

Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Caio Teixeira da Quinta
Eugênio Augusto Jimenes

**Plataforma Web para divulgação e centralização de
eventos aplicando conceitos de Métodos Ágeis
e Lean Startup**

São Paulo
Dezembro de 2016

Plataforma Web para divulgação e centralização de eventos aplicando conceitos de Métodos Ágeis e Lean Startup

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Dr. Alfredo Goldman vel Lejbman
Cosupervisor: Jorge Melegati

São Paulo
Dezembro de 2016

Agradecimentos

Resumo

<em desenvolvimento>

É notável que a Cidade Universitária possui uma imensa diversidade acadêmica e cultural que se manifesta em uma variedade de eventos sendo promovidos e organizados pela Universidade de São Paulo além de iniciativas da própria comunidade para ocupar o espaço público.

Levando em consideração a quantidade de eventos acontecendo de forma simultânea, a extensão física do campus, a pulverização dos eventos por toda sua extensão e a quantidade de informações dispersas entre as várias redes de comunicação muitas vezes não tomamos conhecimento a tempo de alguma oportunidade que poderia ser interessante.

Através de uma votação junto à comunidade USP foi decidido criar o USP Eventos, uma plataforma web colaborativa na qual qualquer membro frequentador do campus pode se informar do que acontece no mesmo além de organizar e divulgar seu próprio evento.

Tomando como base uma metodologia de projeto baseada nos conceitos de Métodos Ágeis e Lean startup foi proposto desenvolver um sistema em Ruby on Rails utilizando ciclos de Build Measure Learn sempre pautando o desenvolvimento segundo o aprendizado obtido através dos comentários e críticas dos usuários.

Palavras-chave: eventos, métodos ágeis, lean startup, desenvolvimento web.

Abstract

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa.

Keywords: keyword1, keyword2, keyword3.

Sumário

Lista de Abreviaturas

MVP	Produto Mínimo Viável (<i>Minimum Viable Product</i>)
CoC	Convenção sobre Configuração (<i>Convention over Configuration</i>)
DRY	Não se repita (<i>Don't Repeat yourself</i>)
ORM	Mapeamento Objeto Relacional (<i>Object Relational Mapping</i>)
POP	Painel de Opinião Popular
UX	Experiência do Usuário (<i>User Experience</i>)
UI	Interface de Usuário (<i>User Interface</i>)
RoR	Ruby on Rails
Pass	Plataforma como serviço (<i>Platform-as-a-Service</i>)

Capítulo 1

Introdução

1.1 Motivação e Objetivos

Devido a sua grande diversidade cultural a Cidade Universitária possui uma grande abundância de eventos sociais acadêmicos ocorrendo em seu dia-a-dia sendo organizados pela própria Universidade ou pelos próprios usuários do campus.

Ao realizar uma enquete junto à comunidade USP foi percebido o interesse em uma plataforma para centralizar os eventos na USP pois devido a extensão física do campus e diversidade de eventos propagação de informações muitas vezes dispersava-se levando os usuários a perder oportunidades interessantes por não tomar conhecimento à tempo.

Em vista desse problema foi proposto criar um Sistema Web colaborativo para centralizar e divulgar os eventos que são organizados pela Universidade de São Paulo e comunidade para isso adotamos uma metodologia de projeto seguindo os preceitos de Lean Startup e Métodos Ágeis.

A escolha da metodologia Lean Startup deu-se pela sua ampla aplicação em projetos com grande grau de incerteza no qual uma abordagem interativa e um desenvolvimento incremental auxiliam no aprendizado e direcionam o desenvolvimento. Durante o projeto descrevemos como tais conceitos auxiliaram no desenvolvimento do sistema e as vantagens dessa abordagem.

1.2 Capítulos

Capítulo 2

Lean Startup

2.1 Lean Startup: O que é

2.1.1 Startup: uma definição

Através da popularização da internet e dos computadores pessoais nos anos 90, nos Estados Unidos, o termo startup foi generalizado para classificar pequenas empresas com propostas inovadoras, sejam por atuarem com as novas tecnologias que surgiram para o grande mercado na época como as chamadas empresas online ou empresas "ponto com" ou pelo seu novo modo de organização e processo de produção.

o ? provê uma definição de uma startup de software a partir do ? que baseia-se nos desafios que ela enfrenta:

- Pouco ou nenhum histórico operacional - Startups com pouca ou nenhuma experiência em desenvolver processos de negócio e gerenciamento organizacional.
- Recursos limitados - Startups tipicamente focam em lançar um produto, promovê-lo e construir alianças estratégicas.
- Múltiplas Influências - Pressão dos investidores, clientes, parceiros e competidores impactam nas tomadas de decisões de uma empresa. Apesar de importantes em média elas tendem a ser inconsistentes.
- Mercado e tecnologias dinâmicas- Empresas de softwares novos frequentemente precisam desenvolver ou operar com tecnologias disruptivas para atuar em potenciais mercados alvos.

Apesar de não ser um rótulo exclusivo para mercado de Tecnologia da Informação alguns locais e atividades foram particularmente associados à classificação devido a revolução tecnológica promovida pela bolha da internet, como no Vale do Silício, área norte do estado da Califórnia, EUA, conhecida até hoje por ser um ecossistema constituído de empresas inovadoras.

Com o passar dos anos e com o impacto da internet no mercado global o termo amadureceu para empresa, grupo ou organização que busca um modelo de negócios escalável geralmente envolvida em implementações de processos inovadores de desenvolvimento e pesquisa de mercado-alvo. Grosseiramente uma startup surge sob uma ideia ou foco que pode ou não dar certo mas consciente desta instabilidade.

Para Steve Blank, acadêmico e empreendedor do Vale do Silício, uma startup vai de fracasso à fracasso com objetivo de aprender com cada falha e assim definir o que não

funciona no processo na qual a empresa esta engajada. Por isso é considerado um modelo de negócio escalável: deve ser flexível perante a constante reação do mercado e da própria produção.

2.1.2 Lean Startup

Lean Startup (Startup Enxuta) é um conceito introduzido em 2008 por Eric Ries, empreendedor de diversas startups do Vale do Silício. Trata-se de uma metodologia de negócios derivada da combinação de outros padrões de desenvolvimento como *Minimal Viable Product* (produto viável mínimo), *Customer Development* (desenvolvimento voltado ao cliente) e *Agile software development* (desenvolvimento ágil de software ou Método Ágil).

Ries propõe que é possível encurtar os ciclos de implementação de um produto (ou solução) adotando uma combinação de testes, hipóteses de negócio e degustações por parte do público-alvo. Através do lançamento periódico de cada versão do produto é possível avaliar não apenas quesitos técnicos como também a reação do mercado. Consequentemente o retorno de cada iteração afeta o planejamento do produto e suas futuras versões. Essa economia em cada ciclo provém do *validated learning* (aprendizagem de validação), ou seja, toda ideia, funcionalidade ou vertente de produto deve ser primeiramente experimentada e testada. Isso evita desperdício de desenvolvimento, abre oportunidades para adaptações e alterações de projeto em casos de falha ou rejeição do cliente.

Versões simples do produto sob cada ciclo de avaliação é uma estratégia derivada do padrão *Minimal Viable Product*, proposta em 1996 por Frank Robinson, CEO da empresa SyncDev, porém popularizado anos depois por Steve Blank.

Robinson propõe o lançamento de uma versão o mais simples possível de modo à antecipar a análise de mercado e assim minimizar o risco de retorno por parte da empresa. A popularidade Steve Blank foi em adaptar a estratégia incluindo o lado do cliente na balança, o que ele chama de Customer Development. Blank vai além de apenas minimizar o risco de retorno: busca compreender as necessidades do cliente. Lean

Startup aprimora ainda mais o conceito para avaliações sob cada interação e assim maximizar o aprendizado e evolução do projeto, alinhado com o desejo do mercado, o chamado ciclo *Build-Measure-Learn* (Construir-medir-aprender).

2.2 Produto Mínimo Viável (MVP)

Uma definição de Produto Mínimo Viável por Eric Ries: "*A Minimum Viable Product is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort.*" ?.

A idéia central do conceito de MVP é maximizar a validação de aprendizado sobre um produto utilizando o menor esforço possível. Como na maioria das startups os recursos financeiros e humanos são bastante escassos o tempo de validação do produto e determinação do interesse do público é um fator decisivo para o sucesso da mesma.

Um MVP deve possuir 3 características ¹:

1. Ter valor suficiente para que uma pessoa queira utilizá-lo ou comprá-lo.
2. Possui suficientes benefícios para reter os chamados usuários pioneiros.
3. Ser capaz de prover um ciclo de feedback suficiente para guiar o desenvolvimento.

¹ Retirado de Techopedia: Minimum Viable Product (MVP): <https://www.techopedia.com/definition/27809/minimum-viable-product-mvp>

Durante a concepção do projeto são definidas algumas hipóteses sobre o produto e na etapa do MVP é definido então qual será o seu núcleo, ou seja, quais funcionalidades ou estratégias queremos testar de modo que possamos validar as nossas hipóteses iniciais e obter o máximo de aprendizado possível utilizando-o para planejar novas funcionalidades e determinar as prioridades para a equipe de desenvolvimento.

Além da validação de aprendizado outra vantagem significativa provida pelo MVP é evitar desperdícios. O MVP permite testar se a funcionalidade ou hipótese sobre um projeto é bem aceita pelo público alvo implementando-a de uma forma simplificada sem dispendar horas a fio de desenvolvimento. Dessa forma caso comprove-se que tal premissa não é interessante para o projeto seu desenvolvimento é interrompido sem que tenham sido desperdiçados tempo e recursos.

Os termos "mínimo" e "máximo" referentes a "máximo aprendizado" e "mínimo produto viável" frequentemente se mostram vagos na documentação do que é um MVP, citando Eric Ries: "[...]the definition's use of the words *maximum* and *minimum* means it is decidedly not formulaic. It requires judgment to figure out, for any given context, what MVP makes sense."

² É importante ressaltar que um MVP não é um produto completo com as funcionalidades mínimas e sim um conjunto de características mínimas que configuram o serviço ou produto que está sendo oferecido. Dessa forma um MVP pode ser apenas um protótipo, um produto completo ou mesmo apenas um *mock-up* do que será oferecido na versão completa.

Alguns tipos de MVP são:

³

- *Vídeo Explicativo*: Um vídeo curto contendo uma explicação clara do que o seu produto faz e porque as pessoas deveriam utilizá-lo. Esse é o caso do dropbox que fez um vídeo ⁴ com cerca de 5 minutos explicando o que era o seu serviço.
- *Landing Page*: Criar uma página inicial contendo uma explicação detalhada do que é o produto que você irá oferecer juntamente com um formulário de contato. Através de uma configuração simples pelo Google Analytics é possível manter um registro de conversões (no caso cadastros do formulário) a fim de medir o interesse das pessoas no seu produto.
- *MVP "Mago de OZ"*: A idéia é criar uma página visualmente completa que funcione como o produto final mas que na verdade exista alguém executando as tarefas manualmente. Esse foi o caso da Zappos hoje a maior vendedora de sapatos dos Estados Unidos.
- *MVP com Consierge*: Ao invés de prover um produto você realiza manualmente o serviço executando exatamente os mesmos passos para o usuário que o a sua empresa realizaria. É um método não escalável e lento para executar pois requer que você esteja em contato direto com o cliente e realize as tarefas manualmente porém isso permite um rápido aprendizado sobre o produto e o cliente.
A empresa Food on the Table ajuda seus consumidores a criarem listas de compras, acharem receitas e conseguirem descontos nos ingredientes em seus supermercados favoritos, inicialmente seus fundadores encontraram uma senhora interessada no serviço e por 10 dólares/semana eles mantinham as listas de compra e procurava, por descontos nos supermercados em que ela fazia compras.

²Fonte: Startup Lessons Learned, Minimum Viable Product: a guide <http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html>

³ Scale my Business: The Ultimate Guide to Minimum Viable Products <http://scalemybusiness.com/the-ultimate-guide-to-minimum-viable-products/>

⁴Link para o vídeo <https://www.youtube.com/watch?v=7QmCUDHpNzE>

2.3 O ciclo de Build-Measure-Learn

Um ciclo de build measure learn é uma abordagem de desenvolvimento do produto que aprimora o tradicional modelo de desenvolvimento "Cascata" utilizado de forma abrangente durante o século XX.

2.3.1 O modelo Cascata

O nome "Cascata" é bastante literal e tem origem na própria estrutura do método (figura ??) que seguia uma série de passos de forma sequencial de maneira bastante direta, como se fosse uma cascata.

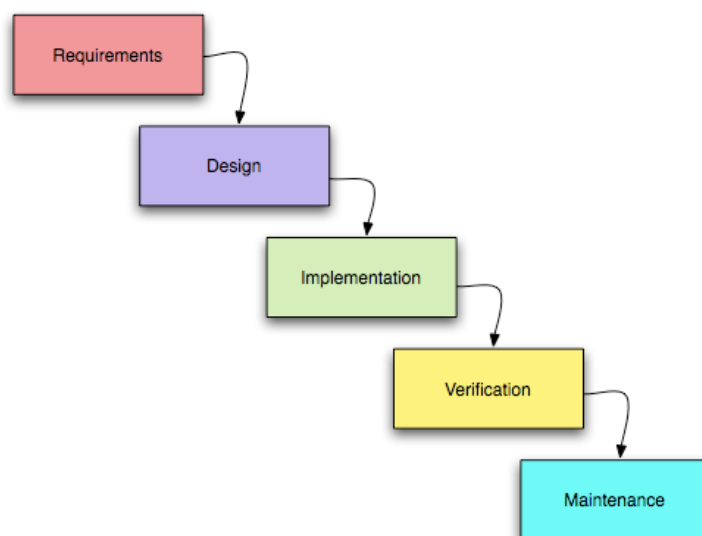


Figura 2.1: *O Modelo Cascata.*

O método consiste em um desenvolvimento sequencial e não-iterativo iniciando com a parte de análise de requisitos e de forma subsequente temos as etapas de design de projeto, implementação, verificação e manutenção.

Uma rápida explicação dos passos:

- *Requerimentos:* Realizar a análise de requisitos do projeto.
- *Design de Projeto:* Focando na estrutura de dados, arquitetura do software, detalhes procedais e caracterização das interfaces é formulado um documento de forma a apresentar os requerimentos de uma forma que possa ser interpretado pelos programadores.
- *Implementação:* Etapa da codificação do projeto propriamente dita.
- *Verificação:* Etapa para teste do produtos visando eliminar qualquer *bug* que possa ter passado despercebido e refinar a lógica interna do software caso necessário.
- *Manutenção:* Etapa para instalação do sistema no cliente, configuração de servidores, etc.

Uma das grandes das críticas dessa abordagem é que dificilmente um desenvolvimento de software segue todas as etapas da forma como o modelo propõe e nem sempre o cliente sabe definir bem os requisitos antes de ver o software funcionando resultando em tempo

e desenvolvimento desperdiçado em funcionalidades que não resolvem o problema além de mudanças tardias no escopo do projeto que encarecem o custo total e poderiam ter sido evitadas e contornadas de maneira mais satisfatória em um modelo com um processo de desenvolvimento iterativo ⁵.

Ao contrário do modelo cascata no qual o contato do cliente com o software se dá apenas no fim do processo de desenvolvimento na etapa de testes o método de *Build-Measure-Learn* privilegia desde o início uma interação contínua com o cliente.

2.3.2 Build-Measure-Learn (Construir-medir-aprender)

Na fase de Verificação do modelo Cascata existia a possibilidade de disponibilizar para os clientes versões alphas ou betas do software em questão, como o foco não era obter um retorno sobre o desenvolvimento e sim verificar a existência de *bugs* o usuário acabava completamente fora do processo de desenvolvimento.

Com o surgimento do *Agile Development* foi possível criar softwares de maneira iterativa e envolver o cliente no processo de porém devido a falta de um arcabouço para testar às hipóteses comerciais acabava-se muitas vezes por desenvolver um software com todas as funcionalidades que o cliente gostaria e ainda sim não obter um sucesso comercial.⁶

O modelo de Construir-Medir-Aprender surge então com o principal objetivo de eliminar as incertezas sobre as hipóteses do produto. Através de um aprendizado rápido sobre o comportamento dos usuários é possível minimizar os riscos e custos desnecessários que persistir em uma idéia equivocada pode causar, mantendo o aspecto iterativo presente na metodologia Agile e obtendo um aprendizado sobre o comportamento do usuário em cada iteração.

O ciclo de *Build-Measure-Learn* é uma das idéias fundamentais do *Lean Startup* consistindo de um ciclo de 3 fases (Construir, Medir e Aprender) porém para melhor ilustrar o que representa cada uma das fases foi incluído na figura ?? outras 3 menores (*ideas*, *data* e *code*).

- *Build*: Inicialmente temos algumas idéias que foram definidas a partir das hipóteses do produto que precisam ser implementadas (*code*) no MVP.
- *Measure*: Implementado o MVP coletamos dados(*data*) de uso e seguindo algumas métricas já definidas avaliamos seu desempenho. Todo o ciclo é baseado na idéia de aprendizado rápido para aprender o máximo possível sobre como os usuários reagem as idéias implementadas.
- *Learn*: A partir então da análise dos dados coletados podemos inferir sobre continuar sobre como continuar o desenvolvimento e o que funcionou ou não. Feito isso implementamos as novas idéias geradas e começamos novamente o ciclo para validá-las.

As etapas do ciclo não precisam necessariamente ocorrer em ordem podendo se sobrepor ou mesmo serem unidas dependendo de como for o ciclo de desenvolvimento. (referencia eric ries)

As alterações de software precisam ser feitas de maneira rápida de modo a testar o mais rápido possível novas idéias portanto é importante ressaltar que as funcionalidades devem se manter simples e diretas pois o foco é o aprendizado gerado e não desenvolver um software

⁵ Fonte: Wikipedia - Waterfall Model https://en.wikipedia.org/wiki/Waterfall_model

⁶Fonyr: Por Steve Blank em <http://venturebeat.com/2015/05/06/build-measure-learn-doesnt-mean-throwing-things-against-the-wall-to-see-if-they-stick/>

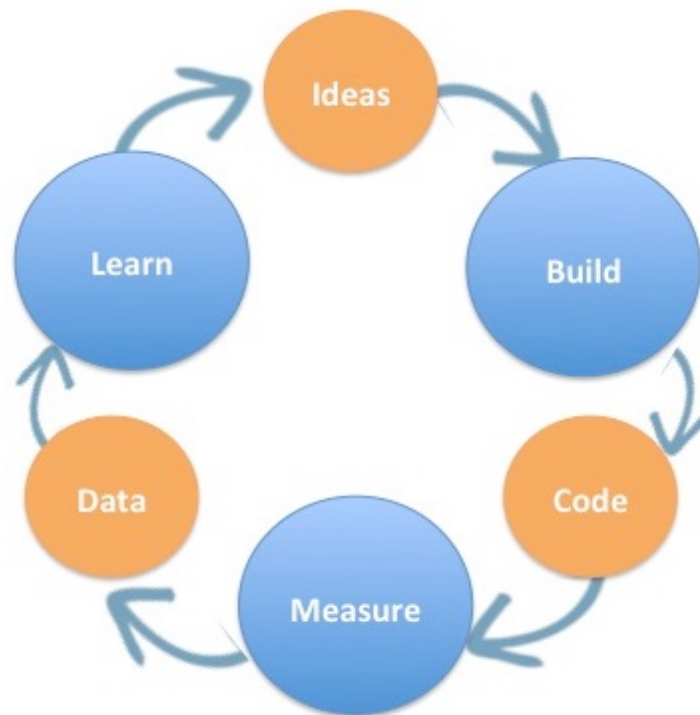


Figura 2.2: O ciclo de *Build Measure Learn*.

ou um protótipo completo pois dessa maneira é possível evitar que sejam desperdiçados recursos em uma funcionalidade que pode não ser bem recebida e não ter continuidade.

Apesar do desenvolvimento ser focado na simplicidade isso não quer dizer produzir um produto de baixa qualidade pois um produto de baixa qualidade ou que possua bugs pode resultar em uma taxa de rejeição inesperada devido a problemas de implementação fazendo com que as conclusões do ciclo possam ser enviesadas.

Para minimizar que um sistema com defeito seja colocado em produção procura-se utilizar ferramentas que auxiliam na integração contínua do software além de construir-se testes automatizados garantindo que seja possível manter um desenvolvimento consistente e confiável sem comprometer o tempo de execução.

O que o Construir-Medir-Aprender perde de vista é que novos empreendimentos, tanto startups quanto novas ideias dentro de empresas já existentes não começam com ideias mas com hipóteses. "Ideia" evoca uma visão que imediatamente requer um plano para se frutificar. Em contraste, "hipótese" indica um palpite com precedentes que requer experimentação e dados para ser validado ou invalidado.⁷

Como o que você constrói deve estar alinhado com o as hipóteses formuladas e a cada ciclo é necessário sempre testar novas hipóteses resultando em diferentes protótipos a figura ?? representa uma variação do ciclo de construir-medir-aprender cuja proposta é enfatizar quais hipóteses deveriam ser testadas.

Adquirindo os benefícios herdados pela metodologia Agile e somando a isso um arcabouço para adquirir aprendizado sobre a utilização do software o ciclo de *Build-Measure-Learn* resulta em um software mais refinado e alinhado com as expectativas dos clientes em contraponto aos problemas apresentados pelo modelo em Cascata.

⁷Endeavor Brasil <https://endeavor.org.br/construir-medir-aprender/>

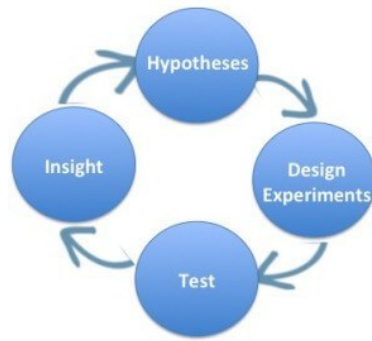


Figura 2.3: *Uma variação para o ciclo de Construir Medir Aprender.*

2.4 Desenvolvimento de Clientes

TODO

Capítulo 3

Métodos Ágeis

3.1 Origem

Na década de 90 com a decadência dos métodos clássicos de desenvolvimento em software, como o modelo cascata, o qual propõe um rígido padrão de etapas sequenciais com maior ênfase no planejamento, considerado lento e burocrático com pouco espaço para alterações ou readaptações, novos modelos surgiram com o objetivo de maior interatividade entre o cliente e produção, flexibilizar as etapas de planejamento e validação do sistema. Dos mais conhecidos e citados até hoje, temos: o desenvolvimento rápido de aplicações de 1994, processo unificado e sistemas dinâmicos de método de desenvolvimento (DSDM) de 1995, Scrum de 1996, *Crystal Clear* e programação extrema (XP) ambos de 1997. Embora originados antes da publicação do Manifesto Ágil em 2001, são hoje colectivamente referidos como métodos ágeis pois influenciaram diretamente na conceituação do mesmo.

Em fevereiro de 2001, 17 desenvolvedores de software se reuniram no resort Snowbird em Utah, EUA, para discutir métodos leves de desenvolvimento. Publicaram o Manifesto para Desenvolvimento Ágil de Software, documento que reúne os princípios e práticas desta metodologia. Mais tarde, algumas pessoas formaram a Agile Alliance, uma organização sem fins lucrativos que promove o desenvolvimento ágil.

3.2 Definição

O Manifesto Ágil é baseado em doze princípios¹:

1. A satisfação do cliente é pela entrega antecipada e contínua do software funcional;
2. Mudança, mesmo em desenvolvimento tardio, devem ser entregues frequentemente (semanas em vez de meses);
3. Uma cooperação estreita e diária entre cliente e desenvolvedor;
4. Os projetos são construídos em torno de indivíduos motivados, entre os quais existe relação de confiança;
5. A maneira mais eficiente e efetiva de transmitir informações é por conversas cara-a-cara;
6. Softwares funcionais são a principal medida de progresso;

¹ Retirado de: <http://agilemanifesto.org/principles.html>

7. Desenvolvimento sustentável, ou seja, produção em um ritmo constante;
8. Atenção contínua à excelência técnica e bom design;
9. Simplicidade em maximizar a quantidade de trabalho não feito;
10. Melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas;
11. Regularmente, a equipe se reuni e reflete sobre como tornar-se mais eficaz, então se ajusta para o que foi acordado.

Comparado a engenharia de software tradicional, desenvolvimento ágil de software tem como alvo principalmente sistemas complexos com características dinâmicas, ou seja, não determinístico e não-linear, onde as estimativas precisas, planos estáveis, e as previsões são muitas difíceis de se obter em estágios iniciais como na própria elaboração do projeto em si do produto, necessitando de um desenvolvimento e planejamento evolutivo.

Existem muitos modelos específicos de métodos ágeis, a maioria busca promover o trabalho em equipe, colaboração e adaptabilidade no processo em todo o ciclo de vida de desenvolvimento. Utiliza-se uma maior quantidade de ciclos em curtos períodos de tempo à fim de minimizar e parcelar o planejamento e estimativas. Cada iteração envolve uma equipe multifuncional de trabalho para cada função necessária: planejamento, análise, projeto, codagem, testes unitários e testes de aceitação. No final da iteração é demonstrado para as partes interessadas (cliente) o que foi desenvolvido no processo. Isto minimiza o risco global de erros e permite que o produto se adapte às mudanças rapidamente. Uma iteração pode não adicionar a funcionalidade suficiente para justificar um lançamento no mercado, mas o objetivo é ter uma versão disponível (com erros mínimos) no final de cada iteração, porém várias iterações pode ser necessária para liberar uma versão de produto ou novos recursos.

Independente do método ágil seguido, cada equipe deve incluir um representante do cliente, por exemplo o "proprietário do produto" no caso do Scrum. Esta pessoa faz um compromisso pessoal de estar disponível para os desenvolvedores em responder questões durante cada iteração. Isto otimiza o retorno de investimento e garante o alinhamento com as necessidades do cliente.

Outra característica comum deste tipo de metodologia é o *stand-up* diário. Em uma breve sessão, os membros da equipe comunicam uns aos outros o que eles fizeram no dia anterior sobre as tarefas relacionadas o ciclo vigente, o que eles pretendem fazer, e quaisquer obstáculos ou impedimentos para continuar ou finalizar a tarefa do dia para a interação em questão.

Diferentes ferramentas e técnicas específicas, tais como a integração contínua, testes de unidade automatizado, programação em pares, o desenvolvimento orientado a testes, padrões de projeto, *Domain-Driven Design*, refatoração de código e outras técnicas são frequentemente utilizadas para melhorar a qualidade e aumentar a agilidade de produção.

3.3 Métodos Ágeis específicos

3.3.1 Scrum

Scrum é um framework mais popular hoje de desenvolvimento ágil de software iterativo e incremental. Um princípio chave do Scrum é o seu reconhecimento de que, durante o desenvolvimento do produto, os clientes podem mudar de opinião sobre o que eles querem e precisam, muitas vezes chamado de volatilidade de requisitos, e que os desafios imprevistos não podem ser facilmente tratadas de forma preventiva ou planejado tradicionalmente.

O framework propõe inicialmente três papéis principais para divisão e organização da equipe de trabalho:

- **Product Owner (Dono do Produto):**

Representa as partes interessadas do produto, ou seja, e da voz do cliente, é responsável por garantir que a equipe agregue valor ao negócio, passa a maior parte do seu tempo em comunicação com as partes interessadas e não deve ditar como a equipe chega a uma solução técnica. Este papel é equivalente a "representante do cliente", papel em alguns outros frameworks ágeis, como Programação Extrema (XP).

- **Scrum Master (Mestre Scrum):**

Responsável direto pela solução de possíveis impedimentos da equipe para entregar as metas de produtos e resultados, seja impedimentos estruturais ou pessoais da equipe. O Scrum Master não é um líder de equipe ou gerente de projeto tradicional, mas age como um mediador entre a equipe e quaisquer influências que a distraem. Garante que o framework Scrum é seguido, ajuda a equipe a seguir os processos acordados, e incentiva a equipe a melhorar. O papel também tem sido referido como um "facilitador da equipe" ou "servo-líder" para reforçar estas duas perspectivas.

- **Development Team (Equipe de Desenvolvimento):**

A Equipe de Desenvolvimento é responsável pela entrega de incrementos do produto no final de cada Sprint (ciclo de interação). A equipe é composta de indivíduos que fazem o trabalho real (analisar, projetar, desenvolver, testar, documentar, etc.). As equipes de desenvolvimento são multifuncionais, com todas as habilidades necessárias para criar um produto.

O ciclo de interação chamado *sprint* é a unidade básica do desenvolvimento em Scrum, é restrito a uma duração específica fixado antecipadamente e é normalmente entre uma semana e um mês, com duas semanas sendo o mais comum.

Cada *sprint* começa com um evento de planejamento (*Sprint Planning*), que visa definir a lista de objetivos à serem tratados ou implementados nesse ciclo (*Sprint Backlog*). Cada *sprint* termina com uma revisão (*Sprint Review*), que analisa os progressos realizados para mostrar aos interessados, além disso é debatido as lições e melhorias à serem aplicadas para as próximas interações (*Sprint Retrospective*).

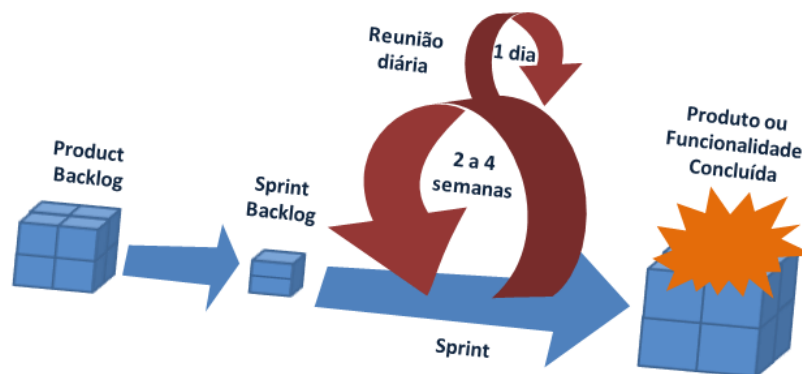


Figura 3.1: O ciclo iterativo Sprint.

3.3.2 Test Driven Development (TDD)

TDD é um processo de desenvolvimento de software que se baseia na repetição de um ciclo de desenvolvimento muito curto: requisitos são transformados em casos de teste específico, em seguida, o software é aprimorado somente para passar no teste específico, mas todos os teste são refeitos a cada interação. Este processo também é chamado de ciclo *Red-Green-Refactor*.

Cada novo recurso a ser implementado começa com a escrita de um teste que define uma função ou melhorias de uma função, o mais sucinto possível. A nova funcionalidade agora passar ser implementada baseado em passar neste novo teste. Esta é principal característica do TDD, que em contrapartida aos demais métodos em que os testes são criados após a implementação da nova função em si. Ao passar no novo teste todo código é então também submetido aos testes anteriores, se passar em todos casos, o programador possui uma maior garantia da integralidade do novo código, ou seja atende aos requisitos de teste e não quebra ou degrada quaisquer outros recursos existentes. Se não passar no testes anteriores, o novo código então deve ser ajustado até que passe, etapa chamada de refatoração.

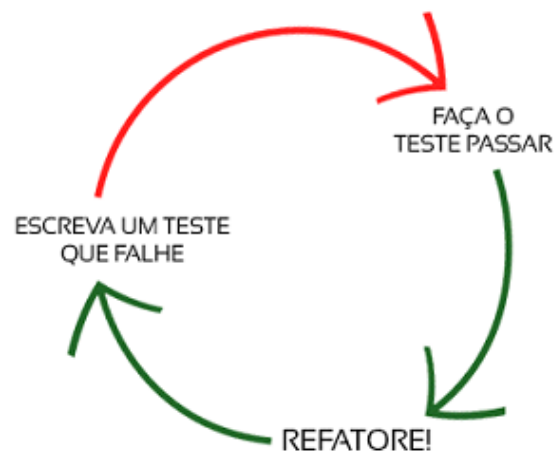


Figura 3.2: O ciclo Red-Green-Refactor.

3.3.3 Continuous Integration (CI)

Integração contínua é um dos pilares da agilidade, garantindo que todo o sistema funcione a cada interação de forma coesa, mesmo que sua equipe seja grande e diversas partes do código estejam sendo alteradas ao mesmo tempo, isso traz um *feedback* diário do desenvolvimento. Essa integração é alinhada diretamente com o conceito de TDD em que cada ciclo é aplicado um conjunto de testes e verificações de integração entre as partes produzidas, além dos testes técnicos da aplicação.

A parte crucial desse processo dentro do conceito de CI é o uso de alguma sistema de controle de versão, estabelecendo como compartilhar informações de maneira sucinta e objetiva, mantendo a última versão do produto válida e ainda saber quem ou qual parte de equipe fez cada alteração, prevenindo desperdício de desenvolvimento, seja em por duplicidade ou refatoração direta. Existe um conjunto de ferramentas para controle de versão centralizado, entre elas temos o *CVS*, *Subversion*, *Git*, entre outros.

O controle de versão, gerência não apenas o código do produto, mas como também a documentação, *scripts* de teste, arquivos de layout e configuração, entre outros. Além disso, e mais importante, é possível criar linhas alternativas de desenvolvimento do produto, chamado de *branches*. O sistema funciona basicamente da seguinte forma: o desenvolvedor faz seu código, efetua um *build* antes de interagir com a base principal do que já foi feito (e testado). Os *builds* são posteriormente integrados a base através de sincronização, o que é feito sob testes e padrões de produção. Essa prática permite uma divisão coesa das tarefas a serem desenvolvidas, eliminando a necessidade de que toda equipe saiba exatamente como cada parte do produto foi ou será feita, trazendo uma visão de linha de montagem: cada desenvolvedor deve saber como fazer especificamente a sua parte, porém consistente de quem deve recorrer caso tenha problemas com outras partes do produto.

3.3.4 Kanban

Kanban é um termo de origem japonesa, significa literalmente “cartão” ou “sinalização”. O conceito foi relacionado com a utilização de cartões para indicar o andamento dos fluxos de produção em empresas de fabricação em série, hoje popularmente utilizado em métodos ágeis. Os cartões são organizados sobre uma determinada etapa do processo de implementação, por exemplo, “para executar”, “em andamento” ou “finalizado”.

A empresa japonesa de automóveis *Toyota* foi a responsável pela introdução desse método devido a necessidade de manter um eficaz funcionamento do sistema de produção em série.

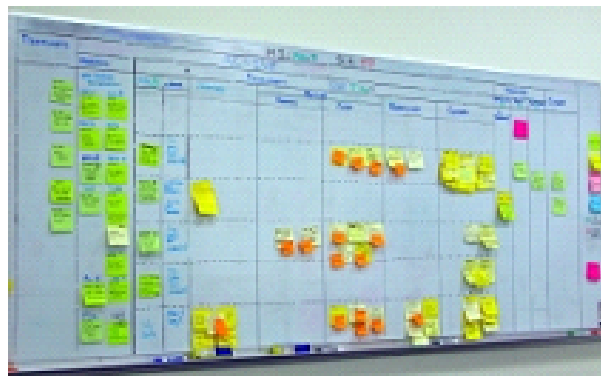


Figura 3.3: Exemplo de um quadro de Kanban.

Capítulo 4

Tecnologias

4.1 Ruby on Rails

4.1.1 Ruby

A criação da linguagem Ruby data de 1995 no Japão por Yukihiro "Matz"Matsumoto sob forte influência de outras linguagens como Perl, SmallTalk, Eiffel, Ada e Lisp e tinha como objetivo equilibrar programação funcional, imperativa e orientação a objetos, segundo o próprio Matz: "Eu queria uma linguagem interpretada que fosse mais poderosa do que Perl e mais orientada à objetos do que Python2.". (inserir referencia)

- Flexibilidade:

Ruby cresceu devido a sua grande flexibilidade sendo possível alterar, remover ou acrescentar partes da linguagem à vontade, imagine o seguinte exemplo: um usuário prefere utilizar a palavra plus ao invés do operador matemático(+), ele poderia então adicionar esse método à classe nativa do Ruby Numeric pois em ruby os operadores matemáticos são considerados açúcares sintáticos.

Exemplo:

```
1 class Numeric
2     def plus(x)
3         self.+(x)
4     end
5 end
6
7 y = 5.plus 6
8 # y agora eh igual a 11
```

- Closures:

Em ruby o closures são chamadas de blocos e são funções que podem ser tratadas como uma variável isso quer dizer que podem ser passadas como argumentos de métodos, serem atribuídas a outras variáveis, etc.

As *closures* armazenam os valores das variáveis que estavam no escopo quando a função foi definida e são capazes de acessar tais variáveis mesmo que sejam executadas em um escopo diferente. ¹

Exemplo:

¹Fonte: Site Point <https://www.sitepoint.com/closures-ruby/>

```

1 search_engines =
2   %w[ Google Yahoo MSN ].map do | engine |
3     "http://www." + engine.downcase + ".com"
4   end

```

- Módulos:

Diferente de outras linguagens orientadas a objetos o Ruby suporta apenas herança simples porém de forma proposital. Em ruby existe o conceito de módulos que são coleções de métodos que podem ser adicionadas à uma classe por meio de um mixin. As classes podem fazer o mixin de um método e receber todos os métodos dele diretamente. Exemplo:

```

1 class MyArray
2     include Enumerable
3 end

```

4.1.2 Rails

David Heinemeier Hansson extraiu o Ruby on Rails do seu próprio trabalho realizado na empresa Basecamp e foi lançado de maneira open-source pela primeira vez em 2004, inicialmente ele estava desenvolvendo seu projeto utilizando PHP porém apesar de conseguir desenvolver de maneira bastante ágil a repetição de código incomodava-o.

Rails é um framework web escrito em Ruby implementado seguindo o padrão MVC totalmente *server-side* sendo considerado portanto um *framework back-end*.

Rails oferece uma estrutura para banco de dados, *web service* e *web pages* além de encorajar padrões de engenharia de software já consagrados tais como:

- *Convention over Configuration* (CoC):

Apesar de algumas críticas por adicionar obscuridade pois não existe uma forma clara de verificar o comportamento esperado ruby adota alguns padrões de configuração visando padronizar o código e tirar do desenvolvedor a decisão de como usar o arcabouço sem que isso tire sua flexibilidade. Por exemplo se existir um objeto chamado User então sua tabela por convenção se chamará users e correspondente *controller* será UsersController (no plural) pois esse é padrão definido para o framework e de outra forma o seu controller ou database não estaria associado ao modelo User, a menos que você mudasse as configurações para não utilizar as convenções padrão.

- *Don't Repeat yourself* (DRY):

Visando diminuir a repetição de informações de qualquer tipo em rails frequentemente ao utilizar o comportamento padrão da linguagem o desenvolvedor terá a impressão de que tal comportamento não tem sem explicação já que sua implementação está escondida dentro do código do próprio Rails porém evitando que dessa forma o programador tenha que reescrever várias vezes um mesmo código para determinado comportamento.

²

- *Active Record Pattern*:

²Fonte: Wikipedia https://en.wikipedia.org/wiki/Don%27t_repeat_yourself

O padrão Active Record sugere uma interface específica para acessar objetos em um banco de dados relacional contendo funções tais como INSERT, UPDATE, DELETE, etc.

Uma tabela ou *view* será associada a uma classe então uma instância de objeto estará associada a uma única entrada na respectiva tabela.

Em Rails a biblioteca ActiveRecord implementa o padrão ORM e além disso acrescenta herança e associações resolvendo dois problemas substanciais do padrão.

ActiveRecord é o "model" padrão do componente MVC porém é possível trocá-lo por outra implementação do framework Rails caso o desenvolvedor prefira.

Em um sentido mais amplo Rails é mais que uma biblioteca de Software ou API: é um projeto central de uma vasta comunidade que produz plugins para facilitar e construir projetos complexos de web site. Foi graças a essa comunidade que compartilham ferramentas e contribuem para o projeto Rails criando bibliotecas open source (chamadas de Ruby Gems ou apenas Gems ³) que o uso de rails tem crescido de forma significativa nos últimos anos ganhando fama entre startups devido a sua facilidade e desenvolvimento que permite criar sistemas complexos e testar idéias rapidamente.

4.2 Heroku

Heroku é uma Paas implementada utilizando *cloud computing* criada em 2007 utilizada como um modelo de *Deployment* para Aplicações Web.

A rede do Heroku roda os aplicativos em containers virtuais que executam um ambiente de execução confiável. Esses containers são chamados de Dynos e podem rodar códigos escritos em Nodejs, Ruby, PHP, Go, Scala, Python, Java, Clojure.

O *deploy* de uma aplicação no Heroku é bastante simples o que contribuiu para sua popularização principalmente entre startups que necessitam de agilidade para subir novas atualizações.

A aplicação é enviada para o Heroku por meio de uma conexão direta via GitHub, Dropbox ou alguma API.

Enviado o código-fonte ele então é convertido em uma aplicação interpretando as dependências de outras bibliotecas seguindo o padrão de cada linguagem. No caso do USP Eventos que foi feito utilizando Ruby as dependências ficam armazenadas no próprio Gemfile da aplicação.

Feito o upload da aplicação um container com uma virtualização de Unix é disponibilizado, o Dyno da aplicação. Tal container é pre-carregado com algumas configurações prévias da aplicação: um nome gerado automaticamente, variáveis de ambiente e *add-ons* se existirem.

O Heroku então inicializa o Dyno com a aplicação, carrega-a e então realiza o *deploy* da mesma.

Através do DNS Server oferecido pelo próprio Heroku a aplicação fica acessível por meio de um domínio na forma <nome da aplicação>.herokuapp.com sendo possível redirecionar seu domínio particular para refletir o DNS disponibilizado. ⁴

³Fonte: Wikipedia <https://en.wikipedia.org/wiki/RubyGems>

⁴Fonte: Wikipedia <https://en.wikipedia.org/wiki/Heroku>

4.3 Travis CI

Travis CI é um serviço de integração contínua usado para testar projetos hospedados no Github.

Toda vez que um commit para o repositório selecionado no Github o Travis irá executar as diretrizes especificadas no arquivo `travis.yml` que contém os comandos necessários para rodar os testes automatizados da aplicação, como é o caso do USP Eventos (figura ??).

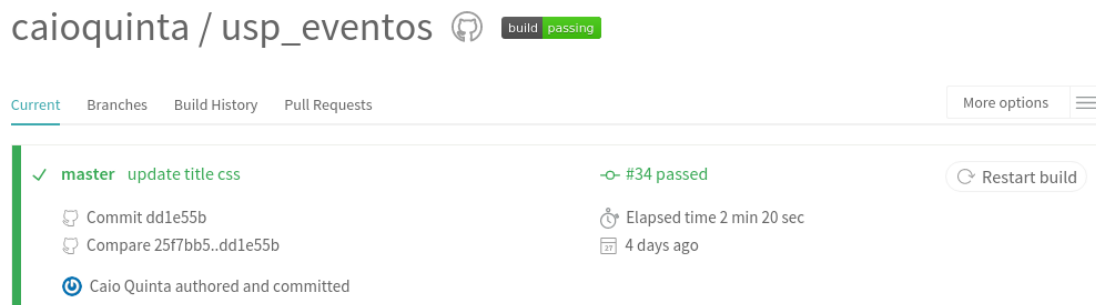


Figura 4.1: O repositório USP Eventos no Travis CI.

4.4 Trello

TODO

4.5 Google Analytics e Google Tag Manager

TODO

4.6 Painel de opiniões Populares - POP

Com o intuito de definir o interesse do público alvo por meio de uma enquete colaborativa utilizamos o POP como sistema de votação devido a sua propriedade interativa na qual os usuários poderiam adicionar itens à enquete principal.

Desenvolvido por estudantes do próprio IME dentro da disciplina de Laboratório de Programação Extrema à pedido da INDX o POP é uma plataforma de pesquisa de opinião pública que possui o objetivo de realizar enquetes nas quais os próprios usuários possam interagir e criar novas opções com o intuito de medir quais são as prioridades de projetos para a região.

Foi permitida a utilização da plataforma implementada em uma instância separada com o nome de POP-TCC. para realização da enquete junto à comunidade USP para definir qual seria o tema do projeto.

Foi feita uma modificação no sistema POP original para o POP-TCC na qual os usuários só poderiam votar de maneira positiva nas enquete ao contrário do sistema original que permitia votos negativos e até ocultamento dos itens da enquete que fossem muito negativados pelos usuários.

Capítulo 5

Usp Eventos

5.1 Definição do Projeto

5.1.1 Motivação

A idéia de desenvolver um sistema utilizando Métodos Ágeis e conceitos de Lean Startup surgiu em dezembro de 2015.

O objetivo era desenvolver um sistema web ou aplicativo voltado para a comunidade USP com a intenção de facilitar de alguma forma o dia-a-dia dos usuários.

Inicialmente existiam algumas propostas de projeto que foram então formalizadas em uma enquete realizada junto a comunidade USP.

5.1.2 Enquete e definição do projeto

No início as seguintes hipóteses de interesse de projeto foram disponibilizadas para votação:

- Usp avisa eventos e incidentes: Um sistema para reportar desde eventos acontecendo no campus (palestras, festas, etc) até outros incidentes (buracos, perigos, etc).
- Usp doações e trocas: Um sistema voltado para os membros da comunidade que desejam fazer doações (sejam móveis, roupas usadas, etc) para outros e aqueles dispostos a receber. organizando filas de interesse, disponibilidade e urgência.

Com o intuito de entender melhor nosso público alvo além de estarmos abertos à outras sugestões precisávamos de um sistema que suportasse não só uma votação fechada como também permitisse que os próprios usuários fossem capazes de adicionar outras propostas de forma dinâmica àquelas já existentes.

Pensando nisso resolvemos utilizar o sistema POP (Painel de Opinião Pública) para efetuar a enquete.

Foi então criada uma instância independente do sistema adaptada chamada POP-TCC (figura ??) utilizando o Heroku que poderia ser acessada pelo endereço pop-tcc.herokuapp.com.

Em 11/01 foi enviado o primeiro e-mail com a enquete do POP-TCC aberta para a lista de e-mails dos alunos do IME com as duas opções iniciais de projeto supracitadas.

A divulgação da enquete concentrou-se principalmente via facebook nas seguintes comunidades:

Ao longo de duas semanas outras opções de projeto surgiram. O resultado final ?? da enquete e a descrição das sugestões seguem abaixo:

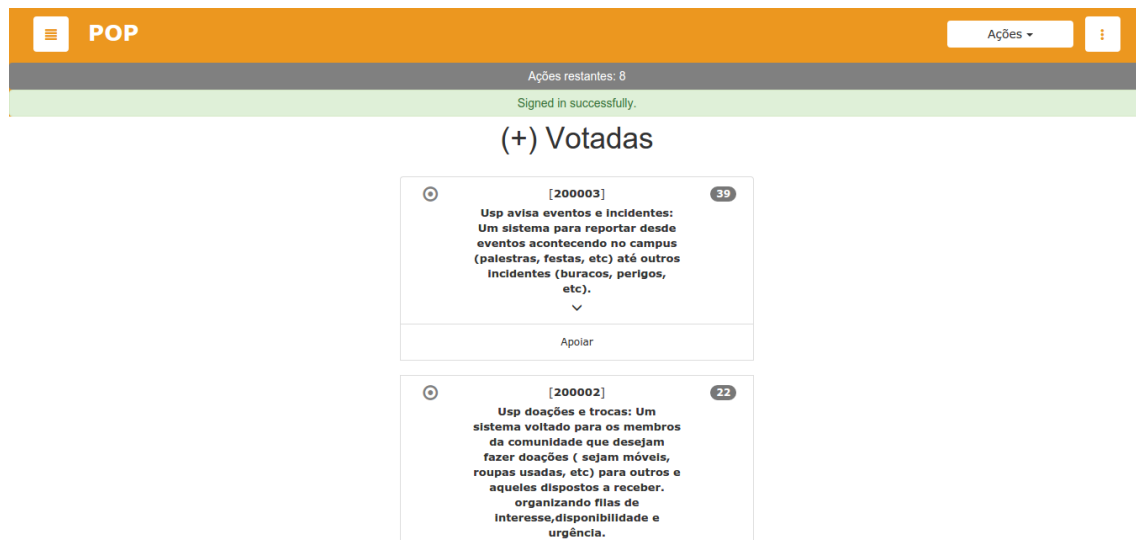


Figura 5.1: Sistema POP-TCC com a enquete para votação.

Tabela 5.1: Comunidades do Facebook na qual foram feitas divulgações

Comunidade	Número de Membros
USP - Universidade de São Paulo	
FAU USP	4000
IME USP	3000
Universidade de São Paulo	5000
Baladas USP	15000

- 1º Lugar (39 votos): Usp avisa eventos e incidentes: Um sistema para reportar desde eventos acontecendo no campus (palestras, festas, etc) até outros incidentes (buracos, perigos, etc).
- 2º Lugar (21 votos): Usp doações e trocas: Um sistema voltado para os membros da comunidade que desejam fazer doações (sejam móveis, roupas usadas, etc) para outros e aqueles dispostos a receber. organizando filas de interesse, disponibilidade e urgência.
- 3º Lugar (20 votos): Usp caronas: os motorizados colocam horário, bairro, quantidade de lugares disponíveis, ponte de embarque e desembarque. os interessados enviam um alerta para os motorizados confirmarem até preencherem as vagas.
- 4º Lugar (18 votos): Mapa de crimes na usp: um app onde roubos, furtos, assaltos, agressões, assédios, discriminações e outros crimes podem ser relatados, georreferenciados no campus.
- 5º Lugar (14 votos): Volta pedalusp - volta e desenvolvimento da ideia que já teve adesão, mas morreu por falta de manutenção. implementação de novos pontos de troca de bikes mais próximos das faculdades e outros pontos estratégicos.
- 6º Lugar (13 votos): Usp extensão: uma plataforma de apoio mútuo para organização, contato, criação e divulgação de projetos de extensão dentro da universidade.
- 7º Lugar (12 votos): Usp gigabyte: um ponto de encontro virtual para reunir o pessoal e tomar uma rodada de suco com a galera. curtidão, azaração e muita gente bonita neste, que é o ponto mais quente da internet!

- 8º Lugar (8 votos): Monitoria voluntária: pessoas divulgam horário e local no qual pessoas podem procurá-las para tirar dúvidas sobre certas disciplinas comuns a vários cursos.

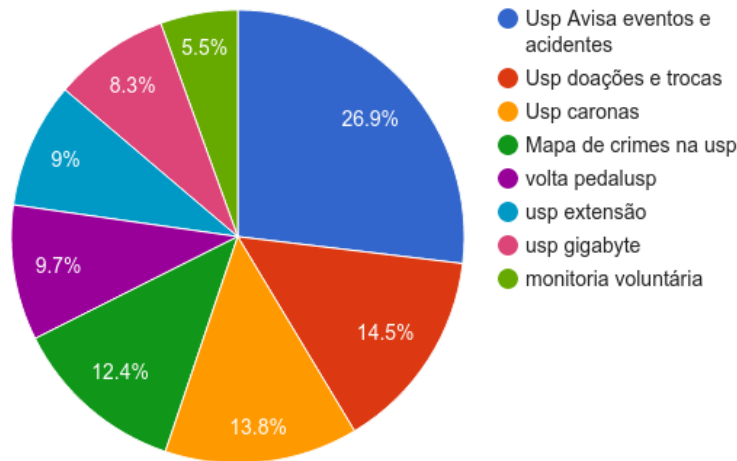


Figura 5.2: Gráfico gerado pelo resultado da Enquete.

O projeto escolhido então foi o Usp Avisa Eventos e Acidentes que seria renomeado apenas para USP Eventos.

5.2 Definindo as características do Sistema

5.2.1 Pesquisa de Sistemas Semelhantes

Definido o tema do projeto teve início uma pesquisa para levantar a existência de sistemas que tivessem uma proposta semelhante de divulgação de eventos e atividades.

Ao realizar essa pesquisa expandimos o escopo para que os sistemas não necessariamente fossem voltados para fins acadêmicos apenas tendo como objetivo principal obter uma base de conhecimento de quais funcionalidades um sistema de divulgação possui ou mesmo encontrar uma solução em código aberto que pudesse ser utilizada como base.

Tal abordagem de pesquisas de concorrentes e projetos semelhantes está presente na metodologia Lean Startup cuja principal justificativa é obter o máximo de conhecimento prévio por meio de idéias já consolidadas e aplicadas por outros. (inserir referencia)

Também Foi enviado um e-mail em março para a lista de Representantes Discentes do próprio IME questionando sobre funcionalidades e sugestões.

Das compilações sobre a pesquisa e resposta obtidas por e-mail destacam-se 3 sites com proposta e conteúdos semelhantes:

- Eventos USP (<http://www.eventos.usp.br/>): Canal de divulgação oficial da USP para eventos ocorrendo em suas dependências para todos os campus.
 Ponto Fortes: Sendo o canal de comunicação oficial da universidade ele mostra-se com uma ótima fonte de conteúdo sobre eventos oficiais.
 Ponto Fracos: É uma via de mão única na qual o usuário apenas se informa dos detalhes do evento mas não tem oportunidade de criar ou divulgar o seu próprio.

- Catraca-Livre (<https://catracalivre.com.br/brasil/>): Principal Portal de atividades culturais e divulgação de eventos costuma ser bastante eclético.
Pontos Fortes: Apesar de não ter um conteúdo personalizável a página de Agenda possui uma grande variedade de opções de filtro. Pontos Fracos: Como divulga eventos por toda a cidade sua navegação é bastante confusa levando o usuário a facilmente perder-se devido à grande quantidade de informações e poucas opções de personalização. Soma-se a isso o fato de que o site é uma forma de comunicação unilateral não permitindo que usuários divulguem e organizem seus próprios eventos.
- SP Cultura (<http://spcultura.prefeitura.sp.gov.br/>): Portal oficial da prefeitura para divulgação de atividades por toda a cidade. Pontos Fortes: Possui bastante opções de filtros além de permitir que um usuário interessado crie um evento e submeta-o para aprovação, bastante intuitivo e de fácil acesso. É baseado em uma solução de código aberto. Pontos Fracos: Os eventos estão distribuídos por um mapa georreferenciado, dando mais ênfase à localização do evento do que sobre sua descrição obrigando o usuário a selecionar primeiro um evento em uma localização para então obter informações do mesmo.

5.2.2 Plataforma Web x Móvel

Para decidir em qual plataforma desenvolver nosso sistema levamos em consideração fatores técnicos e do público-alvo.

De acordo com o gráfico ?? observa-se que as plataformas Android e IOS hoje em dia correspondem respectivamente à 86,2% e 12,9% do mercado mundial de sistemas operacionais mobile totalizando juntas 99,1% do mercado o que significaria que ao desenvolver dois aplicativos nativos para ambas as plataformas corresponderia à ter a acesso à quase totalidade do mercado móvel. ¹

Foram feitas as seguintes considerações quanto ao desenvolvimento de uma aplicação nativa. ²

- O desenvolvimento para Android é feito utilizando a linguagem Java a partir de APIs fornecidas pelo próprio Google enquanto um aplicativo para IOS utiliza Objective-c ou Swift por meio das APIs fornecidas pela Apple. Devido a isso, não há correspondência de código entre ambas plataformas tornando necessário o desenvolvimento de duas aplicações distintas caso queira-se atingir a totalidade do mercado.
- Os aplicativos nativos seguem um padrão bastante rígido de UI/UX que divergem bastante entre si. Os padrões de desenvolvimento para interfaces de um aplicativo Android diverge totalmente de um aplicativo IOS quanto a sua usabilidade e layout fazendo com que seja necessário pensar em duas interfaces distintas.
- A maior vantagem de um aplicativo nativo é ter acesso aos recursos de *hardware* do smartphone tais como câmera ou acelerômetro. Como nossa aplicação não faria uso de nenhum recurso específico tais vantagens não seriam aproveitadas.
- Os aplicativos nativos estão sujeitos às normas e aprovações de suas lojas virtuais, Play Store para o Android e Apple Store para o IOS, fazendo com que o tempo de

¹Fonte: Statista <http://www.statista.com/statistics/254653/mobile-internet-user-penetration-in-brazil/>

²Fonte: Caelum <http://blog.caelum.com.br/aplicacoes-mobile-web-ou-nativa/>

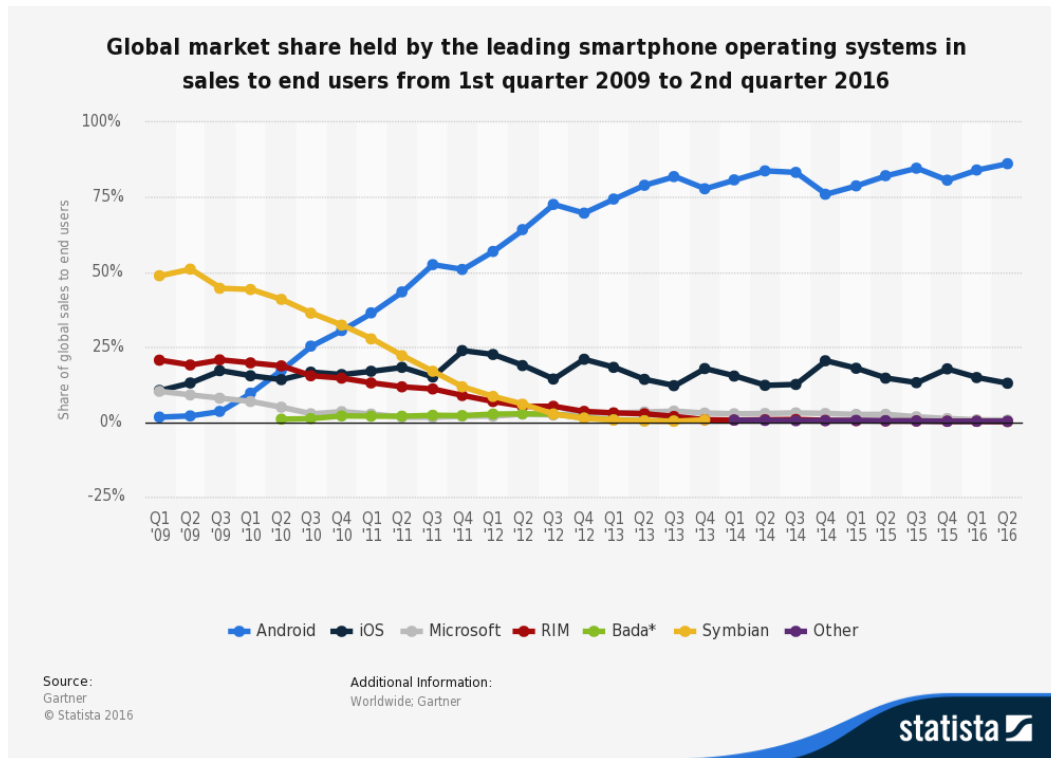


Figura 5.3: *Distribuição de mercado para Sistemas Mobile.*

publicação de uma atualização aumente devido a necessidade de aprovação da loja em questão.

- Não seria possível utilizar a plataforma em Desktops restringindo o público alvo.

Para tomar a decisão de desenvolver uma plataforma web foi levado em consideração principalmente a facilidade de atualização de uma aplicação web por não necessitar da aprovação de uma loja online devido a necessidade de desenvolvimento segundo o modelo proposto pelo modelo *Lean Startup* e possibilidade de acesso por meio de diversas plataformas (tanto via desktop como móveis) com uma única versão da aplicação pois ainda que exista o custo de abrir mão de uma interface específica com a identidade visual de cada plataforma demos prioridade a esse modelo para atingir um número maior de usuários.

Apesar da escolha de um sistema web ao analisar o crescimento do acesso móvel no Brasil ?? que vem aumentando a passos largos no país houve a preocupação de desenvolver-se uma aplicação web híbrida com uma interface totalmente responsiva ³ desde o princípio.

Dessa forma apesar do acesso em um smartphone não ser tão intuitivo quanto em uma aplicação nativa ainda sim teria uma interface funcional garantindo que a navegação em uma plataforma móvel fosse feita sem dificuldades.

5.2.3 Porque escolher Ruby on Rails

O uso do Ruby on Rails entre startups tem crescido nos últimos anos isso devido a alguns fatores intrinsecamente ligados a própria estrutura do arcabouço rails e também

³ Uma interface responsiva de um site ou página é uma versão do layout adaptada para uso em telas menores comumente refere-se a visualização em smartphones

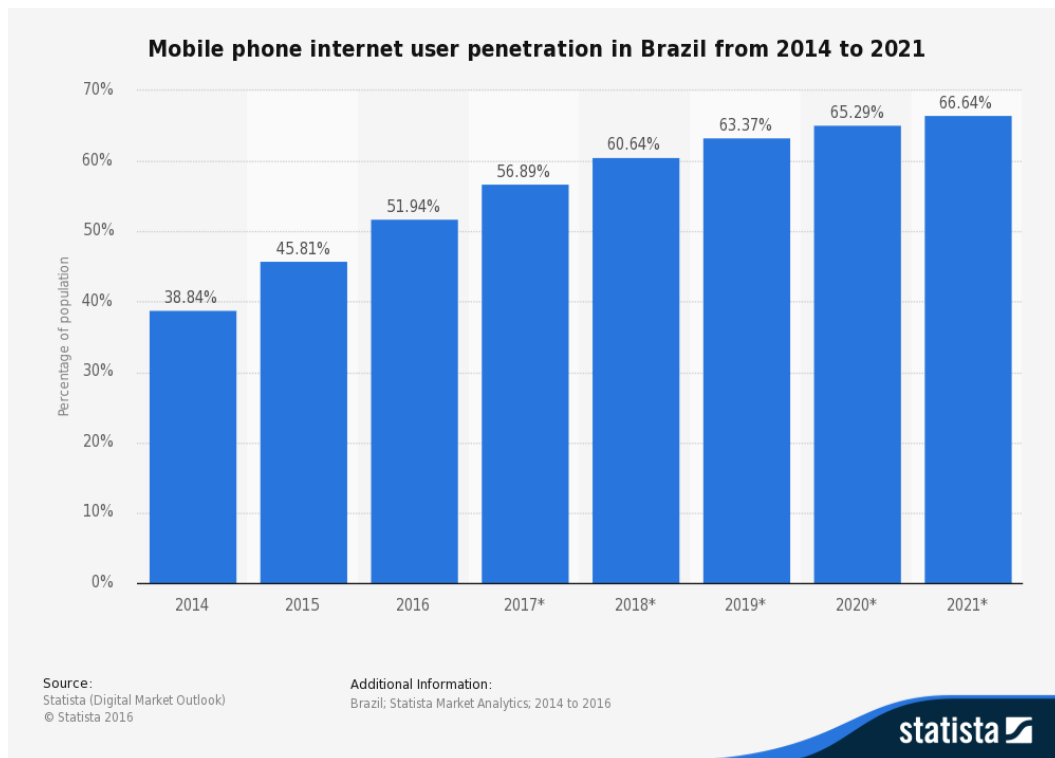


Figura 5.4: *Crescimento do Uso de internet móvel no Brasil.*

à necessidade que startups e o próprio modelo de *Lean Startup* exigem para iterações e alterações de códigos rápidas.

Uma das principais filosofias do Rails, o CoC Convenção sobre Configuração, permite que o desenvolvedor possa, a partir de um conjunto pré-definido de configurações padrões, agilizar o desenvolvimento do código ao tirar de sua responsabilidade fatores de configuração em detrimento de um padrão já estabelecido dando assim ao desenvolvedor mais tempo para concentrar-se em decisões sobre o produto. Essa agilidade em codificar rapidamente resulta em interações mais rápidas contribuindo para uma maior agilidade dentro do ciclo de Contruir-Medir-Aprender.

Outro fator inerente a cultura Rails é o grande enfoque em testes automatizados. Para o desenvolvimento da plataforma USP evento foram utilizadas as gems Rspec e Capybara que permitem não só a realização de testes unitários como também testes de integração de modo muito rápido e direto garantindo assim a confiabilidade do seu projeto evitando que um *bug* imprevisto prejudique seu aprendizado sobre o MVP.

Soma-se a isso a quantidade enorme de ferramentas que auxiliam na integração contínua do código como por exemplo o Travis CI utilizado durante o desenvolvimento da plataforma.

O Travis CI era responsável para que cada commit feito ao repositório principal do projeto fosse executada toda a bateria de testes automatizados enviando um e-mail contendo um relatório sobre o seu resultado, inclusive em caso de falha, garantindo dessa forma que todos tivessem sempre conhecimento das alterações sobre o código além de evitar problemas de conflitos de versões ou mesmo que uma atualização de projeto fosse colocada no ambiente de produção contendo alguma falha.

O fator comunidade é outra vantagem do RoR. Destacando-se pelo seu tamanho, interesse e acessibilidade em tirar dúvidas sempre contribuindo para promover o arcabouço a comunidade que orbita ao redor do Rails foi capaz de criar ferramentas prontas para uso com uma ótima documentação, tutoriais, cursos e guias garantindo assim que qualquer de-

envolvedor interessado sempre tenha em mãos materiais e ferramentas de qualidade para iniciar o desenvolvimento do seu projeto.

Acrescenta-se também à favor do arcabouço Rails sua escalabilidade, rails é utilizado por empresas de grande porte tais como Groupon, Twitter, Basecamp, mostrando que o arcabouço é um arcabouço robusto capaz de lidar com grandes sistemas sem ter uma queda de desempenho.⁴

Para finalizar Rails é também um arcabouço seguro garantindo mesmo em sua configuração padrão ou com o auxílio de alguns plugins proteção contra SQL-Injections e XSS(*Cross Site Scripting*). Além disso os programadores devem seguir o *Secure Life Cycle Development*, figura ??, proposto pela Microsoft, um modelo de desenvolvimento de software cujo principal objetivo é ajudar o a construir software mais seguros e confiáveis e reduzir custos⁵.



Figura 5.5: *Fases do Secure Life Cycle Development.*

5.3 Kanban

Kanban é uma palavra japonesa e significa cartão visual para prover informação de regulamentação do fluxo de estoque e materiais. Possui três regras principais (KNIBERG, H., 2009): visualizar o fluxo de trabalho; limitar o trabalho em cada estágio do fluxo, e medir o tempo de avanço (tempo médio para se completar cada item) (inserir referência).

No contexto de Métodos Ágeis e *Lean Startup* foi utilizado uma abordagem utilizando o Kanban para definir as hipóteses de produtos, o escopo de desenvolvimento e as tarefas para serem executadas durante as etapas de desenvolvimento.

Tradicionalmente o kanban para desenvolvimento de Software possui 3 estágios:

- TO DO: referente a requisitos que ainda estão aguardando para serem desenvolvidos.
- DOING: referente a requisitos que estão sendo desenvolvidos.
- DONE. referente a requisitos que já finalizaram e foram devidamente revisados e testados. (inserir referência).

Cada item adicionado na fila de TO DO é chamada de tarefa. Uma hipótese à ser testada pode ser transformada em uma série de tarefas pequenas à serem completadas durante o período de desenvolvimento.

Caso seja necessário é possível quebrar uma tarefa grande em uma série de tarefas menores. Tomando como exemplo a tarefa de Implementar um filtro de eventos para a página principal. Ela foi quebrada em 3 tarefas menores para serem completadas: Implementar Categorias de Eventos, Permitir adicionar categorias de eventos durante a criação de um novo evento e Implementar filtro de Eventos na página principal de eventos.

O tempo que uma tarefa demora desde sua entrada no quadro até a saída é denominado *Lead Time*. Em um ambiente de Startup o intuito é sempre obter o menor Lead Time possível

⁴Fonte: MLSDev Why startups use Ruby on Rails? <http://mlsdev.com/en/blog/61-why-startups-use-ruby-on-rails>

⁵Fonte: Wikipedia https://en.wikipedia.org/wiki/Microsoft_Security_Development_Lifecycle

para tal é importante estar ciente da taxa de entrega que sua equipe de desenvolvimento é capaz de cumprir e sempre definir as tarefas de modo simples.

Para utilização no projeto USP Eventos foi incluída uma coluna a mais denominada BACKLOG, comumente usada na metodologia Scrum ⁶, na qual foram colocadas idéias que poderiam ou não serem transformadas em tarefas de desenvolvimento ou hipóteses para serem executadas em uma iteração do ciclo de Construir-Medir-Aprender.

Em cada iteração do ciclo especificamos quais hipóteses seriam testadas e a partir delas criamos tarefas para serem implementadas.

Caso um bug fosse detectado uma tarefa era criada na fila de TO DO para resolvê-la.

A ferramenta Trello, figura ??, foi utilizada para simular um quadro Kanban digital, com ela foi possível guiar todo o desenvolvimento do sistema. Com o intuito de tornar melhor a

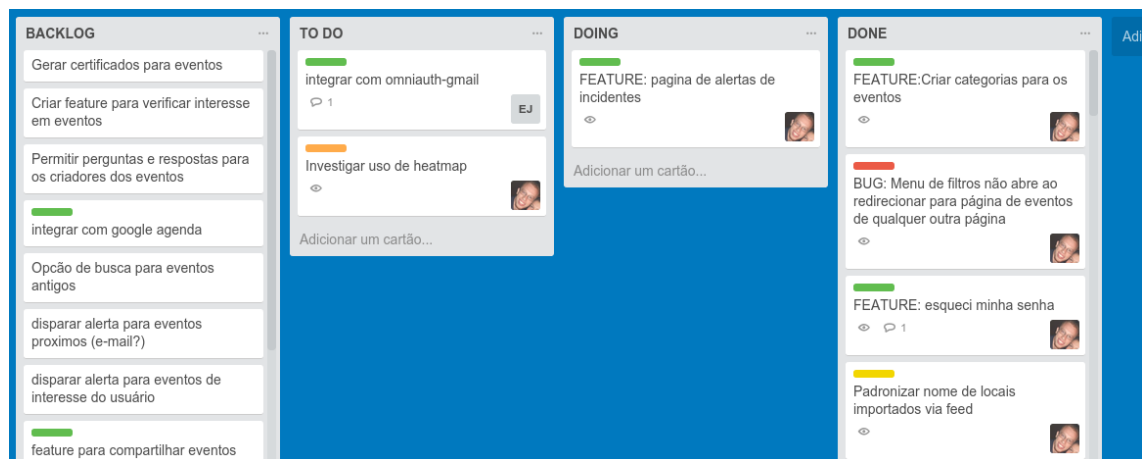


Figura 5.6: Sistema Trello contendo as 4 colunas utilizadas no USP Eventos.

visualização no Trello foram criados alguns Rótulos de cores distintas para cada tarefa:

- BUG (vermelho): Defeito não previsto durante o desenvolvimento.
- FEATURE (verde): Nova funcionalidade para ser implementada.
- REFACTOR (lilás): Melhoria de código sem refletir uma mudança externa.
- MELHORIAS (laranja): Investigar ou implementar o uso de ferramentas externas ao sistema, tal qual a integração com o Google Analytics por exemplo.
- QUICK WIN (amarelo): Melhoria feita rapidamente e não prevista durante a definição de tarefas.

5.4 Primeira Iteração

5.4.1 Construção

< em desenvolvimento >

O primeiro MVP do USP Eventos tinha como objetivo testar as seguintes hipóteses:

- Medir o interesse do público em participar de um evento.

⁶Fonte: Desenvolvimento Ágil [fontehttp://www.desenvolvimentoagil.com.br/scrum/sprint_backlog](http://www.desenvolvimentoagil.com.br/scrum/sprint_backlog)

- Criar uma interface intuitiva e rápida para mostrar informações de eventos para o usuário.

A Página inicial (figura ??) do site possuía acesso para a página de cadastro e login além de um formulário para envio de sugestões.



Figura 5.7: Tela inicial na primeira iteração

O cadastro de usuário (figura ??) pedia apenas nome, e-mail e senha inicialmente porém ainda na primeira iteração foi implementada a opção de login com facebook.



Figura 5.8: Tela de Cadastro na primeira iteração

A página de eventos (figura ??) só poderia ser acessada por um usuário logado tornando essa página e qualquer página de evento específica inacessível para um visitante sem login.

Além disso a página de eventos apenas mostrava-os sem oferecer qualquer opção inicial de filtro.

Todas as páginas foram pensadas também para o acesso via mobile possuindo versões responsivas (figura ??).

Visando evitar que os usuários se depararam com uma página de eventos vazia foi feita uma rake task para consumir o xml gerado pelo feed RSS que o site www.eventos.usp.br. Dessa forma seria possível adicionar de forma mais ágil alguns eventos dentro da plataforma.

Cada thumbnail de eventos presente na página principal de listagem de eventos incluía o nome do evento, localização, data de início e fim além de um botão de "Participar" para que os usuários que estivessem logados e também botões para compartilhamento nas principais redes sociais.



Figura 5.9: Página de Eventos



Figura 5.10: Página de Eventos versão responsiva

Houve também a preocupação de espalhar formulários de Sugestões em diversos pontos do site com o intuito de facilitar a coleta de informações do usuário.

5.4.2 Divulgação

A primeira versão do sistema ficou disponível apartir do dia 5 de maio e sua divulgação foi feita pelo facebook por grupos e comunidades associadas a institutos da USP tais como FFLCH, FAU, IME, ECA, assim como foram enviadas mensagens para as respectivas empresas Júnior e Atléticas.

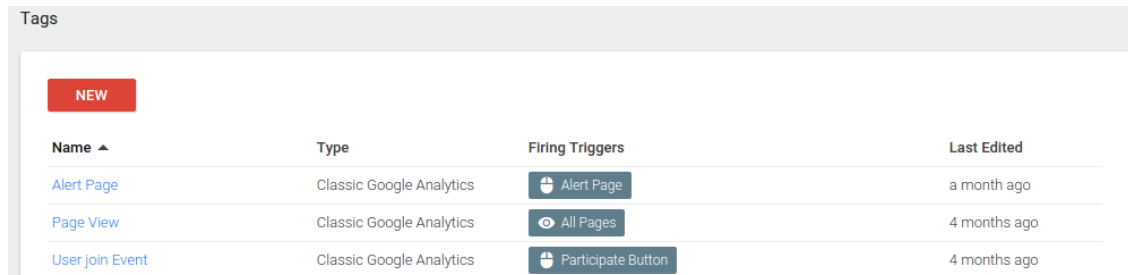
Também divulgamos a primeira versão na lista de alunos e representantes discentes do próprio Instituto de Matemática e Estatística.

5.4.3 Métricas

Para medir o fluxo de usuários dentro do site utilizamos o Google Analytics em conjunto com o Google Tag Manager.

Como métrica chave escolhemos medir a quantidade de usuários que se interessavam por um evento.

Foi criada então uma tag (figura ??) para rastrear os cliques no botão "Participar" presente dentro do thumbnail de um evento na página principal e também na página de detalhes do evento. Dessa forma seria possível mapear o interesse do usuário em um determinado evento.



Name ▲	Type	Firing Triggers	Last Edited
Alert Page	Classic Google Analytics	Alert Page	a month ago
Page View	Classic Google Analytics	All Pages	4 months ago
User join Event	Classic Google Analytics	Participate Button	4 months ago

Figura 5.11: Visualização das Tags pelo Google Tag Manager.

Em paralelo foi possível obter também através do google analytics informações sobre o número de visualizações de páginas. <inserir imagem>

5.4.4 Aprendizado

O maior número de acesso de usuários deu-se sempre em seguida aos posts realizados pelo facebook, alcançando picos de acesso mostrando a importância da divulgação pela plataforma.

Foi observado que os usuários acessam o site porém não realizavam cadastro deixando o site logo em seguida. Alguns feedbacks feitos diretamente relataram que não tiveram confiança para realizá-lo e também não viam vantagem em fazê-lo.

Dentre os feedbacks recebidos através do formulário do site, e-mails e de forma direta compilamos algumas críticas e sugestões:

- Página de eventos e visualização dos mesmos deveriam ser abertas para usuários mesmo sem login.
- Ausência de um filtro de usuários tornou a página de eventos confusa para navegação.
- Botão lateral de adicionar evento estava muito grande e atrapalhando a navegação.
- Falta de cores na página principal tornou cansativa a navegação.
- Clicar no nome do evento no thumbnail para acessar a página do mesmo.
- Ausência de opção de "esqueci minha senha".
- Clicar no botão "Participar" não tinha uma utilidade prática. O evento era salvo porém isso gerava nenhum reflexo no sistema não existindo uma funcionalidade que justificasse sua existência e não havendo razão para que os usuários clicarem no botão.

Alguns comentários selecionados:

Por Veronica: "Seria muito legal poder filtrar os eventos por tags referentes ao local / tipo / assuntos que serão abordados!"

Por Lucas: "Olá! Eu gostaria de ver os eventos por categoria/área de conhecimento (Artes, História, Economia, Engenharia, etc)."

Por Carolina: "Por que devemos nos cadastrar simplesmente para acessar o site? E se a pessoa "simplesmente quer se informar sobre o que está acontecendo"? A minha sugestão é que somente quem quer enviar eventos para o site deveria se ter que se cadastrar. Obrigada e boa sorte no TCC."

Por Karina: ". [Login] Aos usuários que não possuem conta, mas tentam logar, seria ideal que o sistema mostra-se quando o usuário colocou dados incorretos e quando o usuário não possui conta.

. [Página de eventos] Seria melhor que a célula do evento permitisse o click para adentrar detalhes sobre o mesmo

. [Página de eventos] Inserir algumas ferramentas de filtro: datas (início/final ou datas pontuais); tags (algumas tags pré-cadastradas); campus;

. [Página de eventos] Espaço para uma imagem nas células de divulgação do evento daria mais cor e chamaria mais a atenção dos usuários

. [Página de eventos] Seria legal colocar um aviso de inscrições limitadas para eventos que têm tal restrição"

5.5 Segunda Iteração

5.5.1 Construção

Levando em consideração o aprendizado da primeira iteração foi feita uma mudança no fluxo do site para permitir o acesso para a página de eventos sem a necessidade de realizar um cadastro antes ou exigir um *login* do usuário.

Foi criado um filtro para a página de eventos baseado na utilização de tags dessa forma ao criar um novo evento (figura ??)o usuário agora pode escolher 3 dentre 12 tags pré-definidas que servirão como filtro nas página principal.

Figura 5.12: *Página de Cadastro de Novos eventos com filtros*

Visando tornar a navegação dentro da página de eventos mais fluída e menos cansativa foram realizadas algumas modificações visuais na exibição dos eventos (figura ??)

- Remodelagem do thumbnail de Eventos
- Diminuição do botão de adicionar novos eventos para não atrapalhar a navegação

- Adição de um menu lateral com opções de Filtros para os eventos
- Nova listagem personalizada de Eventos segundo os interesses do usuário

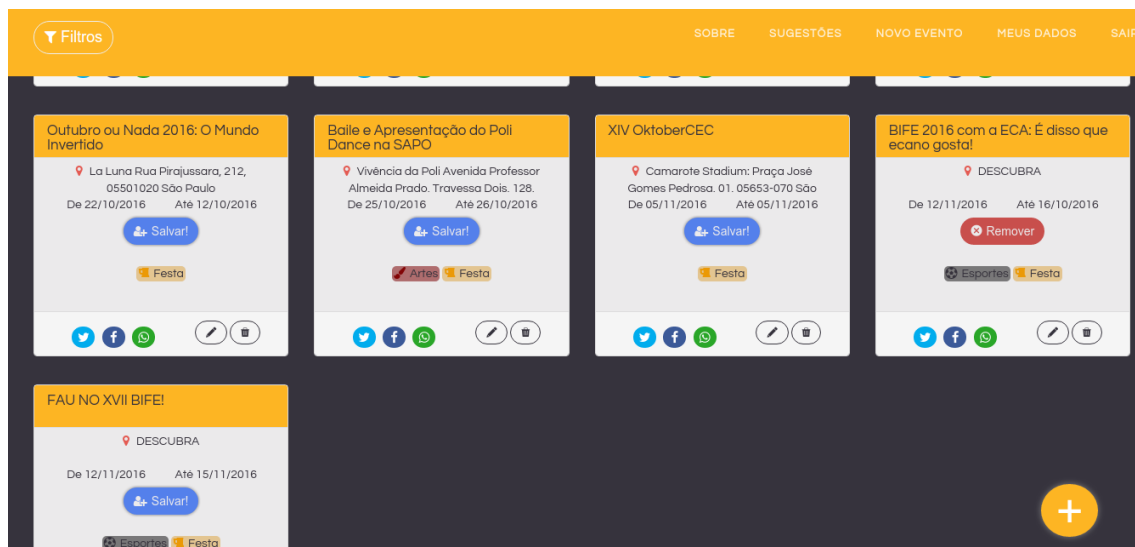


Figura 5.13: *Página de Eventos com as alterações para a segunda interação*

Os filtros (figura ??) estão em um menu lateral que é acionado por um botão na parte superior esquerda. Também é possível selecionar cada *tag* individualmente ao clicar sobre a respectiva nos thumbnails de eventos. As mudanças realizadas (figura ??) no thumbnail de eventos:

- Removido botão de +Info, agora para acessar mais informações basta clicar sobre o nome do evento
- Adicionado Cabeçalho para separar e dar maior ênfase para o título e uma cor de fundo para aumentar o contraste com o plano de fundo a fim de facilitar a leitura
- Adição de tags com as classificações dos eventos facilitando a escolha do mesmo.
- Mudança do nome do botão de "Participar" para "Salvar"

Em conjunto com a criação das tags para eventos foi criado um mecanismo de preferências para o usuário, agora na página de cadastro ou edição de usuário é possível selecionar as tags com as quais o usuário tenha maior afinidade com o intuito de exibir uma listagem personalizada de eventos segundo esses critérios. ⁷

Aproveitando o sucesso do lançamento do aplicativo Pokemon GO ⁸ que gerou uma enorme quantidade de acessos e movimentação pelo Campus foi criado uma página de Alertas ?? no Campus visando atingir o público que estava utilizando o aplicativo para divulgar a localização de Pokemons.

5.5.2 Divulgação

Além da divulgação pelo facebook foram espalhados cartazes em pontos estratégicos pela USP tais como Pontos de Ônibus com grande movimentação, murais próximos aos Bandeijões e também no interior de alguns institutos.

<fotos cartazes no pontos de onibus>

⁷Fonte: Desenvolvimento Ágil http://www.desenvolvimentoagil.com.br/scrum/sprint_backlog

⁸Fonte: Pokemon GO https://en.wikipedia.org/wiki/Pokemon_Go

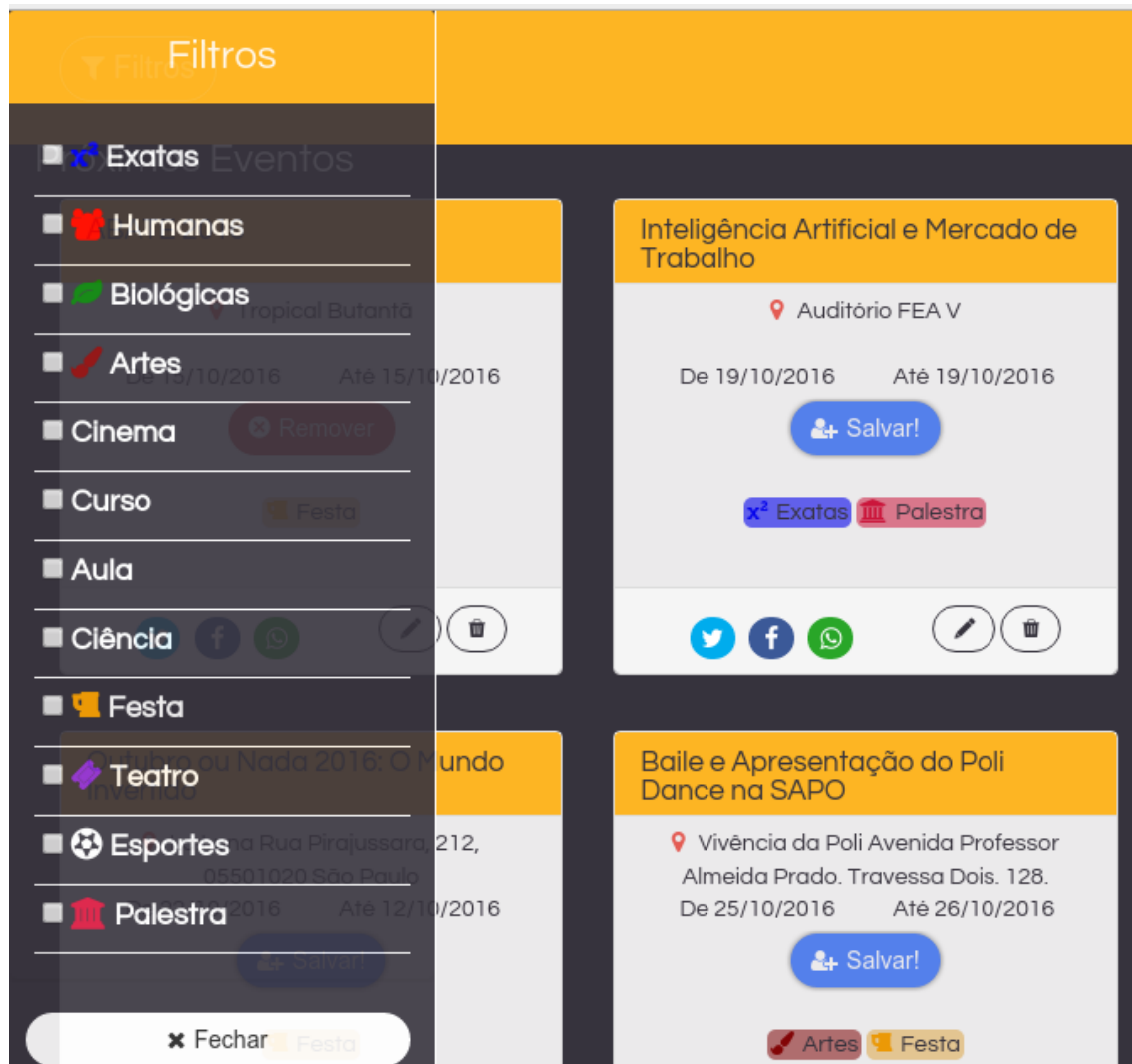


Figura 5.14: *Filtros na página de Eventos*

5.5.3 Métricas

Foi mantido o rastreamento no botão "Participar" porém seu nome foi alterado para "Salvar" para refletir melhor sua utilidade de salvar um evento como interessante para o exibi-lo na seção de "Meus Eventos" da listagem de eventos do usuário.

A nossa métrica chave continuou sendo avaliar o interesse dos usuários em determinado Evento por meio do clique no botão "Salvar".

Com o lançamento da seção de Alertas foi feita uma divulgação via Facebook incentivando os usuários a utilizarem a plataforma com o intuito de divulgar a localização de Pokemons, tal iniciativa rendeu um grande pico de acesso ao site. Foi colocado também uma tag para rastrear o número de clicks no botão "Alertas" com a intenção de medir o interesse na funcionalidade. <grafico google analytics>

5.5.4 Aprendizado

Com a abertura da página de eventos sem a obrigatoriedade de um cadastro mais usuários acessaram a página principal de Eventos porém quase não houveram novos cadastros dificultando assim que um usuário salvasse algum evento para sua lista ou tivesse algum incentivo para retornar ao site.



Figura 5.15: *Esquerda: versão antiga. Direita: Versão atualizada*

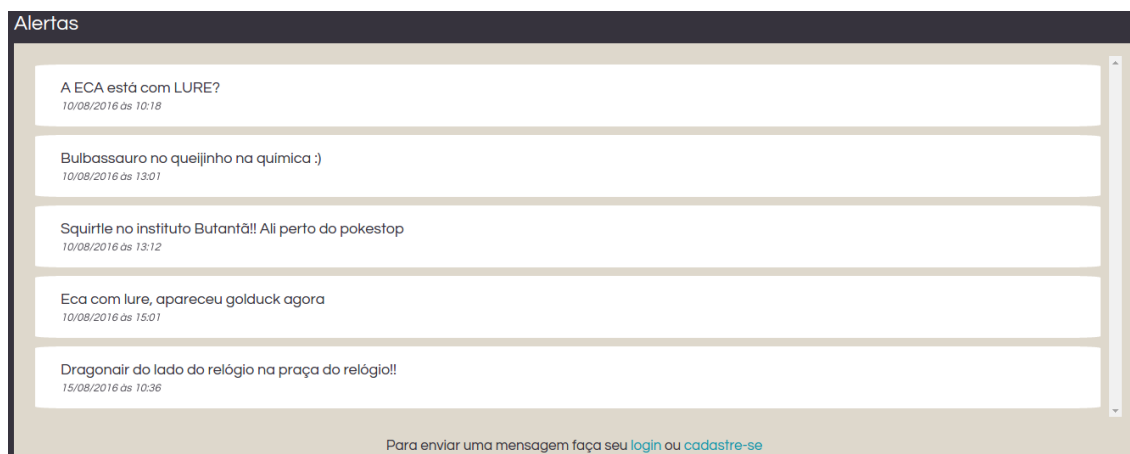


Figura 5.16: *Página de Alertas*

Apesar do pico de acessos com o lançamento da página de Alertas passado o período de pico a funcionalidade foi abandonada pelos usuários, gerando poucos acessos mostrando que talvez não fosse interessante investir em seu desenvolvimento.

Mesmo com a criação e filtros e melhorias visuais a página principal ainda carecia de apelo para navegação.

Alguns comentários recebidos mostrou que ainda seria necessário torná-la mais atrativa visualmente e amigável para o usuário.

Novamente recebemos comentários sobre adicionar a opção de adicionar uma foto ao evento, segue: "Sinto falta de anexo para cartazes. poder enviar uma foto ou cartaz escaneado do evento.- Ferdinand Machado

Capítulo 6

Conclusões

[illegible]

¹Exemplo de referência para página Web: www.vision.ime.usp.br/~jmena/stuff/tese-exemplo

Apêndice A

Título do apêndice

[illegible]

