FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE-FANESE
NÚCLEO DE PÓS-GRADUAÇÃO E EXTENSÃO – NPGE
ESPECIALIZAÇÃO EM MBA *PROJECT MANAGEMENT* 3.0
TRABALHO DE CONCLUSÃO DE CURSO–ARTIGO CIENTÍFICO

Formatado: Fonte: Itálico

RAPHAEL SILVA FONTES

DESENVOLVENDO SOFTWARE ENXUTO: Construindo aplicações sem equipe e com agilidade

Aracaju - SE
2017

RAPHAEL SILVA FONTES

DESENVOLENDO SOFTWARE ENXUTO: Construindo aplicações sem equipe e com agilidade

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Graduação e Extensão – NPGE da Faculdade de Administração e Negócios de Sergipe – FANESE, sob orientação da profª. Maria José de Azevedo Araújo como requisito para obtenção do título de Especialista em MBA *Executive Project Management* 3.0.

_____

~~Avaliador~~Hiram de Oliveira Costa-Silva

_____

Luciano Cerqueira Passos

_____

Raphael Silva Fontes

Aprovado com média: _____

Aracaju (SE), 17 de Julho de 2017.

LISTA DE FIGURAS

FIGURA 1 - PM CANVAS.....8

FIGURA 2 - CONSTRUÇÃO DO CANVAS.....9

FIGURA 3 - PROCESSO DO SCRUM.....10

FIGURA 4 - SCRUM SOLO.....11

FIGURA 5 - DIAGRAMA DE CONTROLE DE VERSÃO DISTRIBUÍDO.....13

FIGURA 6 - ENTRE DESENVOLVEDOR FRONT-END E BACK-END.....14

FIGURA 7 - DIFERENÇA ENTRE VIRTUALIZAÇÃO E CONTAINER.....16

SUMÁRIO

RESUMO 5

1. INTRODUÇÃO 6

2. CONHECENDO OS TIPOS DE SISTEMAS 6

3. MODELO VISUAL - GERÊNCIA DE PROJETO COM CANVAS 7

4. SCRUM 9

4.1. Scrum Solo 11

5. GERENCIAMENTO DE TAREFAS 11

6. CONTROLE NA VERSÃO DO CÓDIGO 12

7. QUAL LINGUAGEM ESCOLHER 13

8. ADMINISTRANDO AMBIENTES 15

9. CONCLUSÃO ..... 18~~16~~

REFERÊNCIAS ..... 19~~17~~

ABSTRACT ..... 20~~18~~

RESUMO

O objetivo deste artigo é demonstrar como a utilização de frameworks existentes, seja *de front-end* ou de *back-end*, juntamente com o aperfeiçoamento de metodologias ágeis e de gerenciamento de projetos, como o Canvas e o *Scrum*, poderão agilizar projetos relacionados ao desenvolvimento de softwares que, antes direcionado a uma equipe, poderão ser direcionados a uma pessoa, neste caso, um programador, que irá se tornar um desenvolvedor *full-stack*, afim de evitar desperdícios de tempo, de materiais e dependência de tarefas, ajudando a otimizar o desempenho nos projetos e sua eficiência nos resultados obtidos.

Palavras-chave: Frameworks; Aplicações, Projetos;

Comentado [Office1]: Colocar palavras em outro idioma em itálico

Formatado: Fonte: Itálico

Formatado: Fonte: Itálico

Comentado [HC2]: Não vi no artigo a integração entre as diversas ferramentas, frameworks, metodologias citadas. Também não entendi qual a sequencia de utilização entre elas, não vi as vantagens e desvantagens em utilizá-las.

Comentado [HC3R2]: Continuei com a dúvida acima.

1. INTRODUÇÃO

Software é qualquer conjunto de instruções que direciona o processador de um computador a executar operações especificas. Normalmente composto por diversas funções, bibliotecas e módulos que tornam o programa executável, recebendo algum dado de entrada, processando e retornando uma saída como resultado desse processamento, alguns desses programas podemos destacar o Word, Excel, navegadores como o Chrome, Firefox e Internet Explorer.

Existem vários tipos de software, cada um com sua peculiaridade e característica exclusiva, que depende da plataforma a qual foi desenvolvida, linguagem e também regra de negócio. O software de sistema é composto por informações e instruções processadas pelo sistema interno que permite que o usuário interaja, como um Sistema Operacional. Já o Software de programação, também conhecido como IDE (Integrated Development Environment), auxiliando o programador a desenvolver softwares, juntamente com a linguagem de programação. Por último há o Software de aplicação que é composto por programas de computadores que permitem ao usuário executar uma, ou série, de tarefas específicas de várias áreas, como engenharia, medicina, pedagogia, urbanismo, administração e etc.

2. CONHECENDO OS TIPOS DE SISTEMAS

Atualmente, há um crescente no desenvolvimento em software de aplicação, estes são mais conhecidos e estão cada vez mais presentes cotidiano das pessoas, seja uma rede social como Facebook, Twitter, um buscador como Google ou Bing, até nas compras como Submarino e Mercado Livre. Muitos desses softwares foram desenvolvidos com o propósito de solucionar problemas ou melhorar situações de uma ou mais pessoas.

As pessoas utilizam o software como produto, independente se estão online ou não, podendo ser um e-mail, comércio-eletrônico, pagamento, gestão de conteúdo, educação, comunicação, ERP e etc. Os softwares como produto podem ser classificados como três tipos:

Para consumidor final: o produto resolve um problema para o consumidor final que está a decidido a pagar um valor para utilização dos serviços, como por exemplo Netflix, Spotify e jogos.

Para empresas: neste caso o produto resolve o problema para as empresas, que tem disposição para pagar um valor maior que os consumidores finais, alguns exemplos são: Google AdWords, SAP, AutoCad e o pacote Adobe.

Mistos: por último o produto resolve um problema tanto para o consumidor final quanto para as empresas, normalmente estas que pagam os custos de manutenção do software. O modelo mais comum de geração de receita neste tipo de produto é o anúncio, pago pela empresa.

Com a divisão dessas categorias, fica claro para qual tipo de público o produto deverá atingir, e isso é muito importante pois todo desenvolvimento terá um custo, independentemente da acessibilidade, haverá custos, logo é necessário gerar receitas para que ele cumpra sua missão de resolver o problema.

Atualmente muitas plataformas ganharam nome e espaço no mercado, como Uber, AirBnb e o BitCoin, e o que determina se uma plataforma é valorizada ou não é a quantidade de usuários e a classificação que os mesmos dão para ela.

O software possui um ciclo de vida e esta deve ser feita no processo de construção do produto. A partir disso, deverá ser definido quais as formas de interação com o cliente, desde o recebimento e elaboração do escopo até a versão operacional do sistema.

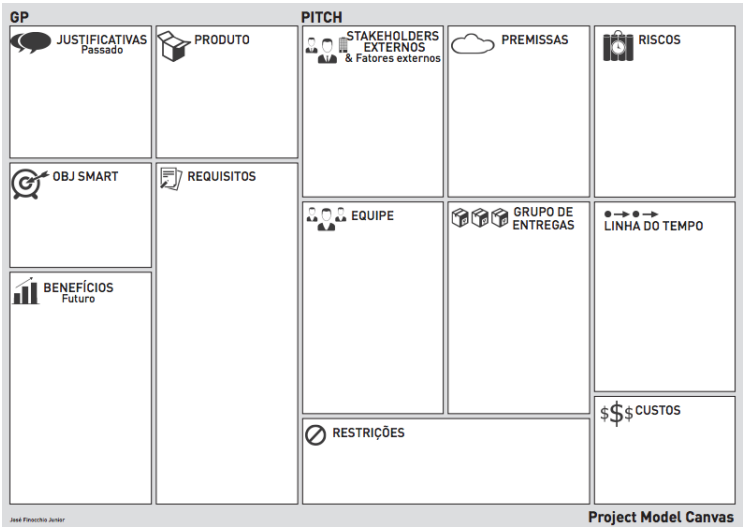
Dentre os modelos do desenvolvimento de software, o mais atrativo e que se encaixa no cenário abordado é o iterativo, que tem a característica de entregar pequenos pacotes, utilizáveis, do produto para o cliente e que este possa dar seu feedback, havendo assim um grande envolvimento entre as partes.

3. MODELO VISUAL - GERÊNCIA DE PROJETO COM CANVAS

Existe uma metodologia visual para o gerenciamento do projeto, denominada PM Canvas (Project Model Canvas), criada por José Finocchio Júnior, a qual facilita a construção do produto sem inúmeros documentos, ideal para o cenário abordado.

O PM Canvas concentra-se no essencial do projeto, e permite que as partes interessadas participem a todo momento no desenvolvimento do produto.

Figura 1 - PM CANVAS



Fonte: http://www.pmcanvas.com.br/download/

O PM Canvas utiliza conhecimentos da neurociência, e é uma agenda sobre a qual os stakeholders irão se debruçar para conceber a lógica do projeto. Toda informação deverá ser colocada em post-its, para que as informações nele contida possam ser curtas, essenciais e pragmáticas.

A construção do Canvas divide-se em 4 etapas. A primeira é denominada de conceber, a qual deverão ser respondidas 5 perguntas: Por que? O que? Quem? Como? Quando e Quanto?.

A segunda etapa é a de integrar, garantindo a consistência entre os blocos e a interação entre os componentes.

A terceira etapa, denominada resolver, é a identificação dos pontos que obstruíram a montagem do Canvas, por causa de indefinições, informações ou contradições.

Por último a etapa da comunicação, que é o processo que servirá como base para gerar outros documentos, sejam apresentações, cronogramas ou até mesmo planos do projeto.

Figura 2 - CONSTRUÇÃO DO CANVAS

Fonte: http://blog.mundopm.com.br/2013/06/05/pm-canvas/

Após o preenchimento do Canvas, muita coisa poderá ser descoberta do projeto, novas perspectivas. Fragmentos de informações que os stakeholders poderiam não conhecer irão ser apresentáveis através do modelo. Juntamente com o modelo de desenvolvimento, uma metodologia de trabalho deverá ser adotada para garantir o cumprimento do prazo e organização de todas as etapas do ciclo.

Para o gerenciamento das entregas, localizado no bloco "Grupo de Entregas", uma metodologia de gestão de tarefas deverá ser aplicada.

4. SCRUM

Scrum é uma ~~metodologia agil~~ferramenta de gestão e planejamento de projetos. Nele os projetos são divididos em Sprints, são ciclos que possuem duração iguais e há um conjunto de atividades que devem ser executadas a fim de finalizar as entregas que foram determinadas na ~~reunião da~~reunião da Sprint Backlog.

Comentado [Office4]: SCRUM é uma metodologia?

Comentado [RF5R4]:

Comentado [HC6]: Scrum é uma metodologia ágil? Uma ferramenta de gestão e planejamento ou um framework?

Comentado [HC7]: Colocar palavras em outro idioma em itálica. Verificar em todo o texto.

A cada dia durante a Sprint a equipe faz uma reunião denominada de Daily Scrum, que tem como objetivo nivelar o conhecimento sobre o que foi feito no dia anterior e identificar qualquer obstáculo. Para manter o engajamento com a equipe, as reuniões são realizadas no mesmo horário e local.

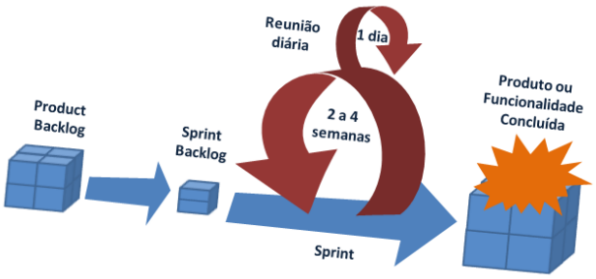
Todas as funcionalidades que serão entregues na finalização do projeto são mantidas em uma lista que é denominada de *Product Backlog*. Essa lista é definida, e mantida, pelo *Product Owner*, além de todas as priorizações das tarefas, e não precisa estar completada no início do projeto, pois com o amadurecimento do projeto novas tarefas serão adicionadas e outras que foram vistas anteriormente poderão ser removidas.

Para garantir que seja entregue tudo aquilo que foi solicitado pelo *Product Owner*, o Scrum Master deve fazer com que a equipe siga as instruções e etapas, além de assegura-la que não haja o comprometimento excessivo.

A equipe de desenvolvimento, denominada de Scrum Team, não há necessariamente uma divisão funcional através de papéis, como programador, designer, tester ou arquiteto, todos trabalham juntos com o objetivo que foi estabelecido na Sprint.

Já o Sprint Backlog é uma lista das tarefas que o time de desenvolvimento se compromete a entregar na Sprint, sendo extraídas do *Product Backlog* com base na priorização feita pelo *Product Owner*.

Figura 3 - PROCESSO DO SCRUM

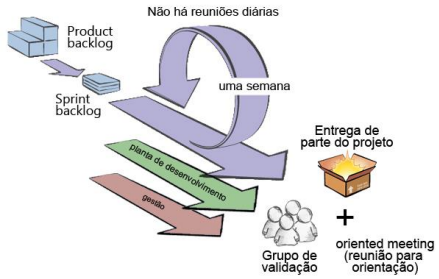


Fonte: http://www.mindmaster.com.br/scrum/

4.1.Scrum Solo

Diferente do Scrum tradicional, o Scrum Solo é utilizado por quem não tem equipe de trabalho. Uma das características desse aperfeiçoamento é a ausência das reuniões diárias com equipe, esta é substituída por uma reunião com o grupo de validação, seja clientes ou usuários finais, a cada entrega da Sprint. O segredo é definir o que é essencial, o que vai ser entregue em cada etapa e sempre com o objetivo de otimizar as entregas e a cada termino de uma Sprint, é necessário que haja uma ata a qual deve registrar a validação da implementação juntamente com o usuário.

Figura 4 - SCRUM SOLO



Fonte: https://engenhariasoftware.wordpress.com/2016/04/17/scrum-solo-2/

5. GERENCIAMENTO DE TAREFAS

A boa gestão das tarefas traz benefícios, pois auxiliam na solução de problemas, identificam defeitos, ajudam na tomada de decisões e economizam tempo, ou seja, é um complemento para a performance e otimização do processo de desenvolvimento do software.

Esses sistemas ~~fornecem~~ auxiliam na obtenção de estimativas realistas, ~~diante~~desde que os dados sejam inseridos corretamente, ~~se corretamente alimentados,~~ como o registro das horas dedicadas a cada tarefa, auxiliando na determinação do tempo para novas tarefas de forma precisa. Também aumenta o foco no trabalho, pois existe o cronometro do início da tarefa até a sua finalização e tudo é contabilizado no deadline.

Comentado [HC8]: Já vem pré-cadastrado a estimativa? Como funciona? Na minha visão, softwares auxiliam no controle e auxiliam e apoiam as estimativas, não as fornecem.

Além disso, o software manterá registros de todos os problemas e dificuldades, posteriormente sendo uma base de conhecimento, auxiliando nos próximos projetos na resolução de problemas recorrentes.

É necessário o conhecimento da rotina do trabalho e saber como organizá-la de forma produtiva. Existem softwares online e gratuitos quem podem auxiliar, como o Asana, que possui a criação de tarefas e sub-tarefas ilimitadas, além de uma interface simples e leve dá uma visão geral das tarefas, possui um armazenamento na nuvem e que pode ser ligado com algumas plataformas de espaço na nuvem como Google Drive e Dropbox, além de possuir aplicativos para celulares e é traduzido para o português.

6. CONTROLE NA VERSÃO DO CÓDIGO

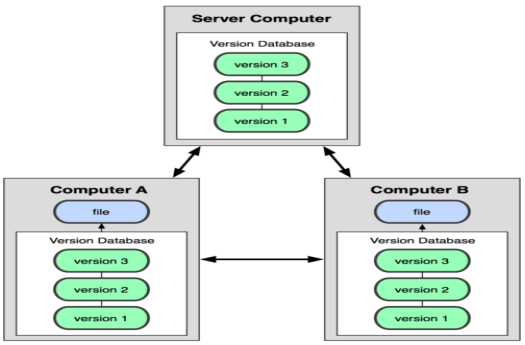
Para o gerenciamento do código fonte do software, um sistema de controle de versão é primordial, pois registrará as alterações feitas num arquivo ou o conjunto ao longo do tempo de trabalho, podendo retomar um ponto do projeto e criar novos caminhos de desenvolvimento sem que atrapalhe o caminho principal.

Existem três tipos de sistemas de controle de versão, os locais, centralizados e distribuídos.

Para o cenário abordado, a utilização do controle de versão distribuído é essencial, pois o desenvolvedor terá uma cópia completa do repositório no computador, assim, se houver alguma falha no servidor, o desenvolvimento do produto não haverá atrasos. A cópia completa fará com que o projeto seja recuperado no momento onde parou, e toda alteração que seja feita após esse reparo será de modo incremental.

Comentado [HC9]: Essa cópia é completa?não pode ser só o diferencial? Ou ainda só o que está permitido ele salvar?

Figura 5 - DIAGRAMA DE CONTROLE DE VERSÃO DISTRIBUÍDO



Fonte: https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o#Sistemas-de-Controle-de-Versão-Distribuídos

Um dos sistemas mais conhecidos para o controle de versão distribuído é o Git, possuindo velocidade, design simples, suporte robusto a desenvolvimento não lineares (milhares de branches em paralelo), totalmente distribuído e capaz de lidar eficientemente com grandes projetos.

O Git não trata os armazenamentos das informações modificadas como uma lista de mudanças por arquivos e sim como um conjunto de snapshots (captura de um determinado instante do objeto). Cada vez que a modificação é salva (commited), o Git retira uma "foto" desse estado e armazena junto com ele uma referência, e se nenhum arquivo foi modificado, a informação não é duplicada.

~~Com~~ Diante do~~a base de~~ gerenciamento e controle do produto ~~já definidos~~demonstrados no artigo, ~~será necessário definir~~ o próximo passo é entender qual a plataforma de desenvolvimento, tecnologias e linguagens deverão ser escolhidas para o projeto.

Comentado [HC10]: Foi definidos aonde?

7. QUAL LINGUAGEM ESCOLHER~~??~~

Existem centenas de linguagens de programação no mundo, cada uma com sua característica. Da mesma forma que a linguagem humana, as linguagens de programação são utilizadas para se comunicar, são compreendidas perfeitamente pelo computador, graças a ajuda

de intéerpretes, compiladores ou de outros tipos similares de software. Intérpretes são programas que leem o código fonte e na medida que a leitura está sendo feita o programa já é executado, à exemplo da linguagem de programação PHP.

Dependendo do projeto, uma linguagem de programação servirá melhor que outra, por ser mais simples, ou por ter uma comunidade mais ativa, com uma boa documentação e com frameworks atualizados e poderosos.

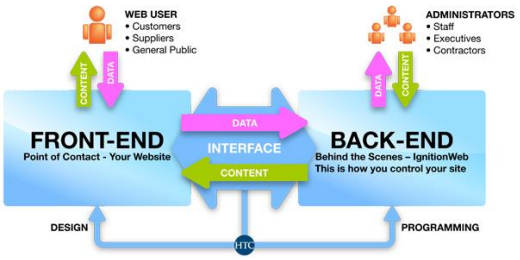
Para auxiliar o trabalho do programador, se faz necessária a utilização de framework que é um conjunto de ~~códigos~~funcionalidades, ~~com funcionalidades~~sendo genéricas, que facilitam o trabalho do desenvolvedor.

As tarefas do dia-a-dia são simplificadas, pois existem módulos implementados, como uma consulta ao banco de dados, além de uma padronização, que facilita o desenvolvimento e dá um fluxo de trabalho dentro do framework, ganhando assim velocidade.

Existem dois tipos de ~~papéeis~~atividades no desenvolvimento do software, o *front-end* que é responsável pela parte de interação com o cliente que irá utilizar o produto, interagindo com a interface. Já o *back-end* é responsável por dinamizar o produto através das linguagens de programação, com ou sem utilização de frameworks ou banco de dados.

Cada um desses papeis são importantes para a construção dos produtos e estes possuem uma grande dependência, pois não há interação com o código sem as entradas do cliente a partir do front nem a saída de um resultado processado sem o *back-end*.

Figura 6 – DIFERENÇA ENTRE DESENVOLVEDOR FRONT-END E BACK-END



Fonte: https://git-scm.com/book/pt-br/v1/Primeiros-passos-No%C3%A7%C3%B5es-B%C3%A1sicas-de-Git#Snapshots,-E-Não-Diferenças

Comentado [HC11]: O que são intérpretes?

Comentado [HC12]: Segundo quem? Isso é uma citação, deve ser referenciado corretamente.

Comentado [HC13]: Framework é um conjunto de códigos? O SCRUM é um framework, então o SCRUM é um conjunto de códigos?

Comentado [HC14]: São papéis?

Formatado: Fonte: Itálico

A rotulação desses papéis com seus atuantes, desenvolver *front-end* ou desenvolver *back-end*, é comum, pois são dois profissionais que cuidam das camadas de cliente e servidor, respectivamente, além disso, cada papel possui tecnologias especificas, como por exemplo HTML, CSS e JS para o *front-end* e o banco de dados (SQL, MySQL, Oracle), além das linguagens de programação, Python, PHP, Java, C#, Swift ~~e~~ etc~~,~~ para o *back-end*. No cenário abordado, não teria como haver um outro papel, além de que, o mercado irá exigir o conhecimento tanto de *front-end* quanto de *back-end* e para isso o programador se tornará um desenvolvedor *full-stack*.

Formatado: Fonte: Itálico

Desenvolvedor *full-stack* é aquele que consegue preencher todos os espaços no desenvolvimento do software, não somente com as linguagens dos papéis anteriormente citados, mas também com o conhecimento em sistemas operacionais, ferramentas de controle de versão, ferramentas de design, experiência do usuário (UX), webservices, além do conhecimento em padrões de arquiteturas de desenvolvimento.

~~Para facilitar o trabalho do desenvolvimento, existem tanto frameworks para o *front-end*, quanto para o *back-end*.~~

Formatado: Fonte: Itálico

Comentado [HC15]: Fiquei na dúvida do que é front-end e back-end. Qual a diferença? São ambientes?visões?framework?qual as principais características de cada um?

Formatado: Fonte: Itálico

Um dos frameworks mais utilizados atualmente é o Bootstrap, que provê componentes do *front-end* (CSS, SASS, LESS e JS) prontos para serem utilizados no projeto. Ele facilitará a implementação de componentes no HTML, podendo ser estilizados, como botões, tipografia, formulários, ícones e tabelas, além da inserção de novos como modais, *dropdown*, *popover*, *tolltips* e etc, assim não sendo necessário reinventar a roda. Este framework possui uma comunidade extremamente ativa, é livre e possui atualizações constantes.

8. ADMINISTRANDO AMBIENTES

Ao iniciarmos um projeto, há a necessidade de isol~~á~~á~~-~~lo dos outros para que não haja disparidade ou impacto entre as versões das bibliotecas, linguagem, framework e até banco de dados. Essa desigualdade é devida a quantidade de projetos de software que uma máquina pode ter, cada um com sua versão da linguagem de programação, bibliotecas e até versões do framework diferentes, independente do sistema operacional. O que pode acontecer é ao instalar uma biblioteca ou plugin, este pode afetar a funcionalidade de outro já existente ou até sobrescreve-lo.

Comentado [HC16]: Porque aconteceria essa disparidade? Não tem softwares que controle paralelamente, juntos ou isolados? Isto é um ponto negativo de qual sistema? Existem pontos positivos?

Comentado [HC17]: Segundo quem? Isso é uma citação? deve ser referenciado corretamente.

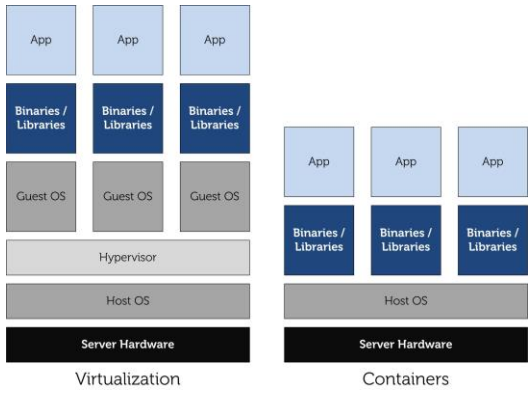
Docker é uma plataforma gratuita, desenvolvida na linguagem Go dentro da empresa Google, que facilita na criação e administração de ambientes isolados.

Nele é possível encapsular uma ampliação ou um ambiente inteiro dentro de um container, após isso é possível transportá-lo para qualquer hospedagem que contenha a plataforma. Isso reduz drasticamente o tempo de deploy, que é o tempo até disponibilizar o serviço, pois não há necessidade da configuração que sempre é feita para subir o ambiente.

Outra facilidade do Docker é a possibilidade na criação de imagens, arquivos estáticos, que automatização a criação da imagem da máquina.

Um container é tudo que faz com que o software rode como um pacote isolado. Diferente das máquinas virtuais comuns (VM's), um container não é um pacote completo do sistema operacional, é apenas bibliotecas e configurações requeridas para a execução do software. Sua imagem é leve, independente e enxuta.

Figura 7 - DIFERENÇA ENTRE VIRTUALIZAÇÃO E CONTAINER



Fonte: http://www.mundodocker.com.br/o-que-e-docker/

Para os desenvolvedores, Docker automatiza atividades repetitivas como a configuração do ambiente de desenvolvimento, fazendo com que seja focado apenas na elaboração do software. Não há necessidade de instalar e configurar complexos banco de dados ou estar trocando entre versões da linguagem de programação. Quando a aplicação está dentro do Docker, a complexidade é colocada dentro do container e assim a construção, compartilhamento e execução são facilitados.

~~11.~~ 9. CONCLUSÃO

O mercado atual busca ganhos em produtividade através da redução de custos, onde cada vez mais os profissionais e as organizações precisam aliar o tempo, habilidades e competências ao seu favor. É importante compreender os benefícios da implementação de desenvolvimento de projetos através de softwares enxutos, para evitar desperdícios, atendendo as necessidades dos clientes da maneira mais simples possível, com um menor valor, e utilizando de todos os recursos disponíveis, trazendo assim o melhor custo benefício para seus clientes e lhes oferecendo maior valor.

REFERÊNCIAS

TORRES, Joaquim. **Gestão de Produtos**: Como aumentar as chances de sucesso do seu software. São Paulo: Casa do Código, 2015.

Desenvolvimento Ágil. **SCRUM**. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 10 Jun. 2017.

~~Desenvolvimento Ágil. **SCRUM**. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 10 Jun. 2017.~~

Melhor Desempenho com um gerenciador de tarefas. Disponível em: <https://blog.runrun.it/melhor-desempenho-com-um-gerenciador-de-tarefas-2/>. Acesso em: 17 Jul. 2017.

PAGOTO, Tiago; FABRI, José Augusto; L'ERARIO, Alexandre; GONÇALVES, José Antônio. **Scrum Solo**. Disponível em: <https://engenhariasoftware.files.wordpress.com/2016/04/scrum-solo.pdf>. Acesso em: 25 Jun. 2017.

JUNIOR, José Finocchio. **Project Model Canvas**: Gerenciamento de Projetos Sem Burocracia. São Paulo: Elsevier, 2013.

SUTHERLAND, Meet Jeff; SCHWABER, Meet Ken. **SCRUM GUIDE**. Disponível em: <http://www.scrumguides.org/>. Acesso em: 15 Jul. 2017.

ABSTRACT

The purpose of this article is to demonstrate how the use of existing frameworks, either front-end or back-end, along with the improvement of agile methodologies and project management, such as Canvas and Scrum, can streamline development-related projects of software that, before directed to a team, can be directed to a person, in this case, a programmer, who will become a full-stack developer, in order to avoid wasted time, materials and task dependency, helping to optimize the performance in the projects and their efficiency in the results obtained.

Keywords: Frameworks; Applications, Projects;