados à qualidade das entregas realizadas. As respostas foram inseridas em uma planilha eletrônica para tabulação de dados cujas algumas perguntas são apresentadas abaixo:

1. Quanto tempo você trabalha no projeto MAA?
2. Qual a nota que você atribui para a qualidade do software que entregamos?
3. Possui conhecimento de integração contínua e testes automatizados?
4. Qual a versão que está em nossa branch master (principal) nesta data?
5. Você acha possível melhorarmos o processo?

A análise das pesquisas apontou uma equipe nova no projeto e que anseia e reconhece que melhorias podem ser realizadas com foco na qualidade. O conhecimento de integração contínua é bom, porém não se usa muito a estrutura legada. Apenas dois dos entrevistados sabiam qual a versão de produção do software na data. Todos foram muito colaborativos e abertos a melhoria inclusive fornecendo novas ideias.

4.1 EQUIPAMENTOS E FERRAMENTAS

Em busca de baixo custo e uso de softwares livres serão utilizadas as ferramentas:

- Hardware: Raspberry Pi 3 com Sistema Operacional Linux Debian tendo em vista é o que mais de aproxima da estrutura atual para construção de servidor de integração de modo a não afetar a estrutura existente. Abaixo os custos, configuração do Hardware RaspberryPi e representação visual do mesmo.

Raspberry Pi B	R$ 242,00
Cartão SD	R$ 20,00

- Configuração do hardware:
  - ✓ 1 GB DE RAM;
  - ✓ SAIDA HDMI E RJ43;
  - ✓ 4 ENTRADAS USB;MICRO SLOT SD;
  - ✓ 40 PINOS PARA CIRC ELETRÔNICO;
  - ✓ ENTRADAS PARA VIDEO, AUDIO, CAM.

Visão superior do Hardware Raspberry Pi B e seus componentes

- Softwares:

    a) Controle de versão: Apache Subversion e GIT;

    b) Servidor de automação: Jenkins;

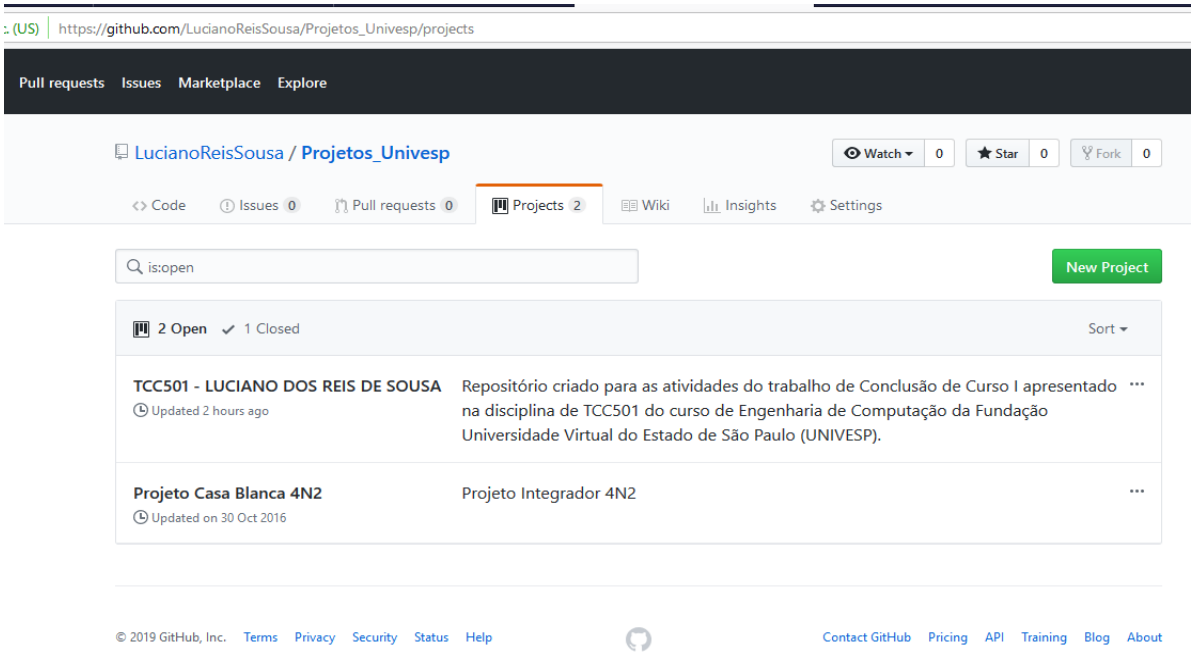
    c) Cobertura código**:** EclEmma, SonarLinte JUnit integrados ao Eclipse;

    d) Testes: SonarQube integrado ao Maven;

    e) Conexão remota ao servidor: WINSCP.

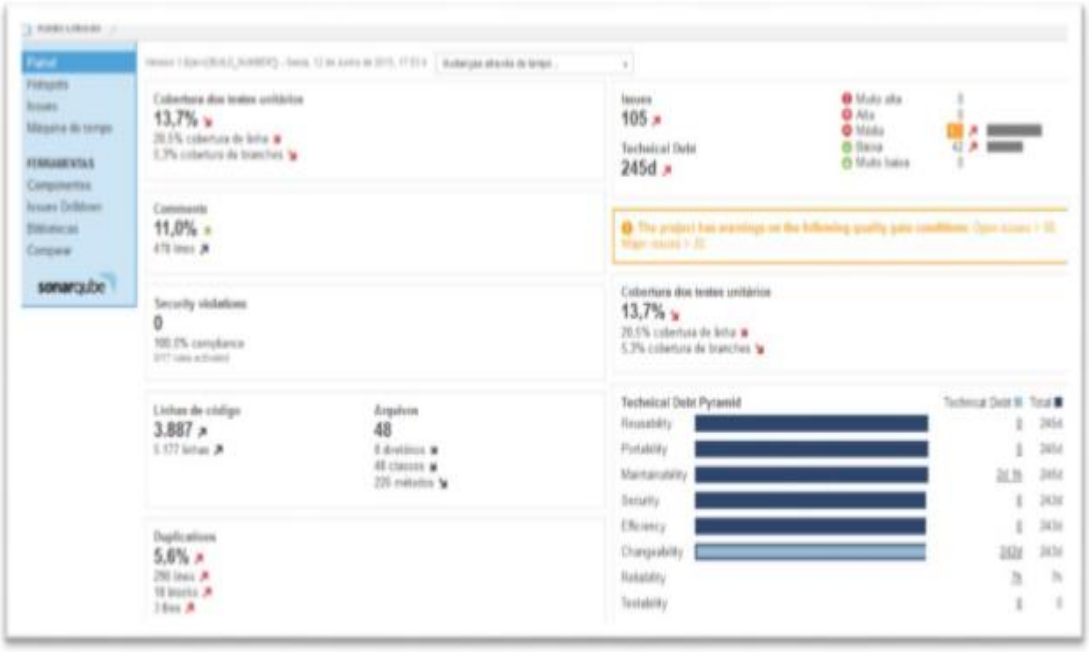
Foi criado um novo Projeto no GITHUB para simular o armazenamento e gerenciamento dos artefatos durante a construção desta proposta. Abaixo o endereço para baixar (clonar) o repositório:

https://github.com/LucianoReisSousa/Projetos_Univesp/projects



Visão do Repositório do Projeto criado no GITHUB

- Relatório esperado de inspeção contínua gerado pelo Sonar após realizados os testes unitários e testes automatizados:



Modelo de relatório de cobertura de testes gerado pelo SonarQube.

4.2 ANÁLISE DOS DADOS

Os relatórios antes e depois da implementação serão comparados para determinação da de implantação desta proposta.

5. CRONOGRAMA

Este projeto de pesquisa está estimado para ser executado em 2 meses para o TCC I e dois meses para o TCC II.

6. CONSIDERAÇÕES FINAIS

Espera-se que esta proposta traga melhorias para o processo de desenvolvimento e manutenção do software MAA, utilizando os princípios da integração contínua, software livre, e boas práticas de desenvolvimento de software.

7. REFERÊNCIAS BIBIOGRAFICAS

NEMETH, EVI. Manual Completo do Linux - São Paulo - SP: Pearson Makron Books, 2014.

DEITEL, H.M. Sistemas Operacionais - São Paulo - SP: Pearson Prentice Hall, 2015.

GIMENEZ, SALVADOR PINILOS. Microcontroladores: Teoria do hardware e do software Aplicações em Controle Digital Laboratório e Simulação - São Paulo - SP: Pearson Education do Brasil, 2012.

TANENBAUM, A. S., WOODHULL, A. S. Sistemas Operacionais: Projeto e Implementação. Porto Alegre: Bookman 2018.

Site UNIVESP, Leis Decretos e Portarias disponível no endereço: https://univesp.br/sobre-a-univesp

Site Raspberry, Documentação, disponível no endedereço: https://www.raspberrypi.org/

PRESSMAN, R. S. Engenharia de Software. [s.l.] McGraw Hill Brasil, 2011.

FOWLER, M. Continuous Delivery. Disponível em: <http://martinfowler.com/bliki/ContinuousDelivery.html>. Acesso em: 12 abril. 2019.

SOMMERVILLE, I. Software Engineering. 9. ed. [s.l.] Addison-Wesley, 2011.

SONAR. SonarQube Disponível em: <http://www.sonarqube.org/>. Acesso em: 18 abril. 2019.

HUMBLE, J. FARLEY, D. Entrega Contínua: Como Entregar Software. 2014.

JENKINS. Hudson's future | Jenkins CI. Disponível em: <http://jenkinsci.org/content/hudsons-future>. Acesso em: 12 mai. 2019.

JENKINS. Plugins. Disponível em: <https://wiki.jenkinsci.org/display/JENKINS/Plugins#Plugins-Buildreports>. Acesso em: 12 abril 2019

MELO, C. DE O.; FERREIRA, G. R. Adoção de métodos ágeis em uma Instituição Pública de grande porte-um estudo de caso. In: Workshop Brasileiro de Métodos Ágeis, Porto Alegre. Anais...2010. Disponível em: <http://agilcoop.org.br/files/WBMA_Melo_e_Ferreira.pdf>. Acesso em: 13 abril. 2019.

MOREIRA, A.: Integração Contínua. [s.d.]. Disponível em: <http://siep.ifpe.edu.br/anderson/blog/?page_id=1015>. Acesso em: 12 abril. 2019