Paulo Caroli, Rafael Magrin & Taina Caetano

DEVOPS

para entrega de produtos enxutos

DevOps para entrega de produtos enxutos

Taina Caetano, Paulo Caroli, eRafael Magrin

Esse livro está à venda em http://leanpub.com/devopsmvp

Essa versão foi publicada em 2015-12-22

Conteúdo

Introdução

DevOps para produtos enxutos é formado por uma série de práticas e técnicas que nós utilizamos na fase de execução de projetos. Após a realização da fase de concepção, a equipe possuirá em mãos uma série de requisitos descrevendo o que deve ser o produto.

Em Direto ao Ponto[1], onde é proposta a técnica da Inception Enxuta, o resultado é o Canvas MVP[2], o qual é o ponto central do modelo de governança. Ele auxilia na organização e visualização das funcionalidades e da sequência de liberação de entrega incremental do produto mínimo viável, o MVP. Mais do que isso, o canvas organiza e planeja entregas do produto além do primeiro MVP.

Aqui, discutiremos os passos subsequentes à concepção, de tal forma que o leitor saia equipado para planejar, direcionar e executar o desenvolvimento do produto.

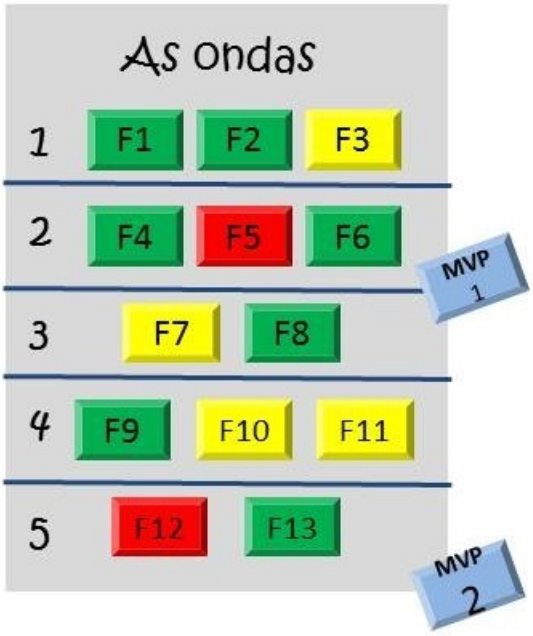
[1] https://leanpub.com/diretoaoponto

[2] http://www.caroli.org/o-canvas-mvp/

Pensamentos Iniciais

Resultado da Concepção

Na concepção do projeto, a equipe trabalhou junta para entender os objetivos do produto, os principais usuários, e o escopo funcional de alto nível. Além disto, a equipe sai da concepção com o sequenciador de features em mãos, o qual organiza e planeja entregas do produto além do primeiro MVP. Tipicamente, times utilizando o sequenciador de features irão deslumbrar e evolução do produto via uma sequência de MVPs. E isso auxilia, e muito, na criação de produtos, baseados nesta forma enxuta de trabalho.



MVP e suas features

Embora uma ferramenta eficiente para descrever e alinhar visualmente o mapa de evolução do produto, o sequenciador de features por si só não resultará em entregas de sucesso.

Independente de excelentes planos, se a equipe não estiver muito coesa e trabalhando efetivamente como um time, a execução do plano fica comprometida.

Para tanto, sugerimos fortemente que o time envolvido na concepção do produto seja o mesmo time envolvido na implementação, caracterizando assim o time do produto.

Uma equipe de produto é formada por um grupo cross-funcional multidisciplinar de pessoas que são coletivamente responsáveis para a entrega e manutenção de um produto. Normalmente, a equipe do produto desempenha um papel crucial dentro da organização: a equipe é responsável pela implementação da estratégia, evolução, definição das funcionalidades, criação e manutenção do produto.

Atividades Iniciais

Antes da equipe colocar a mão na massa e começar a implementação do produto, recomendamos uma checagem para os seguintes fatores. Alguns são logísticos, outros envolvem trabalho individual e em equipe para que estejam prontos:

- Uma mesa onde todos poderão se sentar juntos;
- Computadores e monitores;
- Licenças para os softwares necessários;
- Software de desenvolvimento instalado.
- Um quadro visual demonstrando as tarefas do time;
- Acesso ao sistema de controle de versão de código à todos os membros do time;
- Decisão sobre as tecnologias utilizadas para o MVP 1.
- Pipeline básico de entrega contínua funcionando.

Disponibilizar um quadro visual descrevendo as tarefas do time é importante para dar visibilidade, tanto para stakeholders externos quanto para os próprios membros do time se manterem alinhados com relação ao que deve ser feito. As próprias tarefas deste momento inicial podem (e devem!) ser descritas no quadro, com responsáveis associados à elas conforme elas são executadas.

Outro fator que deve ser garantido antes do início do desenvolvimento do produto é a definição das tecnologias que serão utilizadas. Não é necessário, neste momento, definir toda uma arquitetura e tentar antecipar todos os problemas que podem vir a acontecer durante o desenvolvimento. Entretanto, uma direção inicial se faz necessária para que o time comece.

A Sprint Zero

Para times usando a metodologia Scrum, é comum o termo sprint Zero. Apesar de não começarmos a contar do zero (contamos 1, 2, 3...), times Scrum comumente se referem a primeira sprint como Sprint Zero, sendo essa a Sprint dedicada a atividades de set-up, as as quais não contemplam tarefas diretamente relacionadas a criação de features do produto.

Independente de utilizar Scrum ou outra metodologia para a gerenciamento e acompanhamento da criaçao do produto, sugerimos utilizar este conceito de dedicar um tempo inicial (uma ou duas semanas) para realizar as tarefas iniciais preparatórias para o ambiente de trabalho.

Comece logo

A sessão tarefas de set-up compartilha uma sequencia de atividades para levantar as tarefas para o mínimo necesário de set-up para começar o trabalho. O número de tarefas levantadas vai variar de time para time. O perigo de levantar muitas tarefas, é que levaria muito tempo até começar o trabalho nas features do MVP, gerando, assim, ansiedade para todos que esperam o primeiro MVP.

Portanto, consideramos a combinação da ideia de Sprint Zero com a tarefas de set-up, e com isso, deixar claro a todos que o time não vai trabalhar em features do MVP neste curto período inicial (uma ou duas semanas). E, da mesma forma que o time pede por esse período inicial para set-up, ele se compromete a trabalhar efetivamente nas features do MVP após esse Sprint Zero.

Ou seja, depois do zero, o time vai começar a contar; sejam semanas ou sprints (para quem utilizar Scrum). E assim que começar a contar, estará trabalhando efetivamente nas tão esperadas features do MVP.

Tarefas de set-up

Para dar início a criação do produto precisamos de definições básicas sobre o ambiente de trabalho e sobre as normas de interação entre as pessoas. Para tanto, precisamos buscar o alinhamento e definição das mesmas.

Na mesma linha de pensamento que utilizamos para a definição de MVP, precisamos definir o mínimo necessário em relação ao ferramental e ao ambiente de trabalho para que possamos começar a construir o produto. Reunindo todos membros do time em uma sala, realizamos um workshop de duração média de duas horas, onde são realizadas a seguinte sequência com três atividades: Moscou, Votação Mais / Menos e Quem/O que/Quando, atividades detalhadas em Fun Retrospectives[3].

Atividade: Moscou

A atividade Moscou auxilia na conversa de alinhamento sobre o que é essencial para começarmos, em relação ao que pode vir depois.

Passo a passo da atividade

1. Divida o canvas em três sessões: precisa ter (Must Have), deveria ter (Should Have) e poderia ter (Could Have). O nome da atividade é Moscou pois lembra, em Inglês, MustShouldCould, ou, cortando algumas letras, MuShCou.
2. Selecione três cores de post-it explique a categoria de cada cor. Por exemplo: amarelo para comunicação, verde para pipeline de entrega contínua, e rosa para ambiente de desenvolvimento.
3. Distribua post-its coloridos e canetas a todos participantes.

[3] http://www.funretrospectives.com

4. Peça que cada participante escreva no post it (usando a cor de acordo com a categorização adequada) o que ele acredita ser importante para o time.
5. Peça aos participantes que coloque os posts no canvas comum de acordo com o seu entendimento em relação ao que precisa, deveria, ou poderia ser feito.

As duas atividades a seguir complementam a atividade Moscou para buscar o alinhamento do time para o iniciar a criação do produto. Sugerimos fortemente descartar os posts da sessão de *poderia ter*. Relembramos que o objetivo principal é buscar o mínimo de alinhamento para dar início a criação do produto. Portanto, o time deve focar sua atenção para a sessão *deveria ter*, ou, em outras palavras, o que é essencial para começarmos a trabalhar efetivamente.

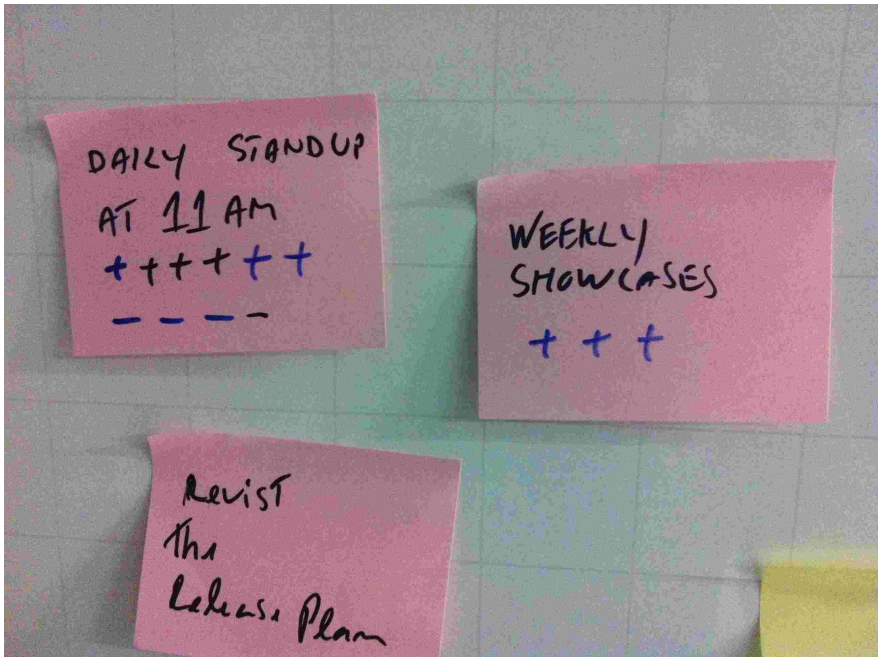
Atividade Votação Mais / Menos

Esta atividade permite aos participantes serem mais claros sobre concordâncias e discordâncias sobre os itens levantados. É tipicamente usada para focar a conversa nos itens que mais interessam ao grupo.

Passo a passo da atividade

1. Instrua os participantes sobre as regras de votação:
    - Cada participante tem direito a votar 5 “+” e 3 “–” (cada voto será representado por uma marca de “+” ou “–” no post-it.
    - Os participantes podem colocar mais de um voto em um cartão.
    - (+) representa sua concordância com uma anotação e você quer falar sobre ela.
    - (–) representa sua discordância com a anotação e você quer falar sobre ela.

“Por favor, vote nos itens que você quer discutir. Os itens com mais votos serão selecionados primeiro.”



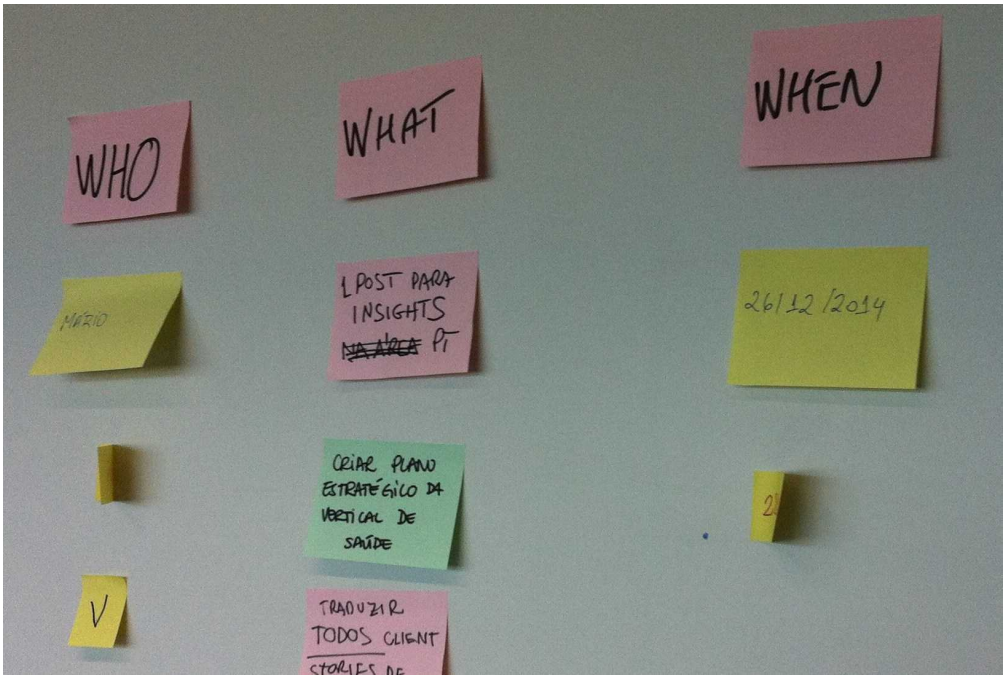
Votação Mais Menos

Atividade: Quem/O que/Quando

A atividade Quem/O que/Quando ajuda a definir comprometimento e ações de acompanhamento para os itens levantados. A partir desta atividade, os "próximos passos" ou "itens de ação" estarão claramente definidos e visíveis para o time.

Passo a passo da atividade

1. Crie uma estrutura de tabela que tenha como títulos de coluna QUEM / O QUE / QUANDO.
2. Peça para os participantes selecionarem (da atividade anterior) ou escreverem um passo concreto com o qual possam se comprometer. Estes deve ser tanto (1) passos que eles deve seguir; ou (2) passos nos quais acreditam bastante.
3. Cada passo concreto selecionado formará uma linha na tabela Quem/O que/Quando. Peça para os participantes para:
    - Colocar o post-it com o passo na coluna O QUE;
    - Escrever seus nomes na coluna QUEM;
    - Definir QUANDO o item estará feito.



Quem/O que/Quando nas colunas da tabela

Ao focar a discussão no formato Quem/O quê/Quando, você pode conectar pessoas a ações claras que elas definiram e com as quais se comprometeram. Ela permite aos participantes serem claros em relação a seu comprometimento e responsabilidades, tornando visível para todo o grupo QUEM vai fazer O QUE e QUANDO.

Esta atividade é inspirada na matriz Quem/O que/Quando no livro Gamer Storming[4].

Desenvolvimento Iterativo

Acabou a semana da inception enxuta, e agora o time tem o entendimento sobre o MVP e está alinhado sobre qual são as primeiras features a serem criadas.

As features e o MVP estão em alto nível, mostrando claramente o que tem ser feito, entretanto sem os detalhes sobre como isso vai ser feito. O time agora precisa aprofundar nos detalhes de como serão criadas essas features.

Para tanto o time deve utilizar de algumas dentre várias técnicas que auxiliam nesse entendimento mais detalhado. Histórias do usuário, prototipação rápida, desenvolvimento baseado em comportamento (BDD) e testes de aceitação são exemplos de algumas das técnicas para ajudar com o detalhamento das features.

Usando algumas dessas técnicas o time estará aprofundando sobre a forma com a qual as features serão criadas. Entretanto, aprofundar não é complicar! Fique muito atento para não estar adicio-

[4] http://gamestorming.com/

nando alguma técnica de detalhamento que estará complicando e distanciando o entendimento em relação as features e o MVP.

Ao primeiro sinal de que alguém está trabalhando em algo que não está vinculado ao MVP, pare e reavalie. Provavelmente estamos complicando algo desnecessariamente.

Mantenha sempre o canvas MVP próximo ao time. Sempre que alguém explicar alguma tarefa ou algo que esteja fazendo, aponte para o canvas MVP. Pergunte como tal tarefa está relacionada com o MVP.

Uma boa resposta ajuda todos a entenderem como tal tarefa está aprofundando sobre o que está sendo feito para o MVP. Uma resposta confusa pode ser um bom indicador de um complicador. Cuidado, aprofundar não é complicar!

Composição da Equipe

.

Práticas de Kanban/Scrum

.

Prototipação Rápida

.

MVP, features e histórias do usuário

O canvas MVP retrata a relação entre o MVP e as features. As features e o MVP estão em alto nível, mostrando claramente o que tem ser feito, entretanto sem os detalhes sobre como isso vai ser feito. O time agora precisa aprofundar nos detalhes de como serão criadas essas features.

As features são tipicamente descritas em um nível mais elevado do que as histórias do usuário. Portanto, antes de começar a trabalhar em uma feature, a mesma deve ser analisada e detalhada em suas respectivas histórias de usuário.

Feature é a descrição de uma ação ou interação de um usuário com o produto. Por exemplo: imprimir nota fiscal, consultar extrato detalhado, e convidar amigos do facebook. A descrição de uma feature deve ser o mais simples possível. O usuário está tentando fazer uma coisa. O produto dever ter uma feature para isso. Que feature é essa?

Exemplo de Funcionalidade: Consultar partidas sem geolocation

Tipicamente, as equipes de desenvolvimento trabalham com histórias de usuários. Portanto, você deve detalhar um pouco mais, e realizar o mapeamento de features para histórias.

História de usuário

História de usuário é um formato textual para a descrição concisa de um requisito que busca responder a três indagações básicas: quem? o que? e por que? Tipicamente uma funcionalidade de alto nível (feature) é desmembrada em algumas histórias do usuário.

Como um PERFIL DO USUÁRIO / PERSONA

Eu quero FUNCIONALIDADE ESPECÍFICA DO PRODUTO

Para que VALOR DO NEGÓCIO

O acrononimo INVEST [ref] ajuda a escrever boas histórias. Esta história é... Independente? Negociável, tem Valor para o negócio (Valuable, em ingles)?, Estimável?, Small (em ingles, ou de tamanho relativamente pequeno)? Testável?

Outro acrononimo que auxilia é o 3Cs [ref]. Em ingles, os três Cs são: card, conversation, and confirmation. Histórias de usuários têm três aspectos críticos . O cartão da história deve seguir um modelo simples (como um ... Eu quero .. para que ...). A partir deste modelo simples (card), teremos conversas colaborativas (conversations) para melhor entender e detalhar a história, até que, quando pronta, recebemos a confirmação (confirmation), geralmente pelo dono do produto (do ingles Product Owner, ou PO).

Critério de aceite

Critério de aceite é um formato textual que descreve como testar uma funcionalidade. Tipicamente uma história do usuário terá alguns critérios de aceite.

Dado que CENÁRIO INICIAL

Quando AÇÂO REALIZADA

Então ESTADO ESPERADO

Os critérios de aceite (ACs) definem os limites para uma história de usuário. A partir deles todaos envolvidos irão saber se a história está completa, e devemos buscar a confirmação do PO sobre a mesma.

Tarefas

É muito comum quebrar uma história em pedaços ainda menores sobre o trabalho que deve ser feito. Essas são as tarefas. Ao listar as tarefas necessárias para construir uma história, a equipe de desenvolvimento entra em detalhes técnicos de como os pedaços menores da histórias serão implementados. Diferente das histórias, as tarefas não seguem um formato textual definido. Essas são mais diretas, com uma linguagem bem técnida, da equipe de desenvolvimento para a equipe de desenvolvimento.

Uma tarefa identifica algo que precisa ser feito, geralmente, algo necessário para alguma história. Como tal, a tarefa não será necessariamente independente, ou demostrar o valor de negócio. A maioria das tarefas tendem a ser para programadores, descritas com termos utilizdos pelos mesmos. Alguns exemplos de tarefas: alterar campos da tabela ABC, criar dados de teste para XYZ, automatizar os scripts de BLAH, e assim por diante.

Um exemplo

Segue abaixo um exemplo de feature que foi dividida em três histórias, com alguns critérios de aceite e tarefas.

Funcionalidade: Consultar partidas sem geolocation

História 1

Como um peladeiro não-cadastrado

Eu quero consultar partidas próximas a um endereço informado

Para que eu encontre um jogo próximo ao meu local atual

História 2

Como um peladeiro cadastrado

Eu quero consultar partidas próximo ao meu trabalho

Para que eu encontre um jogo próximo ao meu trabalho

História 3

Como um peladeiro cadastrado

Eu quero consultar partidas próximo a minha residência

Para que eu encontre um jogo próximo a minha residência

Critérios de aceite da história 2 (exemplo 1)

Dado que existe uma partida a menos de 10 kilometros do meu trabalho

Quando procuro por uma partida próxima ao meu trabalho

Então encontro tal partida

Critérios de aceite da história 2 (exemplo 2)

Dado que não existe nenhuma partida a menos de 10 kilometro do meu trabalho

Quando procuro por uma partida próxima ao meu trabalho

Então não encontro nenhuma partida

Tarefas da História 2

- criar UI para mostrar partidas próximas ao local de trabalho (com dados hardcoded)
- criar lógica no backend para busca por proximidade (hardcoded 10 km)
- alterar parametro hardcoded para campo de configuração
- criar dados de teste para buscar partida nas proximidades do trabalho
- alterar DB para incluir endereço de trabalho

O canvas MVP retrata a relação entre MVP e funcionalidades. Mas, tipicamente, as equipes Scrum trabalham com histórias de usuários. Portanto, você deve detalhar um pouco mais, e realizar o mapeamento de funcionalidades para histórias.

As funcionalidades são tipicamente descritas em um nível mais elevado do que as histórias do usuário. Portanto, antes da reunião de planejamento da sprint, o próximo conjunto de funcionalidades deve ter sido analisado e detalhado em suas respectivas histórias de usuário.

Segue abaixo um exemplo de funcionalidade que foi dividida em três histórias.

Funcionalidade: Consultar partidas sem geolocation

Histórias 1: Como um peladeiro não-cadastrado Eu quero consultar partidas próximas a um endereço informado Para que eu encontre um jogo próximo ao meu local atual

História 2: Como um peladeiro cadastrado Eu quero consultar partidas próximo ao meu trabalho Para que eu encontre um jogo próximo ao meu trabalho

História 3: Como um peladeiro cadastrado Eu quero consultar partidas próximo a minha residência Para que eu encontre um jogo próximo a minha residência

Princípios Técnicos

Preparar Ambiente de Desenvolvimento

.

Integração Contínua

.

Entrega Contínua

A entrega contínua[5] está cada vez mais se consagrando como a prática que guiará times maduros para o futuro. Atualmente, a mesma equipe que adotou testes automatizados e configurou um servidor de integração contínua para aumentar a qualidade do produto final busca automatizar desde testes até a implantação do sistema, garantindo o máximo de qualidade e previsibilidade nas tarefas da equipe, principalmente nas mais críticas.

Desde que as práticas de XP ref xp foram compartilhadas, desenvolvedores de software buscam melhorar a qualidade do seu trabalho através de técnicas como testes automatizados e desenvolvimento orientado a testes (TDD). Em um segundo momento, essas equipes caminham para a integração contínua, sendo que cada pedaço de código passa por todos os testes para garantir que problemas sejam detectados o mais rápido possível, fornecendo uma visão centralizada de qualidade e possibilitando que os problemas sejam expostos para toda a equipe.

Uma característica chave da entrega contínua é que ela precisa do envolvimento de toda a equipe. Adotar testes melhora a qualidade do produto final, o que é uma tarefa primordial dos desenvolvedores. Nesse sentido, a entrega contínua procura tirar as dificuldades técnicas do caminho e fazer com que a equipe de negócios (stakeholders, clientes etc.) possa decidir quando entregar. E quem melhor do que ela para tomar tal decisão?

Infraestrutura como Código

O conceito chave aqui é que é possível modelar a infraestrutura como se fosse código, de modo que sua abstração, design, implementação e deploy ocorra da mesma maneira que utilizamos quando

[5] http://martinfowler.com/books/continuousDelivery.html

desenvolvendo aplicações. Assim, trabalha-se com esse código utilizando as mesmas ferramentas e práticas que são utilizadas em projetos modernos de software.

O código que modela, builda e gerencia a infraestrutura é submetido ao sistema de controle de versão juntamente com o código da aplicação, tornado o setup controlado e repetível. A mudança de pensamento é que a infraestrutura deve ser redeployavel a partir do código; um código que deve ser desenvolvido utilizando as metodologias já estabelecidas em anos em que a indústria amadureceu.

Arquitetura Mínima Viável (MVA)

O MVP permite que a equipe de possa entregar um produto nas mãos de usuários rapidamente, à um baixo custo. Quanto mais cedo o produto for utilizado, mais se pode aprender com os usuários para melhorá-lo. A melhor maneira de aprender e evoluir é ouvir de usuários que utilizam o software, não de pessoas olhando para uma folha de papel em branco tentando visualizar como um sistema deve funcionar.

Um dos desafios que a abordagem MVP apresenta é de que ela pode levar à falta de uma arquitetura. Quando construindo um produto do zero, a pressa para entregá-lo pode vir ao custo de uma arquitetura sustentável. Deve haver balanço entre velocidade de entrega e qualidade. É necessária uma arquitetura mínima viável (MVA)[6].

Então, como definimos qual o MVA? Recomendamos realizar perguntas como estas:

- Quantos usuários utilizarão o sistema no lançamento inicial? Nos primeiros 6 meses? Dentro de um ano?
- Os usuários iniciais serão internos, externos, ou ambos?
- Quantas transações por segundo esperamos no lançamento? Nos primeiros 6 meses? Dentro de um ano?
- Como os usuários começarão a utilizar o sistema?
- Qual o nível de segurança e auditabilidade exigido no lançamento? Dentro de 6 meses? Dentro de um ano?

Há diversas outras perguntas para guiar a discussão. Estas perguntas ajudam a equipe a definir os requisitos básicos para lançar no mercado. Embora não entre no mérito de definir uma arquitetura completa e perfeita para o produto final, isto é diferente de ignorar a definição de uma boa arquitetura. O foco é no quanto de investimento é necessário para lançar e, em seguida, definir um plano para evoluir a arquitetura conforme o número de usuários aumenta e requisitos são adicionados.

Tentar construir o sistema perfeito raramente é a abordagem correta. Vamos dizer que a resposta do dono do produto para estas perguntas sejam as seguintes:

[6] http://www.kavistechnology.com/blog/minimal-viable-architecture/

- Haverá apenas cinco usuários internos nos primeiros três meses. Seis meses a partir de agora os nossos dois primeiros clientes externos estarão usando os sistemas em modo piloto. Um ano a partir de agora lançaremos o sistema para todos os clientes.
- No lançamento haverá uma quantidade trivial de transações. Daqui a seis meses o tráfego será moderado. No próximo ano, o tráfego será extremo.
- Inicialmente, adicionaremos usuários ao sistema através de uma interface. No futuro, os clientes terão gestão de ID self-service. Espero que isto aconteça em um ano a partir de agora.
- Nós só precisamos de segurança mínima para a primeira versão. Para o piloto, precisamos de segurança e audit completos.

Com base nessas respostas, muita discussão e definição pode ser adiada para após o lançamento da primeira versão. Por que gastar tempo na estratégia de cache agora, quando não vemos a necessidade no horizonte de um ano? Nós podemos colocar fora diversas tarefas de gerenciamento de ID para mais tarde também. Deve-se evitar discussões imediatas sobre requisitos que só virão no futuro, de forma que seja implementado apenas o estritamente necessário da melhor forma possível. É por isso que o MVA é tão importante.

Esta abordagem depende da disciplina e confiança entre o dono do produto e da equipe de desenvolvimento, para garantir que ambas obtenham os resultados desejados: o produto é entregue rapidamente com um sistema bem arquitetado e com pouca dívida técnica.

Atividade: Definindo Arquitetura

.

Micro-serviços de Domínio

.

Automação

.

Trunk único

.

Compartimentalização

.

Feature Toggles

Com a utilização de Trunk Based Development (TBD) surge a dúvida de como fazer para que funcionalidades em desenvolvimento não impeçam que o Trunk esteja sempre pronto para ir para produção. A solução mais utilizadas com este objetivo é a utilização de feature toggles.

As feature toggles funcionam como um interruptor que permite manter uma funcionalidade desligada em produção até que todo o desenvolvimento da funcionalidade esteja concluído.

A maneira mais simples de se implementar uma feature toggle é colocar uma propriedade no arquivo de configuração da aplicação descrevendo o estado da toggle, ativado ou desativado, e colocar um condicional no código para verificar se a funcionalidade está ativa antes de executar esta.

imagem com exemplo do arquivo de configuração e do código

Existem outras maneiras mais complexas de implementar toggles para permitir mais flexibilidade na maneira de ativar/desativar estas. Por exemplo, é possível criar mecanismos para que somente parte dos usuários tenha a funcionalidade ativada, permitindo fazer A/B ou canary release.

Um grande benefício da utilização de toggles é que permite separar o processo de deploy de uma funcionalidade e o processo de release desta, ou seja, o código da funcionalidade pode estar pronto em produção, mas invisível aos usuários. Isto é muito útil quando você tem que desenvolver uma funcionalidade que tem uma data específica para ser ativada, por exemplo, como funcionalidades relacionadas a novas legislações, ou quando você quer submeter uma aplicação mobile ao processo de aprovação da apple store ou google play, mas quando o back-end ainda está em desenvolvimento.

Como qualquer técnica, toggles também adicionam complexidade adicional no seu processo de desenvolvimento. Por isso é muito importante remover o código referente a uma toggle assim que este não for mais necessário e tentar manter o menor número possível de toggles no código.

Tipos de feature toggles:

Toggles de desenvolvimento: tem vida curta e são utilizadas para esconder o código em desenvolvimento;

Toggles de operações: são utilizadas para permitir que algumas funcionalidades não críticas ou ainda experimentais sejam desligadas em produção em momentos críticos. Por exemplo, desligar a funcionalidade de recomendações em dias de grandes promoções, para diminuir o tempo de carregamento da página em um site de e-commerce;

Toggles de negócios: são utilizadas pelo negócio para fazer experimentações como, por exemplo, A/B testing.

MVP em Produção

.

Deploy

.

Decouple Deploy e Release

.

Acompanhamento

.

Monitoração

.

Hypex

.

Próxima Fase: MVP 2 em Construção

.

Kanban

.

Conclusão

.