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

Caio Teixeira da Quinta
Eugênio Augusto Jimenes

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

São Paulo
Dezembro de 2016

Desenvolvimento de uma 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. Alfredo Goldman vel Lejbman
Cosupervisor: Jorge Melegati

São Paulo
Dezembro de 2016

Agradecimentos

0.1 Caio

Primeiramente, gostaria de agradecer ao professor Alfredo e ao Jorge por acreditarem no projeto desde o começo e darem todo o suporte e ajuda necessárias para seu desenvolvimento ao longo do ano, muito obrigado.

Aos amigos Thiciene, Nico, Denise, Rafel Fuzitani, Everton e todos aqueles que de alguma forma ajudaram ou simplesmente me aguentaram nesse período, muitíssimo obrigado.

À Fernanda, minha namorada, pelo suporte, atenção e conselhos valiosos. Muito obrigado por me motivar e escutar sempre que eu precisei, te amo.

Não foram poucas as dúvidas e problemas pessoais que tive ao longo de toda a graduação então seria injusto não dedicar esse trabalho aos meus pais que acima de tudo nunca deixaram de acreditar em mim mesmo quando eu já havia desistido. Dizer que esse trabalho também é de vocês ainda é muito pouco então prefiro dizer que é apenas a primeira de muitas conquistas que virão. Obrigado por estarem sempre ao meu lado.

Caio Teixeira da Quinta, Novembro de 2016

0.2 Eugenio

Agradeço ao orientador, professor Alfredo Goldman vel Lejbman, e ao coorientador, Jorge Melegati, pelo apoio e suporte neste projeto. Muito obrigado pela liderança e paciência.

Dedico este trabalho, especialmente, à minha irmã, Carolina Jimenes, que sempre me apoiou e incentivou desde antes da graduação. Provou-me que dislexia não é uma deficiência, mas sim uma outra forma de ver e enfrentar o mundo.

Eugenio Augusto Jimenes, Novembro de 2016

Resumo

Neste texto descreveremos todo o ciclo de desenvolvimento da plataforma USP Eventos. A ferramenta foi criada a partir da constatação de que a Cidade Universitária possui grande diversidade acadêmica e cultural que manifesta-se em uma variedade de eventos que são promovidos e realizados em toda sua extensão. Levando em consideração essa grande quantidade de eventos é esperado que existam problemas para sua divulgação. Ao realizar uma enquete junto à comunidade USP constatou-se a necessidade de um sistema para centralizar a divulgação desses eventos. Como consequência desse resultado foi proposto criar o USP Eventos. Para auxiliar no desenvolvimento do sistema foi utilizada uma abordagem que combinou a metodologia *Lean Startup* com conceitos de Métodos Ágeis. Ao longo do processo foi possível observar os benefícios e desvantagens das metodologias escolhidas e desenvolver um sistema direcionado aos interesses dos usuários.

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

Abstract

In this text we will describe the entire development cycle of the platform USP Events. The tool was created on because the Campus has great academic and cultural diversity, manifesting itself in a variety of events that are promoted and done throughout the Campus' extension. Taking in consideration this great amount of events, some trouble in disclosure them is expected. By making an enquiry to the USP community it was determined the need for a system that would centralize the propagation of these events. As a consequence of this result, the creation of USP Events was proposed. To help the system development an approach combining the methodology Lean Startup with the concepts of Agile Methodologies was used. Throughout the process it was possible to observe the advantages and disadvantages of the chosen methodologies and to develop a system directed to the user's interests.

Keywords: agile methodologies, lean startup, web development.

Sumário

Lista de Abreviaturas

MVP	Produto Mínimo Viável (*Minimum Viable Product*)
CoC	Convenção sobre Configuração (*Convention over Configuration*)
DRY	Não se repita (*Don't Repeat yourself*)
ORM	Mapeamento Objeto Relacional(*Object Relational Mapping*)
POP	Painel de Opinião Popular
UX	Experiência do Usuário (*User Experience*)
UI	Interface de Usuário (*User Interface*)
RoR	Ruby on Rails
PasS	Plataforma como Serviço (*Platform as Service*)
GA	Google Analytics
SO	Sistema Operacional

Capítulo 1

Introdução

1.1 Motivação e Objetivos

A Cidade Universitária possui uma variedade de eventos sociais e acadêmicos, que ocorrem, por vezes, simultaneamente em toda sua extensão. Esse cenário se dá pela complexidade cultural existente no campus, que envolve discentes, docentes e comunidade.

Junto à comunidade USP, foi realizada uma enquete via e-mail, na qual constatou-se o interesse em uma plataforma para divulgar e centralizar os inúmeros eventos, acadêmicos ou não, da Cidade Universitária. Para atender a essa demanda de interesse, foi proposta a criação do USP Eventos, um sistema web voltado para os usuários se informarem sobre o que ocorre no campus, além de incentivar a organização de eventos de modo a ocupar o espaço público.

Para o desenvolvimento do projeto foi escolhida uma abordagem baseada em conceitos de *Lean Startup* e Métodos Ágeis. A escolha dessa metodologia combinada dá-se pela sua ampla aplicação em projetos com grande grau de incerteza, nos quais um tratamento interativo e um desenvolvimento incremental auxiliam no aprendizado sobre os interesses do usuários.

O objetivo do trabalho foi aplicar os conceitos de *Lean Startup* e Métodos Ágeis em um projeto prático a fim de desenvolver um software consistente que atendesse às necessidades dos usuários, além de possibilitar observar as vantagens e desvantagens das abordagens escolhidas.

Os dois primeiros capítulos dessa monografia tem como objetivo contextualizar os conceitos de *Lean Startup* e Métodos Ágeis. Em sequência o capítulo Tecnologias descreve brevemente os programas, serviços, linguagens e arcabouços utilizados durante o desenvolvimento. O capítulo USP Eventos mostra com detalhes os ciclos de desenvolvimento do projeto. Em Conclusões é feita uma análise dos resultados obtidos e, finalizando, a Parte Subjetiva na qual são feitos comentários sobre os desafios, disciplinas relevantes e próximos passos.

Capítulo 2

Lean Startup

2.1 Lean Startup: O que é

2.1.1 Startup: uma definição

Por meio da popularização da Internet e dos computadores pessoais nos anos seguintes de 1990, 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 "ponto com", seja pelo seu novo modo de organização e processo de produção.

Paternoster (2014) provê uma definição de uma *startup* de software baseada nos desafios que ela enfrenta:

- Pouco ou nenhum histórico operacional: *startups* possuem pouca ou nenhuma experiência em desenvolver processos de negócio e gerenciamento organizacional;
- Recursos limitados: *startups* tipicamente focam em lançar um único 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, nas startups elas tendem a ser inconsistentes;
- Mercado e tecnologias dinâmicas: empresas novas de softwares frequentemente precisam desenvolver ou operar com tecnologias disruptivas para atuar em potenciais mercados alvos.

Com o passar dos anos e com o impacto da internet no mercado global, o termo *startup* amadureceu para englobar 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 (Blank, 2003).

2.1.2 Lean Startup

Lean Startup (*Startup* Enxuta) é um conceito introduzido por Eric Ries, empreendedor de diversas *startups* do Vale do Silício. Trata-se de uma metodologia de projeto derivada da combinação de outros padrões de negócios como Produto Mínimo Viável (*Minimal Viable Product*), Desenvolvimento de Clientes (*Customer Development*) e Desenvolvimento Ágil de Software ou Método Ágil (*Agile Software Development*).

Ries propõe que é possível encurtar os ciclos de implementação de um produto (ou solução) adotando uma combinação de testes, hipóteses de negócio e experimentações em conjunto com o público-alvo. Por meio do lançamento periódico é possível avaliar não apenas quesitos técnicos como também a reação do mercado. Consequentemente o retorno de cada interação afeta o planejamento do produto e suas futuras versões (Ries, 2011).

Esse desenvolvimento cíclico é chamado de ciclo de Construir-Medir-Aprender e baseia-se não só em construir uma versão atualizada do produto como também em obter um aprendizado válido por meio de experimentos que permitam comprovar a aceitação ou não das hipóteses de negócio. Proposta em 1996 por Frank Robinson, CEO da empresa SyncDev[1], trata-se da produção de versões simples do produto em múltiplos ciclos de avaliação, estratégia derivada do padrão de Produto Mínimo Viável, popularizado anos depois por Steve Blank (Junk, 2000).

Robinson propõe o lançamento de uma versão, o mais simples possível do produto, de modo a antecipar a análise de mercado e assim minimizar o risco de retorno por parte da empresa. A inovação de Steve Blank foi adaptar essa estratégia, incluindo o lado do cliente, o que ele chama de Desenvolvimento do Cliente (*Customer Development*). Blank vai além de apenas minimizar o risco de retorno, busca compreender as necessidades do cliente.

O *Lean Startup* aprimora ainda mais o conceito de avaliações sob cada interação com o intuito de maximizar o aprendizado e alinhar a evolução do projeto com o desenvolvimento do cliente.

2.2 Produto Mínimo Viável (MVP)

O conceito de Produto Mínimo Viável (MVP) é baseado em construir uma versão do produto de modo a maximizar a validação de aprendizado utilizando o menor esforço possível. O tempo de validação do produto é um fator decisivo para o seu sucesso: no cumprimento da demanda do mercado e no uso otimizado de recursos da empresa. (Ries, 2011)

Um MVP deve possuir 3 características [2]:

1. Ter valor suficiente para que uma pessoa queira utilizá-lo ou comprá-lo;
2. Possuir suficientes benefícios para reter os chamados usuários pioneiros (*early adopters*); [3]
3. Ser capaz de prover um ciclo de *feedback* suficiente para guiar o desenvolvimento.

Durante a concepção do projeto são definidas algumas hipóteses sobre o produto. Na etapa do MVP são definidas quais funcionalidades ou estratégias deseja-se testar, de modo que possam validar as hipóteses iniciais e obter o máximo de aprendizado possível.

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 despender horas a fio no seu desenvolvimento. Assim, 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 "produto mínimo viável" e "máximo aprendizado", respectivamente, frequentemente se mostram vagos na documentação do que é um

[1] Retirado de SyncDev: http://www.syncdev.com/minimum-viable-product/ Acesso em: 19 ago. 2016.

[2] Retirado de Techopedia: Minimum Viable Product (MVP): https://www.techopedia.com/definition/27809/minimum-viable-product-mvp Acesso em: 19 ago. 2016.

[3] Os primeiros consumidores de um produto que acaba de tornar-se disponível

MVP. Como o próprio Eric Ries esclarece, a aplicação desses termos não é rígida e variam de acordo com o contexto e julgamento de quem o estiver aplicando. [4].

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. Desta forma um MVP pode ser apenas um protótipo ou mesmo apenas um *mockup* do que será oferecido na versão completa.

Alguns tipos de MVP são: [5]

- Vídeo explicativo: um vídeo curto contendo uma explicação clara do que o produto faz e porque as pessoas deveriam utilizá-lo. Esse é o caso do *Dropbox* que fez um vídeo[6] 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 será oferecido, assim como um formulário de contato. Por meio do Google Analytics é possível manter um registro de conversões (no caso cadastros do formulário) a fim de medir o interesse das pessoas no produto;

- *MVP "Mago de OZ":* A ideia 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":* Em vez de prover um produto, realiza-se manualmente o serviço, executando exatamente os mesmos passos para o usuário que a empresa realizaria. É um método não escalável e lento para executar, pois requer que se esteja em contato direto com o cliente e realize as tarefas manualmente. No entanto, isso permite rápido aprendizado tanto sobre o produto como sobre e o cliente.

  Esse foi o caso da empresa *Food on the Table* que 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 procuravam por descontos nos supermercados em que ela fazia compras.

2.3 O ciclo de Construir-Medir-Aprender (Build-Measure-Learn)

Com o surgimento das Metodologia Ágeis foi possível criar softwares de maneira interativa e envolver o cliente no processo. Porém, devido à falta de um arcabouço para testar as hipóteses comerciais, acabava-se muitas vezes por desenvolver um software com todas as funcionalidades que o cliente gostaria mas sem obter um sucesso comercial.[7]

O modelo de Construir-Medir-Aprender surge então com o principal objetivo de eliminar as incertezas sobre as hipóteses do produto, alinhando-o com as expectativas dos usuários.

[4]Fonte: *Startup Lessons Learned, Minimum Viable Product: a guide* http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html

[5] Fonte: Scale my Business: The Ultimate Guide to Minimum Viable Products http://scalemybusiness.com/the-ultimate-guide-to-minimum-viable-products/ Acesso em: 19 ago. 2016.

[6]Link para o vídeo https://www.youtube.com/watch?v=7QmCUDHpNzE Acesso em: 3 set. 2016.

[7]Fonte: 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/ Acesso em: 19 ago. 2016.

Por meio do aprendizado rápido sobre o comportamento dos usuários é possível minimizar os riscos e custos de funcionalidades desnecessárias, mantendo o aspecto interativo presente na Metodologia Ágil e obtendo um aprendizado sobre o comportamento do usuário a cada interação.

Este modelo consiste em um ciclo de 3 fases (figura 2.1):

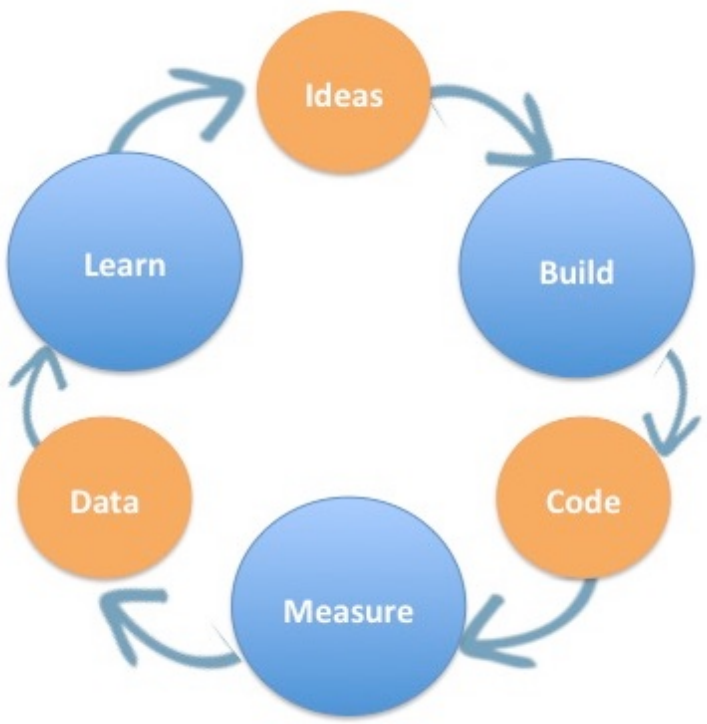


Figura 2.1: *O ciclo de Build Measure Learn.*
Retirada de: https://steveblank.files.wordpress.com/2015/05/ideas-build-code-measure.jpg, acesso em: 20 jul. 2016.

- Construir (*Build*): algumas ideias são definidas a partir das hipóteses do produto que precisam ser implementadas no MVP;
- Medir (*Measure*): implementado o MVP coleta-se os dados de uso, avaliando seu desempenho. Todo o ciclo é baseado na ideia de aprendizado válido para coletar o máximo possível de informação sobre a reação dos usuários.
- Aprender (*Learn*): a partir da análise dos dados coletados é possível inferir sobre como prosseguir o desenvolvimento e definir novas hipóteses para iniciar um novo ciclo.

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. (Ries, 2011)

As alterações de software precisam ser feitas de maneira rápida, de modo a testar o mais rápido possível novas ideias. Portanto, é importante que as funcionalidades sejam simples e diretas. O foco é o aprendizado e não desenvolver um software ou um protótipo completo.

Para minimizar o risco de que um sistema com problemas seja colocado em produção e acelerar o processo de desenvolvimento, procura-se utilizar ferramentas que auxiliem na integração contínua do software, além da execução de testes automatizados. A utilização desses recursos permite 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 iniciativas dentro de empresas já existentes, não começam com ideias mas com hipóteses.

O conceito de "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. (Blank, 2015)

Como a construção deve estar alinhada com as hipóteses formuladas e a cada ciclo é necessário sempre testar novas hipóteses, a figura 2.2 representa uma variação do ciclo de Construir-Medir-Aprender, cuja proposta é enfatizar quais hipóteses devem ser testadas.

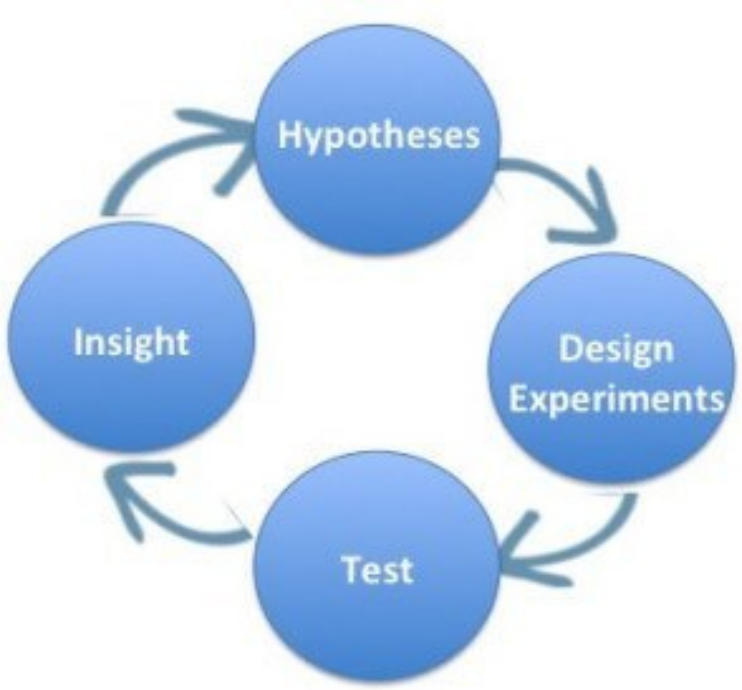


Figura 2.2: *Uma variação para o ciclo de Construir-Medir-Aprender.*
Retirada de: https://steveblank.files.wordpress.com/2015/05/hypotheses-experiment.jpg, acesso em: 20 jul. 2016.

2.4 Desenvolvimento de Clientes

Steve Blank, em seu livro "Os 4 passos para a epifania", explica que o modelo de Desenvolvimento de Clientes não é um substituto para o modelo de Desenvolvimento de Produto e que na verdade ambos devem ser executados em paralelo. No mesmo livro ele define o modelo de Desenvolvimento de Clientes de uma *startup* com a premissa: "Aprender e descobrir quem são os clientes iniciais de uma empresa, e em quais mercados eles estão, requer um processo separado e distinto do Desenvolvimento de Produtos. A soma dessas atividades é o Desenvolvimento de Clientes". (Blank, 2003)

É um processo interativo que parte da premissa de que os fatos estão fora do escritório. Dentro dele só existem opiniões e, portanto, o empreendedor deve buscar o quanto antes validar suas hipóteses fundamentais no mercado.[8].

[8]Manual da *Startup* Fonte: http://www.manualdastartup.com.br/blog/customer-development-o-processo-para-se-chegar-ao-productmarket-fit/ Acesso em: 19 ago. 2016.

O processo é dividido em 4 passos (figura 2.3), sendo que os dois primeiros acontecem antes do ajuste do produto ao mercado, com foco no aprendizado e validação de hipóteses, enquanto os outros dois têm foco no crescimento e otimizações.

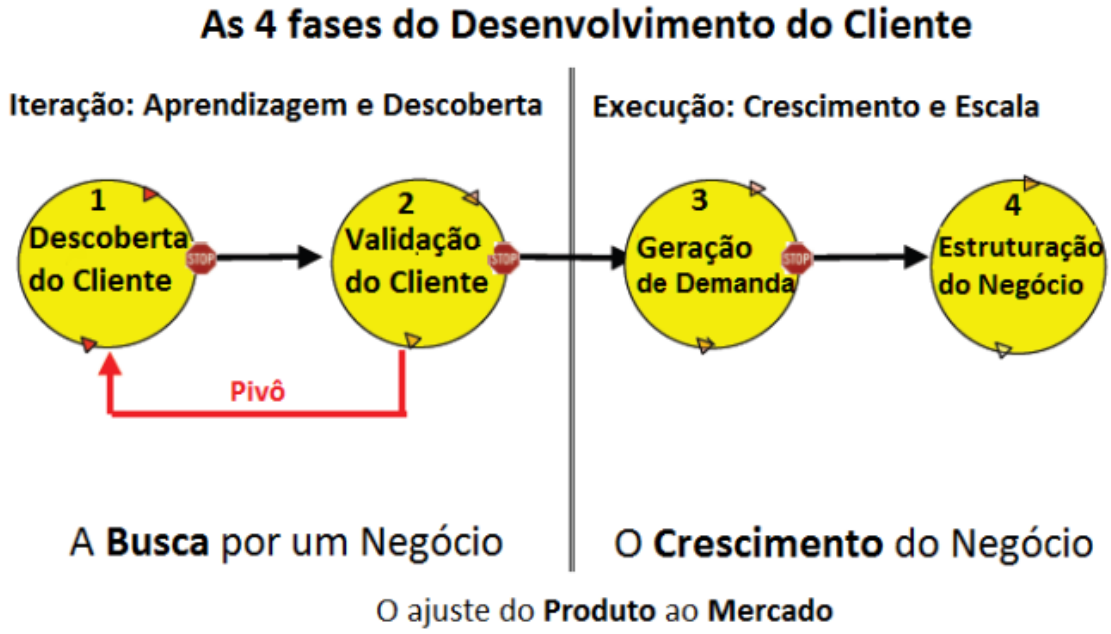


Figura 2.3: *Os 4 passos do ciclo de Customer Development.*
Retirada de: http://www.manualdastartup.com.br/blog/customer-development-o-processo-para-se-chegar-ao-productmarket-fit, acesso em: 14 set. 2016.

A primeira fase do ciclo é compreendida pelos dois primeiros passos: Descoberta do Cliente e Validação do Cliente. Na Descoberta do Cliente o objetivo é provar que existe um problema a ser solucionado em um mercado grande o suficiente e que o produto supre essa necessidade. Para isso é proposto sempre estar em contato com o cliente, realizando entrevistas de modo a obter um conjunto mínimo de funcionalidades e testá-las em um MVP.

Já na etapa de Validação do Cliente o foco é provar que existe uma maneira rentável de se adquirir e manter consumidores. Nessa etapa é necessário descobrir se de fato existem clientes dispostos a pagar pelo produto e se o produto provoca uma mudança na rotina do usuário.

A segunda fase, Geração de Demanda e Estruturação do Negócio, é a fase para crescimento do negócio, na qual o foco passa a ser a execução. Nessa fase o foco deixa de ser achar o encaixe de mercado para concentrar-se em escalar o crescimento da empresa.

2.5 Conceitos Utilizados

Para o desenvolvimento do USP Eventos foram realizados 3 ciclos de Construir-Medir-Aprender. Em cada ciclo foi construído um MVP para testar as hipóteses levantadas com o intuito de medir o *feedback* dos usuários e planejar o ciclo seguinte.

Em paralelo foram utilizados conceitos de Desenvolvimento de Clientes com o objetivo de entender e descobrir quem era o público-alvo. O contato com os clientes deu-se por meio de formulários de sugestões, questionários e conversas face-a-face com os usuários da plataforma.

Capítulo 3

Métodos Ágeis

3.1 Origem

Na década de 1990 com a decadência dos métodos clássicos de desenvolvimento de software - tal 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 obter maior interatividade entre o cliente e a produção, além de flexibilizar as etapas de planejamento e a validação do sistema.

Dentre esses modelos, os mais conhecidos e citados até hoje são: 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 coletivamente referidos como Métodos Ágeis, já que influenciaram diretamente na conceituação do manifesto.

Em fevereiro de 2001, dezessete desenvolvedores de software se reuniram no resort Snowbird em Utah (EUA), para discutir métodos leves de desenvolvimento. Na ocasião publicaram o Manifesto para Desenvolvimento Ágil de Software, documento que reúne os princípios e práticas deste conjunto de metodologias. Mais tarde, algumas pessoas formaram a *Agile Alliance*, uma organização sem fins lucrativos que promove o Desenvolvimento Ágil[1].

3.2 Definição

O Manifesto Ágil é baseado em doze princípios[2]:

1. A satisfação do cliente é pela entrega antecipada e contínua do software funcional;
2. Boas mundanças são sempre bem-vindas, mesmo que em desenvolvimento tardio;
3. Novas funcionalidades devem ser entregues frequentemente (semanas em vez de meses);
4. Uma cooperação estreita e diária entre cliente e desenvolvedor;
5. Os projetos são construídos em torno de indivíduos motivados, entre os quais existe relação de confiança;

[1]Retirado de: http://agilemanifesto.org/history.html Acesso em: 12 set. 2016.
[2]Retirado de: http://agilemanifesto.org/principles.html Acesso em: 12 set. 2016.

6. A maneira mais eficiente e efetiva de transmitir informações é por conversas face-a-face;
7. Softwares funcionais são a principal medida de progresso;
8. Desenvolvimento sustentável, ou seja, produção em um ritmo constante;
9. Atenção contínua à excelência técnica e bom design;
10. Simplicidade em maximizar a quantidade de trabalho não feito;
11. Melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas;
12. Regularmente, a equipe se reúne e reflete sobre como tornar-se mais eficaz, então se ajusta para o que foi acordado.

Comparado à engenharia de software tradicional, o Desenvolvimento Ágil de software tem como alvo principalmente sistemas complexos com características dinâmicas, ou seja, não-determinístico e não-linear, sendo que as estimativas precisas, os planos estáveis, e as previsões são difíceis de se obter em estágios iniciais como na própria elaboração do projeto em si do produto. Isso resulta na necessidade de desenvolvimento e de planejamento evolutivo. (Larman, 2004)

Existem muitos modelos específicos de Métodos Ágeis. A maioria busca promover o trabalho em equipe, a colaboração e a adaptabilidade no processo em todo o ciclo de vida de desenvolvimento. Utiliza-se uma maior quantidade de ciclos em curtos períodos de tempo a fim de minimizar e parcelar o planejamento e estimativas. Desse modo, cada iteração envolve uma equipe multifuncional de trabalho para cada função necessária: planejamento, análise, projeto, desenvolvimento, testes de unidade e testes de aceitação. No final da iteração é demonstrado para as partes interessadas, como o cliente, o que foi desenvolvido no processo. Por fim, isto minimiza o risco global de erros e permite que o produto se adapte às mudanças rapidamente.

Independente do método ágil seguido, cada equipe deve incluir um representante do cliente no processo. No caso do Scrum, essa é a função do "proprietário do produto", por exemplo. Por compromisso, este colaborador se coloca disponível para os desenvolvedores, respondendo questões durante cada iteração. A inclusão deste representante 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 ao 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, os testes de unidade automatizado, a programação em pares, o desenvolvimento orientado a testes, os padrões de projeto, o *Domain-Driven Design*, a refatoração de código entre outras, 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 dos arcabouços mais populares hoje de Desenvolvimento Ágil de Software. Um princípio chave do Scrum é o 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, pois desafios imprevistos normalmente não podem ser tratados de forma preventiva ou planejados tradicionalmente.

O arcabouço 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, é a 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, mas não aponta necessariamente como a equipe deve chegar 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.

  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.

  Ele é o responsável por garantir que o Scrum seja seguido, ajudando a equipe a seguir os processos acordados e incentivando 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* (figura 3.1) é a unidade básica do desenvolvimento em Scrum. É restrito a uma duração específica fixada antecipadamente, 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 a serem tratados ou implementados nesse ciclo (*Sprint Backlog*). Por fim, termina com uma revisão (*Sprint Review*), que analisa os progressos realizados para mostrar aos interessados, além disso é debatido as lições e melhorias a serem aplicadas para as próximas interações (*Sprint Retrospective*).

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 testes são refeitos a cada interação. Este processo também é chamado de ciclo *Red-Green-Refactor* (figura 3.2).

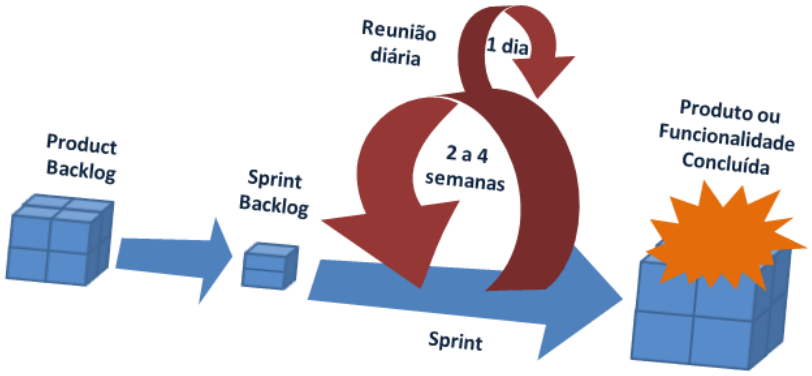


Figura 3.1: *O ciclo interativo Sprint.*
Retirada de: http://www.mindmaster.com.br/scrum/, acesso em: 12 set. 2016.

Cada novo recurso a ser implementado começa com o desenvolvimento de um teste que define uma função ou melhorias de uma função, o mais preciso e sucinto possível. A nova funcionalidade agora é implementada com objetivo em passar neste novo teste. Esta é principal característica do TDD, que o diferencia dos demais métodos em que os testes são criados após a implementação da nova função.

Ao passar no novo teste todo código é então também submetido aos testes anteriores, se passar em todos, o programador possui uma maior garantia da integralidade do novo código, ou seja, atende aos requisitos de teste e não degrada quaisquer outros recursos existentes. Se não passar em um ou mais dos testes anteriores o novo código então deve ser ajustado até que passe em todos os testes. Essa etapa é chamada de refatoração.

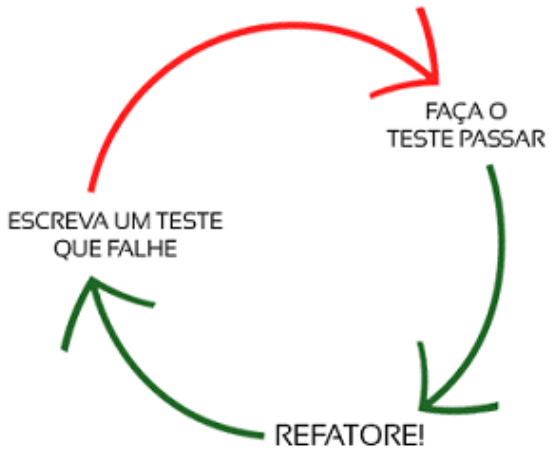


Figura 3.2: *O ciclo* Red-Green-Refactor.
Retirada de: http://tdd.caelum.com.br/, acesso em: 12 set. 2016.

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 algum 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 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 gerencia não apenas o código do produto, mas 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 à base por meio 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 à 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 (figura 3.3) para indicar o andamento dos fluxos de produção em empresas de fabricação em série, hoje popularmente utilizado em Métodos Ágeis. Os cartões são organizados sob 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.

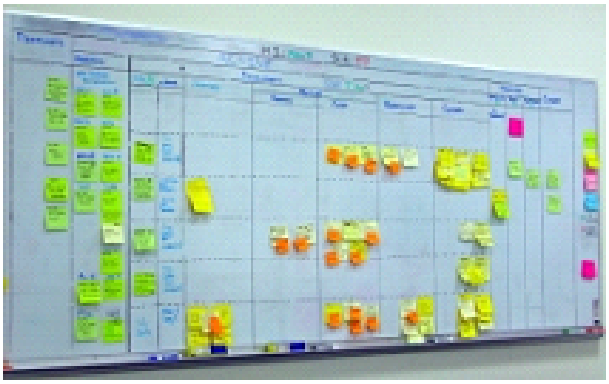


Figura 3.3: *Exemplo de um quadro de* Kanban.
Retirada de: https://www.significados.com.br/kanban/, acesso em: 12 set. 2016.

3.4 Um contraponto: O modelo Cascata

O modelo Cascata, algumas vezes chamado de ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com o levantamento de necessidades por parte do cliente, avançando para as fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do software concluído (figura 3.4). (Pressman, 2011)

Segundo Pressman (2011), o modelo em cascata foi o primeiro paradigma de desenvolvimento criado pela Engenharia de Software, que teve sua essência retirada de outras áreas da Engenharia.

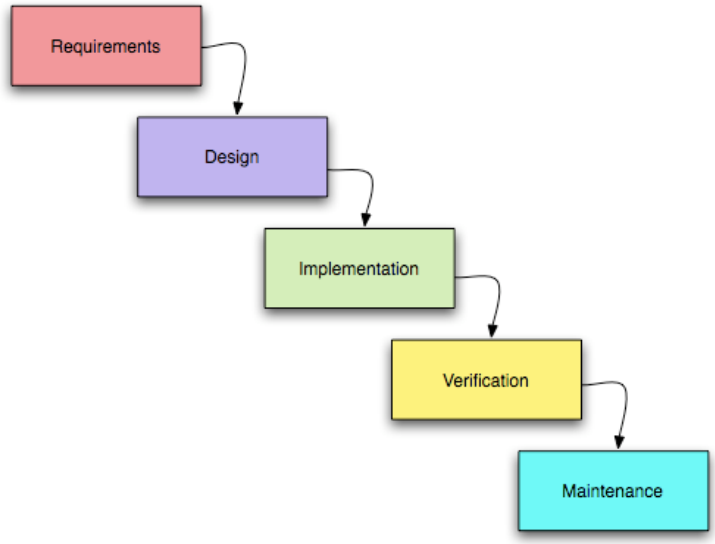


Figura 3.4: *O Modelo Cascata.*
Retirada de: http://modelocascata.blogspot.com.br/, acesso em: 20 jul. 2016.

As fases do Modelo são:

- Requerimentos: realizar a análise de requisitos do projeto.
- Design de Projeto: focando na estrutura de dados, arquitetura do software, detalhes procedurais 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 produto 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 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. (Pressman, 2011)

Mudanças tardias no escopo do projeto encarecem o custo total e poderiam ter sido evitadas e contornadas de maneira mais satisfatória em um modelo com um processo de desenvolvimento iterativo [3].

O Desenvolvimento Ágil tem pouco em comum com o Modelo em Cascata. Na visão de alguns este modelo é desacreditado, apesar de ser um modelo de uso comum. O modelo em Cascata é uma das metodologias com maior ênfase no planejamento, seguindo seus passos a partir da captura dos requisitos, análise, projeto, codificação e testes em uma sequência pré-planejada e restrita. [4]

3.5 Conceitos Utilizados

A organização do projeto foi feita utilizando uma abordagem com Kanban, separando as tarefas em 4 colunas: BACKLOG, TO DO, DOING, DONE.

Foram construídos testes de unidade para as funcionalidades e também testes de aceitação com o intuito de testar o sistema como um todo, simulando o seu funcionamento real. As bibliotecas Rspec e Capybara, desenvolvidas para o arcabouço Rails, foram utilizadas para construção dos testes. A biblioteca SimpleCov (figura 3.5) oferece uma interface gráfica para visualizar as partes do código testadas e a porcentagem de cobertura total.

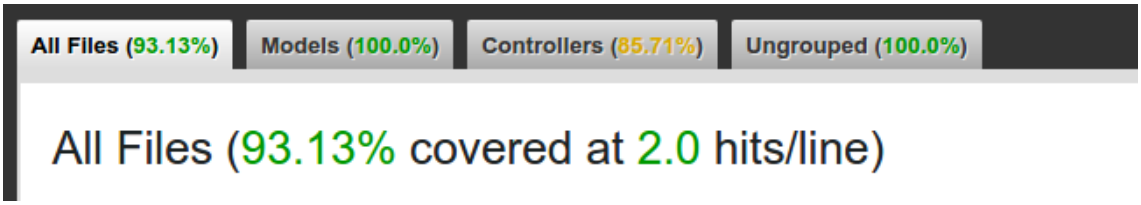


Figura 3.5: *93.13% do Sistema com cobertura de testes*

Em conjunto com os testes foi implementado um processo de integração contínua na qual, a cada commit realizado, o sistema passava por toda a bateria de testes e, no caso de sucesso (figura 3.6), era atualizado automaticamente.

```
456 /home/travis/.rvm/rubies/ruby-2.2.3/bin/ruby -I/home/travis/build/caioquinta/usp_eventos/vendor/bundle/ruby/2.2.0/gems/rspec-
    core-3.4.4/lib:/home/travis/build/caioquinta/usp_eventos/vendor/bundle/ruby/2.2.0/gems/rspec-support-3.4.1/lib
    /home/travis/build/caioquinta/usp_eventos/vendor/bundle/ruby/2.2.0/gems/rspec-core-3.4.4/exe/rspec --pattern spec/\*\*\
    {,/\*/\*\*\}/\*_spec.rb
457 .........................
458
459 Finished in 12.83 seconds (files took 2.39 seconds to load)
460 25 examples, 0 failures
461
462 Coverage report generated for RSpec to /home/travis/build/caioquinta/usp_eventos/coverage. 149 / 160 LOC (93.13%) covered.
```

Figura 3.6: *Travis CI com todos os testes aprovados*

[3] Fonte: Waterfall Model https://en.wikipedia.org/wiki/Waterfall_model Acesso em: 12 set. 2016.

[4] Fonte: Wikipedia - Desenvolvimento Ágil de Software: https://pt.wikipedia.org/wiki/Desenvolvimento_%C3%A1gil_de_software#Compara.C3.A7.C3.A3o_com_o_modelo_em_cascata Acesso em: 12 set. 2016.

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. Inicialmente o objetivo era equilibrar programação funcional, imperativa e orientação a objetos. (Documentation, 2016)

- Flexibilidade

  A Linguagem Ruby cresceu devido à sua grande flexibilidade. Sendo possível alterar, remover ou acrescentar partes da linguagem a vontade.

  Como no 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 os operadores matemáticos são considerados açúcares sintáticos nesta linguagem.
  Exemplo:

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

- Closures

  Nesta linguagem, 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:

[1] Fonte: Site Point https://www.sitepoint.com/closures-ruby/ Acesso em: 29 ago. 2016.

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

- Módulos:

  Módulos são formas de agrupar métodos, classes e constantes prevenindo conflitos de nomes e permitindo a fácil implementação de Mixins.

  Diferente de outras linguagens orientadas a objetos, Ruby permite apenas herança simples porém isso é contornado através dos Mixins que permitem a uma classe receber mais de um módulo diferente, herdando assim todos seus métodos e definições.
  Exemplo:

```
1 class MyArray
2           include Enumerable
3     end
```

4.1.2 Rails

Ruby on Rails é um arcabouço escrito em linguagem Ruby, implementado seguindo o padrão MVC[2] totalmente *server-side*, sendo considerado portanto um arcabouço *back-end*.

Este arcabouço oferece também uma estrutura para banco de dados *web service* e *web pages*, além de encorajar padrões de engenharia de software já consagrados tais como[3]

- *Convention over Configuration* (CoC)

  Convenções de configuração visando padronizar o código. Ao adicionar convenções é retirada do desenvolvedor a decisão de como usar o arcabouço, porém isso não diminui sua flexibilidade.

  Um exemplo prático: ao se criar um objeto chamado "User", então sua tabela por convenção se chamará "users" e o correspondente *controller* será UsersController (no plural) pois esse é padrão definido pelo arcabouço.

  Vale ressaltar que é possível alterar essas convenções para adaptar-se às necessidades do desenvolvedor.

- *Don't Repeat yourself* (DRY)

  É definido como "Todo pedaço de informação deve ter uma única, não ambígua, representação autorizada com o Sistema" [4].

  Isso significa que uma modificação em uma parte do sistema não deve modificar outra parte não relacionada assim como elementos que são logicamente relacionados quando modificados ocorrem de forma previsível e uniforme.

[2]Modelo-Visão-Controlador: Na qual o Modelo é a camada que contém os dados e lógica da aplicação, a Visão é a camada de entrada e saída de dados e o Controlador faz a conexão entre ambas camadas, fonte: https://pt.wikipedia.org/wiki/MVC Acesso em: 29 ago. 2016.

[3]Fonte: Ruby on Rails https://en.wikipedia.org/wiki/Ruby_on_Rails Acesso em: 29 ago. 2016.

[4]Fonte: Wikipedia https://en.wikipedia.org/wiki/Don%27t_repeat_yourself Acesso em: 29 ago. 2016.

- *Active Record Pattern*

  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 e então uma instância de objeto estará associada a uma única entrada na respectiva tabela.

  Em Ruby on Rails a biblioteca ActiveRecord implementa o padrão ORM além de acrescentar 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 arcabouço 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.

As bibliotecas criadas pela comunidade para serem usadas em conjunto com o Rails são distribuídas em código aberto e chamadas de Ruby Gems ou apenas Gems[5].

4.1.3 Porque escolher Ruby on Rails

O uso do *Ruby on Rails* entre *startups* tem crescido nos últimos anos devido a alguns fatores intrinsecamente ligados à própria estrutura do arcabouço Rails e também à necessidade que *startups* e o próprio modelo de *Lean Startup* exigem para iterações e alterações de códigos rápidas. (Udovychenko, 2016)

Um dos principais pontos do Ruby on Rails, o CoC, 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. (Morrice, 2015)

Dessa forma, o desenvolvedor possui mais tempo para concentrar-se em decisões sobre o produto. Essa agilidade em codificar rapidamente resulta em interações mais rápidas, contribuindo para maior agilidade dentro do ciclo de Construir-Medir-Aprender.

Outro fator a favor do arcabouço Rails é o grande enfoque em testes automatizados. (Morrice, 2015)

Para o desenvolvimento da plataforma USP Eventos foram utilizadas as gems Rspec e Capybara que permitem não só a realização de testes de unidade como também testes de aceitação, de modo muito rápido e direto, garantindo assim a confiabilidade do projeto. 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 a cada *commit* realizado 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 desenvolvedor interessado sempre tenha em mãos materiais e ferramentas de qualidade para iniciar o desenvolvimento do seu projeto. (Udovychenko, 2016)

[5]Fonte: Wikipedia https://en.wikipedia.org/wiki/RubyGems Acesso em: 29 ago. 2016.

Acrescenta-se também a favor do arcabouço Rails sua escalabilidade. Rails é utilizado por empresas de grande porte tais como Groupon, Twitter, Basecamp, mostrando-se um arcabouço robusto capaz de lidar com grandes sistemas sem ter queda de desempenho. (Udovychenko, 2016)

Rails é também um arcabouço seguro garantindo proteção contra SQL-Injections e XSS (*Cross Site Scripting*).

Além disso, os programadores que contribuem para o arcabouço devem seguir o *Secure Life Cycle Development* (figura 4.1) proposto pela Microsoft, um modelo de desenvolvimento de software cujo principal objetivo é ajudar a construir softwares mais seguros e confiáveis e reduzir custos [6].

Figura 4.1: *Fases do Secure Life Cycle Development.*

4.2 Heroku

Heroku é uma PaaS implementada utilizando *cloud computing* criada em 2007 e utilizada como um modelo de *Deployment* para Aplicações Web[7].

A aplicação é enviada para o Heroku por meio de uma conexão direta via GitHub, Dropbox ou alguma outra API que permite ao Heroku executar os aplicativos em containers virtuais.

Enviado o código-fonte, este 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, chamado de Dyno da aplicação. Tal container é pré-carregado com algumas configurações da aplicação, tais como 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. Dessa forma, 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.

4.3 Travis CI

Travis CI é um serviço de integração contínua usado para testar projetos hospedados no Github. Toda vez que um commit é feito para o repositório selecionado no Github, o Travis executa 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 4.2).

[6] Fonte: Wikipedia https://en.wikipedia.org/wiki/Microsoft_Security_Development_Lifecycle Acesso em: 22 out. 2016.

[7] Fonte: Heroku https://en.wikipedia.org/wiki/Heroku Acesso em: 29 ago. 2016.

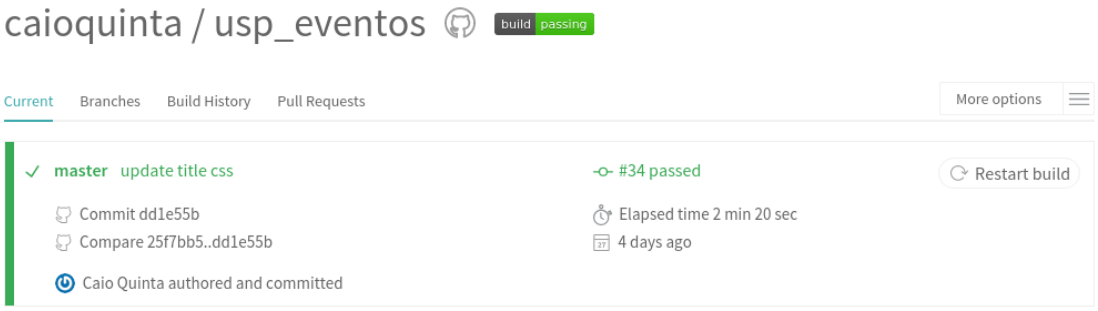


Figura 4.2: *O repositório USP Eventos no Travis CI.*

4.4 Trello

Trello[8] é um gerenciador de projetos online desenvolvido pela Fog Creek Software lançado em 2011. Possui uma interface amigável na qual é possível criar tarefas e colunas conforme as preferências do usuário, sendo bastante utilizado em conjunto com uma abordagem *kanban* para gerenciamento.

A ferramenta permite que em cada tarefa (figura 4.3) sejam adicionadas sua descrição, arquivos relevantes, prazo de término, uma etiqueta personalizável para identificação e também uma opção para os membros conversarem sobre o andamento da mesma.

Figura 4.3: *Uma tarefa definida no Trello.*

4.5 Github

Github é um serviço que disponibiliza repositórios git baseado na *web* lançado em 2008. O serviço de controle de versão é implementado pelo git enquanto o Github implementa outras funcionalidades próprias, como gerenciamento de tarefas, wikis próprias e *bug tracking*.

É possível integrar o seu repositório no Github com outros serviços. No caso do USP Eventos o repositório no Github foi integrado com o Travis CI e também com o próprio

[8] Disponível em: www.trello.com

Heroku (figura 4.4).

Dessa forma, caso um *commit* para a *branch* "Produção" fosse aprovado pelo Travis CI, então o Heroku automaticamente o colocava em produção.

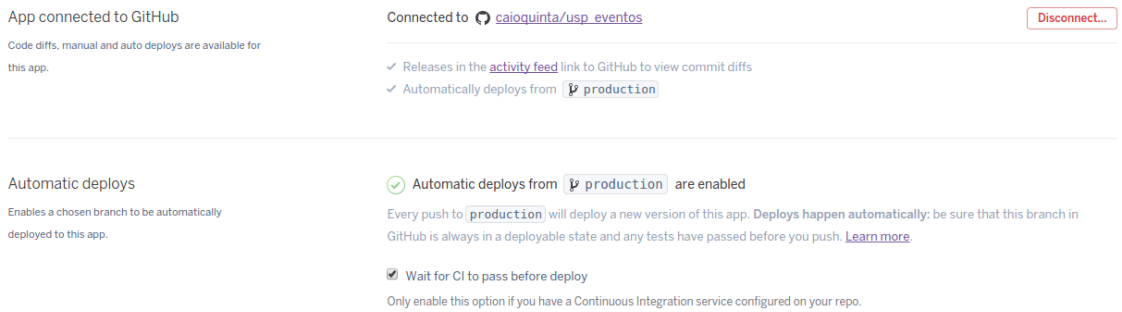


Figura 4.4: *Tela de Administração do Heroku para integração com o Github e* deploy *automático.*

4.6 Google Analytics e Google Tag Manager

O Google Analytics é uma plataforma de análise de dados oferecida pelo Google que permite por meio dos relatórios gerados pela plataforma obter uma série de informações quanto ao tipo de usuário que visualiza a página, o fluxo do site e a origem do acesso.

Com o uso de *tags* é possível criar eventos que são personalizados e disparados de acordo com a navegação do usuário dentro do site. Tais *tags* podem ser implementadas diretamente com um pequeno código em javascript para integração com o Google Analytics ou utilizando o Google Tag Manager.

O Google Tag Manager é uma plataforma intermediária que provê acesso e configuração de tags personalizadas para obtenção de dados pelo Google Analytics sem que seja necessário modificar diretamente o código-fonte do sistema. A opção de utilizar o Google Tag Manager no projeto deu-se principalmente pela facilidade de criar-se novas tags e alterações além de garantir uma maior organização das informações.

Dentro do projeto foi utilizado as informações obtidas pelo Google Analytics para validação de aprendizado entre as iterações.

4.7 Painel de opiniões Populares - POP

Com o intuito de definir o interesse do público alvo por meio de uma enquete colaborativa, foi utilizado o POP como sistema de votação devido à possibilidade dos usuários poderem 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 junto à comunidades para auxiliar na tomada de decisões e encaminhamento de opiniões para as autoridades responsáveis.

Foi permitida a utilização da plataforma implementada em uma instância separada com o nome de POP-TCC realizando apenas uma pequena modificação no sistema POP original.

No POP-TCC os usuários só poderiam votar de maneira positiva nas opções, ao contrário do sistema original, que permitia votos negativos e até ocultamento dos itens que obtivessem um grande número de negativações pelos usuários.

4.8 HeatMap

O serviço fornecido pela plataforma Heatmap.me consiste em prover uma API que é capaz de capturar os cliques em uma determinada página e mostrá-los na forma de uma mapa de calor.

Um mapa de calor é uma representação gráfica dos cliques em uma página na qual conforme uma determinada região for recebendo mais cliques sua cor é alterada proporcionalmente (figura 4.5).

Figura 4.5: *Um exemplo de uma página utilizando o HeatMap.*
Retirada de: https://heatmap.me/, acesso em: 02 out. 2016.

As cores inicialmente começam em um tom verde quando clicadas poucas vezes, sendo gradativamente alteradas para cores mais "quentes" tais como laranja ou vermelho conforme cliques na mesma região são feitos.

4.9 Typeform

A empresa Typeform oferece um serviço de formulários online para execução de pesquisas simples ou complexas.

A ferramenta é adequada para entrevistas de satisfação e opinião do cliente, oferecendo uma interface gráfica bastante amigável, além de templates configuráveis para o tipo de pesquisa que o usuário deseja realizar.

Após a execução da pesquisa é possível exportar os resultados em planilhas ou integrar com o seu banco de dados caso desejar.

Capítulo 5

Usp Eventos

5.1 Definição do Projeto

5.1.1 Motivação

A ideia 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 à 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 a 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.

O sistema POP (Painel de Opinião Pública) foi escolhido para efetuar a enquete, pois permitia uma criação dinâmica de opções pelos usuários. Foi então criada uma instância independente do sistema adaptada chamada POP-TCC (figura 5.1) utilizando o Heroku que poderia ser acessada pelo endereço pop-tcc.herokuapp.com.

Em 11/01/2016 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 páginas listadas na tabela 5.1.

Ao longo de duas semanas outras opções de projeto surgiram. O resultado final (figura 5.2) da enquete e a descrição das sugestões seguem abaixo:

Tabela 5.1: *Comunidades do Facebook na qual foram feitas divulgações*

Comunidade	Número de Membros
USP - Universidade de São Paulo	9000
FAU USP	4000
IME USP	3000
Universidade de São Paulo	5000
Baladas USP	15000

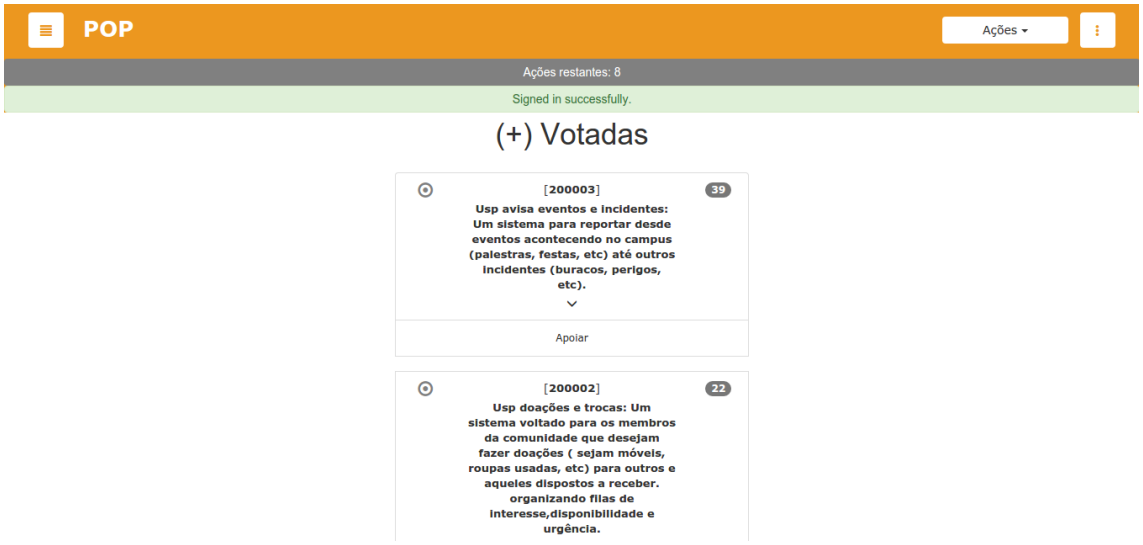


Figura 5.1: *Sistema POP-TCC com a enquete para votação.*

- 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: Desenvolvimento da ideia que já teve adesão, mas morreu por falta de manutenção. Implementação de novos pontos de troca de bicicletas 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 pra reunir o pessoal e tomar uma rodada de suco com a galera;

- 8ºLugar (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.

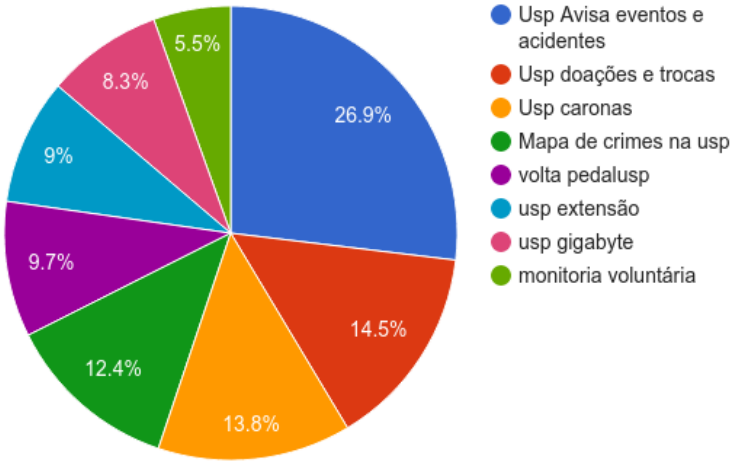


Figura 5.2: *Gráfico gerado pelo resultado da Enquete.*

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

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.

Em paralelo a essa pesquisa, também foi definida a plataforma sobre a qual o projeto seria desenvolvido.

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 os sistemas que não necessariamente fossem voltados para fins acadêmicos. O objetivo principal era obter uma base de conhecimento de quais funcionalidades um sistema de divulgação possui ou encontrar uma solução em código aberto que pudesse servir como base.

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 social da USP para eventos ocorrendo em suas dependências para todos os campus.
  Pontos Fortes: Sendo o canal de comunicação oficial da universidade, mostra-se como uma ótima fonte de conteúdo sobre eventos sociais.
  Pontos 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.
  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 de São Paulo 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 foi levado em consideração fatores técnicos e do público-alvo.

De acordo com o gráfico 5.3 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 móveis, totalizando juntas 99,1% do mercado, o que significaria que ao desenvolver dois aplicativos nativos para ambas as plataformas corresponde a ter a acesso a 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. 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 divergem 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.

[1] Fonte: Statista http://www.statista.com/statistics/254653/mobile-internet-user-penetration-in-brazil/ Acesso em: 22 out. 2016.

[2] Fonte: Caelum http://blog.caelum.com.br/aplicacoes-mobile-web-ou-nativa/ Acesso em: 22 out. 2016.

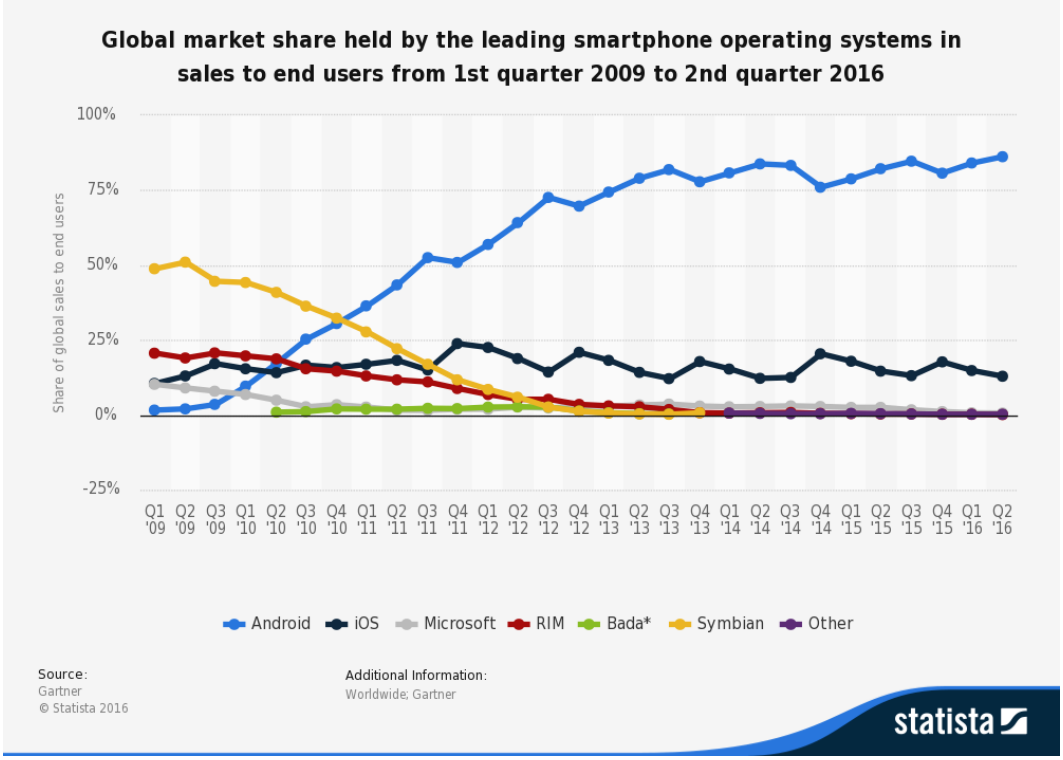


Figura 5.3: *Distribuição de mercado para Sistemas Móveis.*

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

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

A escolha de uma aplicação web deu-se principalmente pela facilidade de atualização, por não necessitar da aprovação de uma loja online, ganhando em agilidade para realizar novos experimentos.

Apesar da escolha de um sistema web ao analisar o crescimento do acesso móvel no Brasil (figura 5.4), que vem aumentando a passos largos no país, houve a preocupação em desenvolver 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.

5.3 Kanban

Kanban é uma palavra japonesa que significa cartão visual. 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). (Filho, 2014)

[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

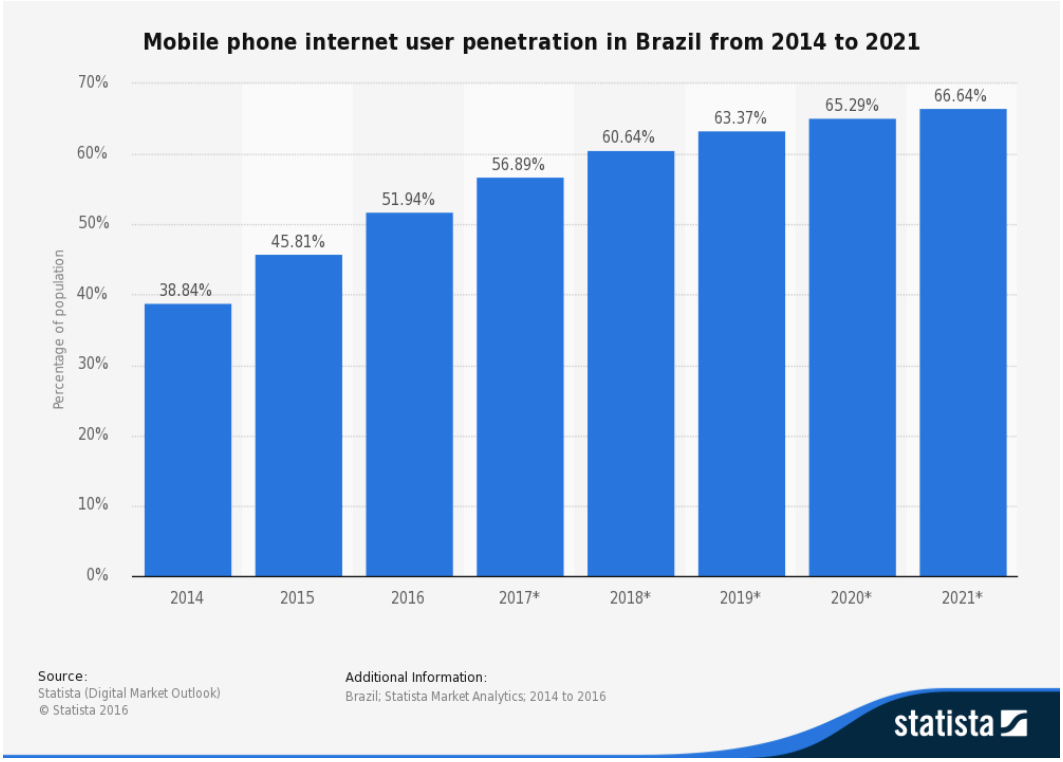


Figura 5.4: *Crescimento do Uso de internet móvel no Brasil.*

No contexto de Métodos Ágeis e *Lean Startup* foi utilizado uma abordagem com 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: (Filho, 2014)

- 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.

Cada item adicionado na fila de TO DO é chamada de tarefa. Uma hipótese a ser testada pode ser transformada em uma série de tarefas pequenas a 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 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 [4], na qual foram colocadas ideias que poderiam ou não ser transformadas em tarefas de desenvolvimento ou hipóteses para serem executadas em uma iteração do ciclo de Construir-Medir-Aprender. Em cada iteração do ciclo foi especificado quais hipóteses seriam testadas e a partir delas criado 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 5.5) foi utilizada para simular um quadro Kanban digital. Com ela foi possível guiar todo o desenvolvimento do sistema.

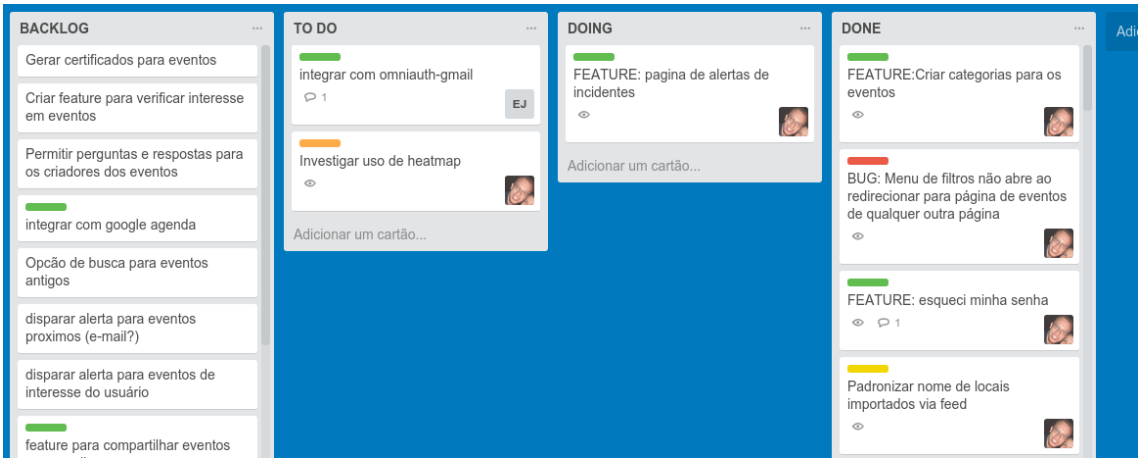


Figura 5.5: *Sistema Trello contendo as 4 colunas utilizadas no USP Eventos.*

Com o intuito de tornar melhor a 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;
- QUICK WIN (amarelo): Melhoria feita rapidamente e não prevista durante a definição de tarefas.

5.4 Primeira Iteração

5.4.1 Construção

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

- Medir o interesse do público em participar de um evento;
- Criar uma interface intuitiva e rápida para mostrar informações de eventos para o usuário.

Figura 5.6: *Tela inicial na primeira iteração*

A página inicial (figura 5.6) do site possuía acesso para a página de cadastro e *login* além de um formulário para envio de sugestões.

O cadastro de usuário (figura 5.7) pedia inicialmente apenas nome, e-mail e senha porém ainda na primeira iteração foi implementada a opção de *login* com Facebook.

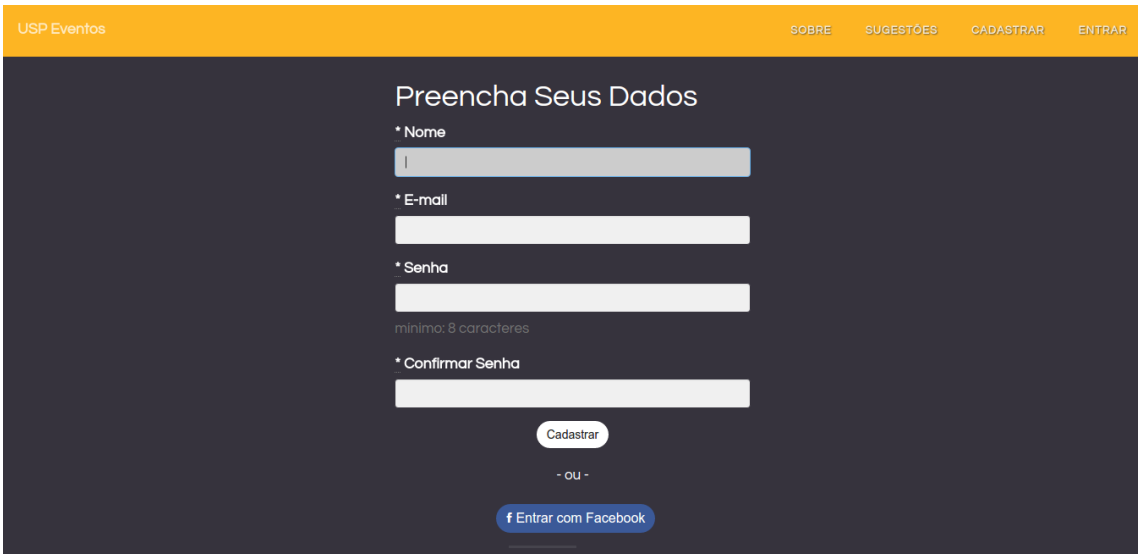


Figura 5.7: *Tela de Cadastro na primeira iteração*

A página de eventos (figura 5.8) 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 móvel, possuindo versões responsivas (figura 5.9).

Visando evitar que os usuários se deparassem com uma página de eventos vazia foi feita uma *Rake Task* [5] para consumir o xml gerado pelo *feed* RSS do 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 os usuários que tivessem realizado *login* e também botões para compartilhar nas redes sociais.

[4]Fonte: Desenvolvimento Ágil fontehttp://www.desenvolvimentoagil.com.br/scrum/sprint_backlog Acesso em: 22 out. 2016.

[5]Rake é um programa implementado em *Ruby* que permite ao usuário implementar *tasks* que são executadas ao serem chamadas

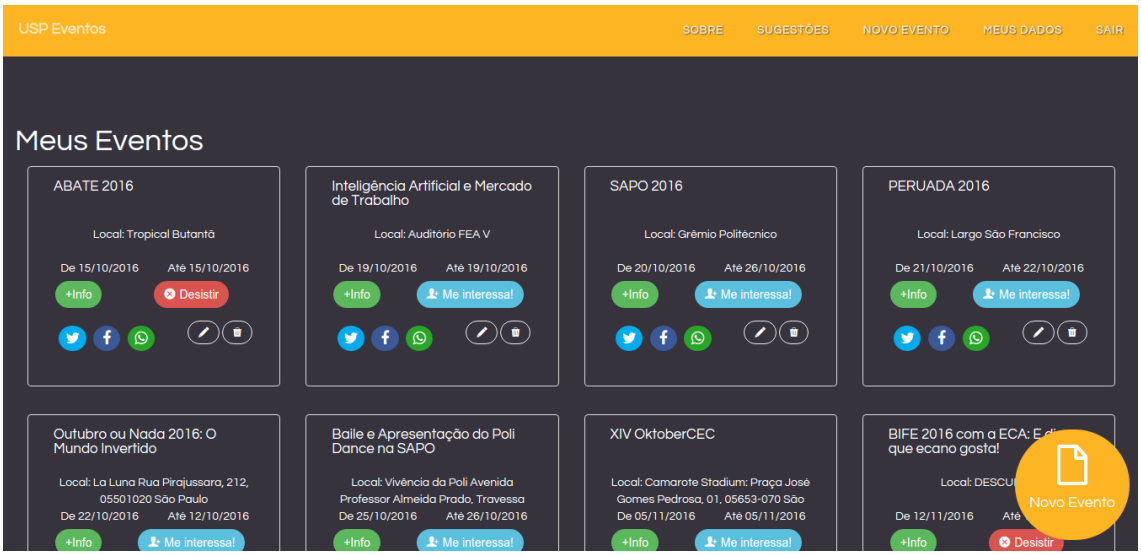


Figura 5.8: *Página de Eventos*

Figura 5.9: *Página de Eventos versão responsiva*

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

5.4.2 Divulgação

A primeira versão do sistema ficou disponível a partir do dia 5 de maio de 2016 e sua divulgação foi feita pelo Facebook por grupos e comunidades associadas a institutos da USP tais como FFLCH, FAU, IME e ECA, assim como foram enviadas mensagens para as respectivas empresas Júnior e Atléticas.

Também foi divulgado 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 foi utilizado Google Analytics em conjunto com o Google Tag Manager e como métrica chave escolhemos medir a quantidade de usuários que se interessavam por um evento.

Foi criada então uma *tag* (figura 5.10) 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.

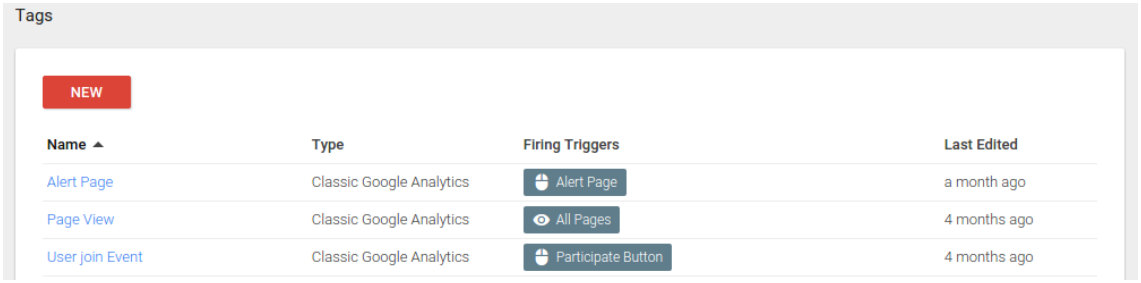


Figura 5.10: *Visualização das Tags pelo Google Tag Manager.*

Em pararelo foi possível obter também por meio do Google Analytics as seguintes informações (figura 5.11):

- Visualizações de Páginas: "Exibições de página" refere-se ao número total de páginas visualizadas. Exibições repetidas de uma única página são consideradas;
- Páginas / Sessão: Número total de sessões no período.
- Duração Média da Sessão: Uma sessão é o período que um usuário permanece ativamente engajado com seu site, aplicativo etc. Todos os dados de uso (exibições de tela, eventos, comércio eletrônico etc.) são associados a uma sessão;
- Usuários: Os usuários que realizaram pelo menos uma sessão no período selecionado. Inclui usuários novos e recorrentes;
- Taxa de Rejeição: A taxa de rejeição é a porcentagem de visitas a uma única página (ou seja, visitas nas quais a pessoa sai de seu site na mesma da página de entrada, sem interagir com a página);
- Porcentagem de Novas Sessões: Uma estimativa da porcentagem de primeiras visitas.

É possível observar uma grande taxa de rejeição inicial ao site no período associado principalmente na página inicial com cerca de 761 sessões e 423 desistências (figura 5.12).

Durante o período observado foram registrados apenas 184 cliques em 56 sessões únicas no botão "Participar".

Através do gráfico da divisão de uso por tipo de Sistema Operacional (figura 5.13) foi possível observar que a grande maioria dos usuários acessa o site por meio *notebooks* ou computadores pessoais.

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 acessavam o site porém não realizavam cadastro deixando-o logo em seguida e resultando em um número elevado de desistências na página inicial.

Dentre os retornos recebidos pelo formulário do site, e-mails e de forma direta foram compiladas algumas críticas e sugestões:

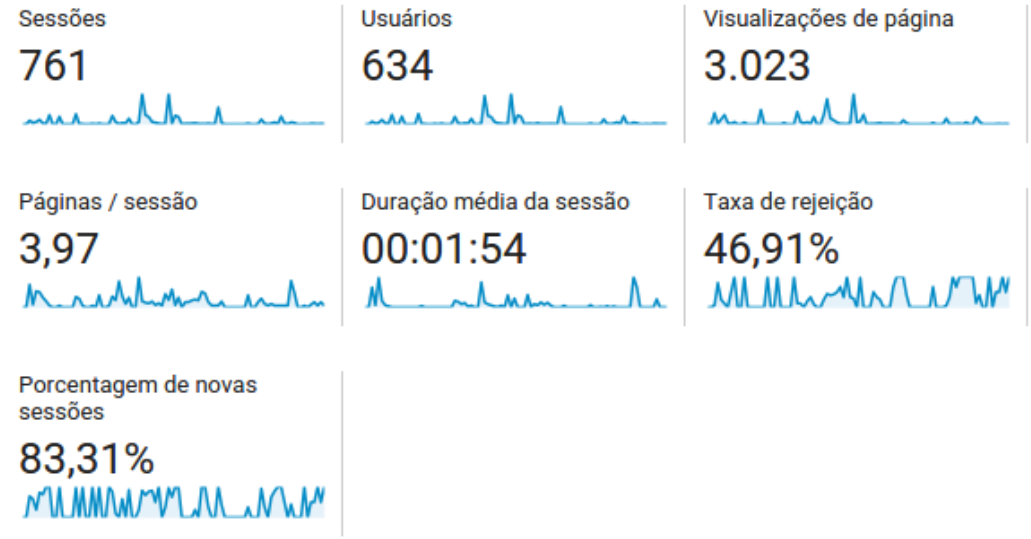


Figura 5.11: *Dados obtidos pelo G.A de 01/05 até 31/07*

Figura 5.12: *Fluxo de Comportamento de 01/05 até 31/07.*

- 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 não gerava nenhum reflexo no sistema, não existindo uma funcionalidade que justificasse sua existência e não havendo razão para que os usuários clicassem 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).*"

	761 Porcentagem do total: 100,00% (761)	761 Porcentagem do total: 100,00% (761)
1. Linux	374	49,15%
2. Macintosh	234	30,75%
3. Windows	82	10,78%
4. Android	64	8,41%
5. iOS	7	0,92%

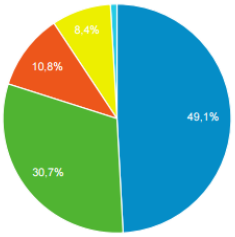


Figura 5.13: *Porcentagem de uso por S.O de 01/05 até 31/07.*

- 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 (inicio/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.

As hipóteses a serem testadas foram:

- Verificar se as alterações visuais foram bem aceitas;

- Testar a hipótese da necessidade de Filtro para Eventos.

Foi criado um filtro para a página de eventos baseado na utilização de *tags*. Dessa forma ao criar um novo evento (figura 5.14) o usuário agora pode escolher 3 dentre 12 tags pré-definidas que servirão como filtro na página principal de eventos.

Visando tornar a navegação dentro da página de eventos mais fluída e menos cansativa foram realizadas algumas modificações visuais na exibição dos eventos (figura 5.15).

- 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;

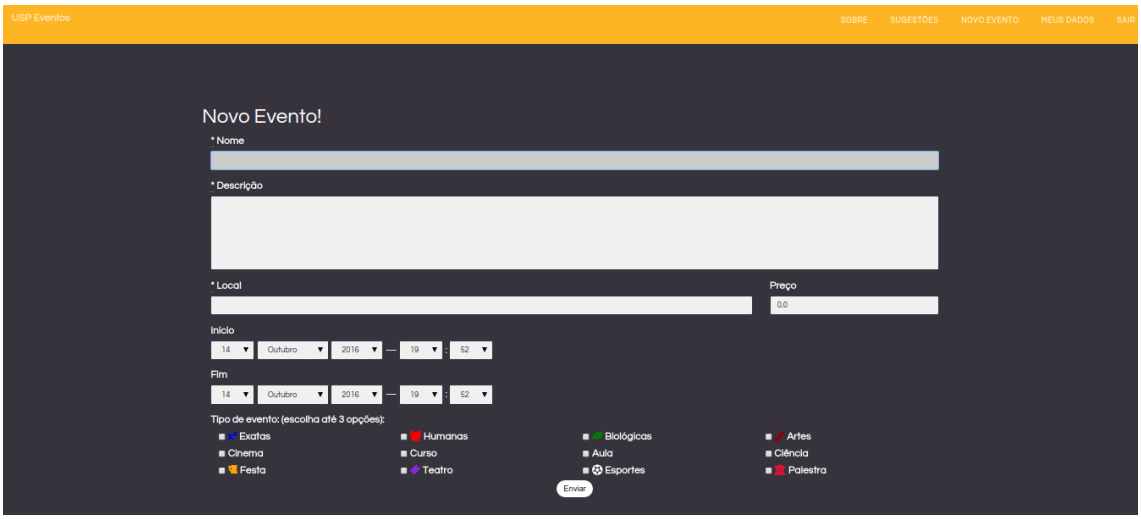


Figura 5.14: *Página de Cadastro de Novos eventos com filtros*

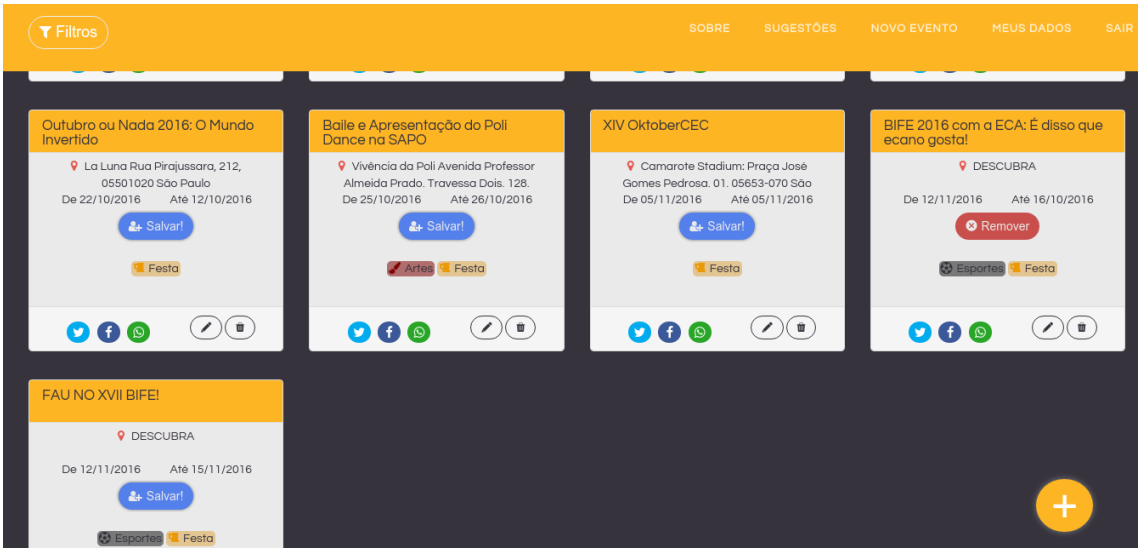


Figura 5.15: *Página de Eventos com as alterações para a segunda iteração*

- Nova listagem personalizada de Eventos segundo os interesses do usuário.

Os filtros (figura 5.16) 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 5.17) 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 sua escolha.
- 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

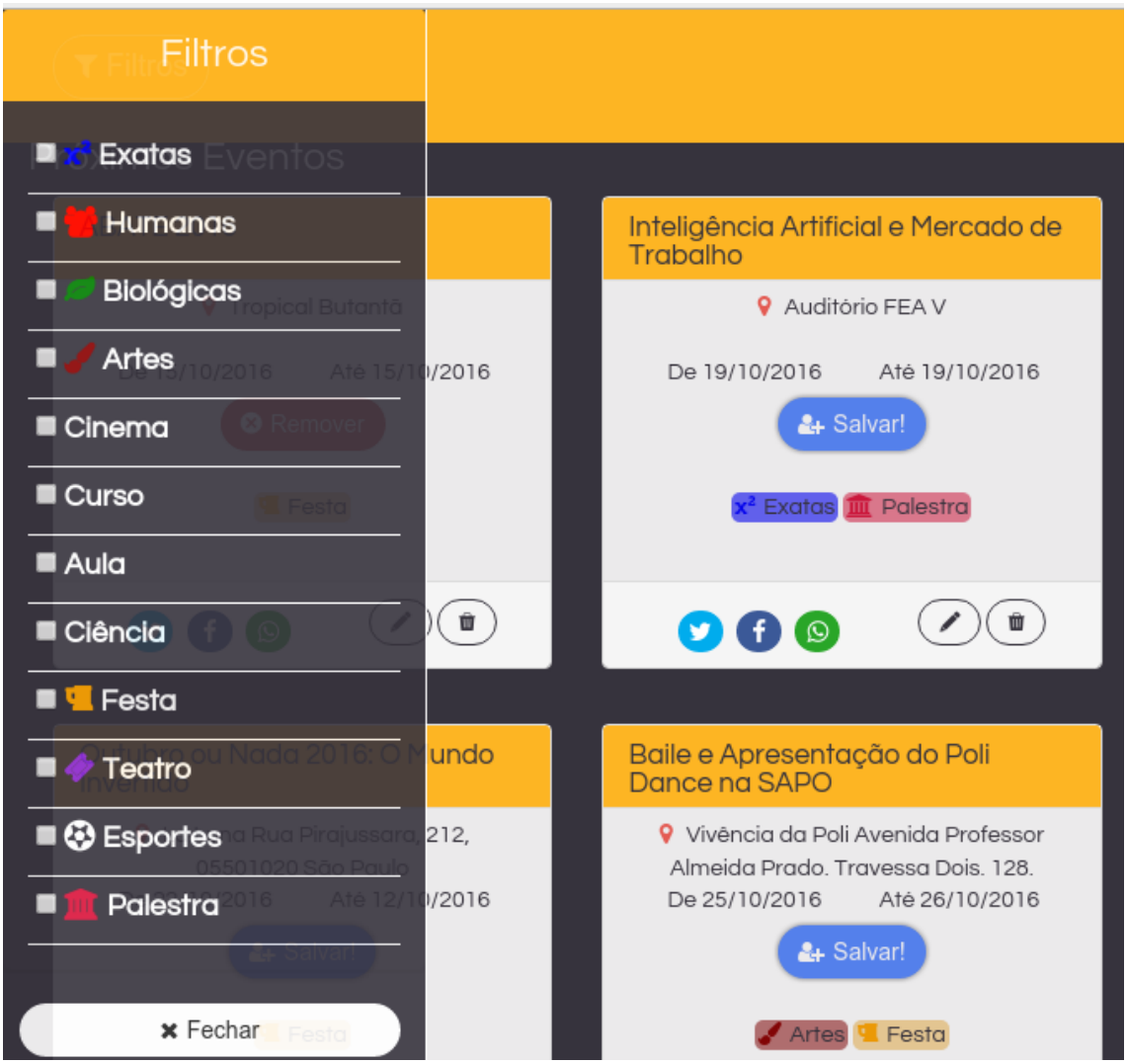


Figura 5.16: *Filtros na página de Eventos*

tags com as quais o usuário tenha maior afinidade, com o intuito de exibir uma listagem personalizada de eventos segundo esses critérios.

Em julho ocorreu o lançamento do jogo para smartphone Pokemon GO, baseado em georreferenciamento, cujo objetivo era explorar localidades reais em busca dos Pokemons para capturá-los. Esse lançamento movimentou uma enorme quantidade de pessoas pelo campus na época.

Visando aproveitar essa movimentação foi criada uma página chamada Alertas (figura 5.18) para atingir o público que estava jogando com intuito de que eles pudessem divulgar a localização dos Pokemons utilizando o USP Eventos.

A implementação da página de Alertas tão rapidamente só foi possível devido à grande flexibilidade que o modelo de Construir-Medir-Aprender oferece pois somente assim foi possível integrar uma nova funcionalidade não prevista dentro do escopo do projeto e medir sua eficácia.

Além disso, com o auxílio de testes automatizados, ferramentas para integração contínua e a agilidade do desenvolvimento em Rails foi possível desenvolver e colocar as alterações no ambiente de produção sem comprometer a integridade do sistema como um todo.

Por fim foi adicionada uma página "Sobre" com informações sobre os responsáveis pelo site assim como seus objetivos.

Figura 5.17: *Esquerda: versão antiga. Direita: Versão atualizada*

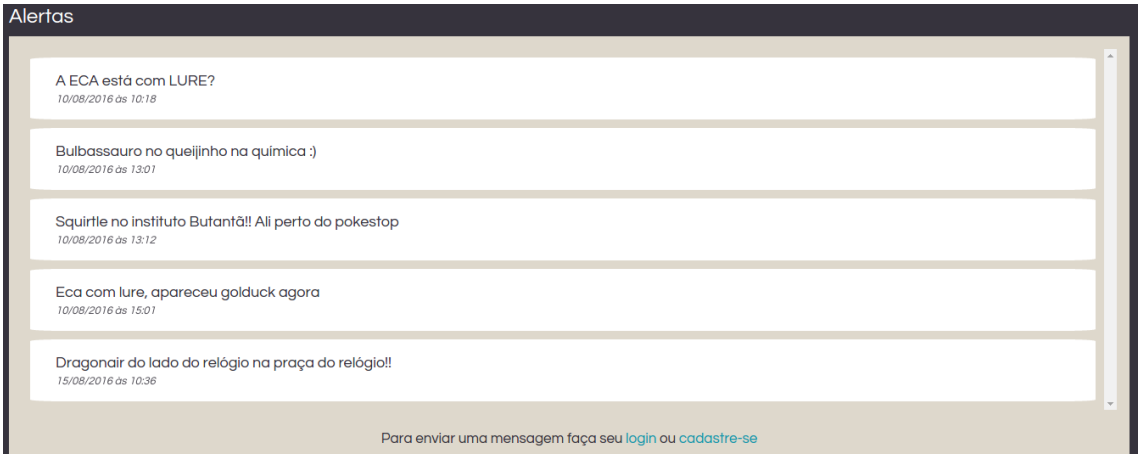


Figura 5.18: *Página de Alertas*

5.5.2 Divulgação

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

5.5.3 Métricas

Foi mantido o rastreamento pelo Google Analytics do botão "Participar", porém seu nome foi alterado para "Salvar" com o intuito de refletir melhor sua utilidade: salvar um evento como interessante para exibi-lo na seção de "Meus Eventos" da listagem de eventos do usuário. No entanto, a métrica chave continuou sendo avaliar o interesse dos usuários em determinado Evento, agora por meio do clique no botão "Salvar".

Pelos resultados obtidos através do Googgle Analytics (figura 5.20) foi observado uma diminuição no número de sessões. Entretanto também houve uma diminuição significativa na taxa de rejeição do site, caindo de 46,91% para 33,16%.

O tempo médio por sessão também aumentou, passando de 1:54 minutos para 3:37 minutos, mostrando um aumento na retenção de usuários acessando a plataforma.

Com o lançamento da seção de Alertas foi feita uma divulgação via Facebook, incen-

Figura 5.19: *Cartaz de divulgação.*

tivando os usuários a utilizarem a plataforma com o intuito de divulgar a localização de Pokemons.

Foi colocado também uma *tag* para rastrear o número de cliques no botão "Alertas" com a intenção de medir o interesse na funcionalidade. Dessa forma foram observados 305 cliques nesse botão no período observado.

Analisando o gráfico de divisão por tipo de Sistema Operacional (figura 5.21) foi observado que o acesso pelo sistema Android proporcionalmente mais que dobrou em relação ao período anterior, passando de 8,4% para 19,36%.

5.5.4 Aprendizado

Com a abertura da página principal de Eventos, sem a obrigatoriedade de um cadastro, mais acessos foram registrados, porém quase não houve novos cadastros, dificultando assim que um usuário salvasse algum evento para sua lista.

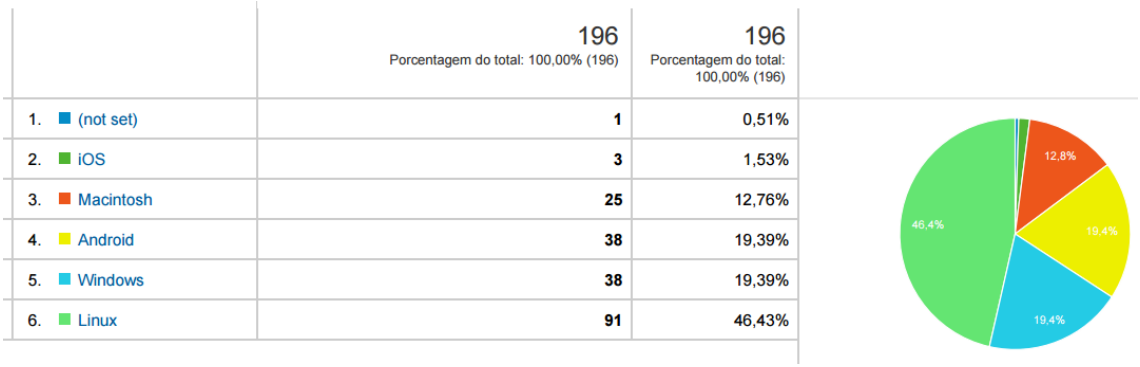
Com a divulgação por cartazes foi possível constatar um aumento na utilização em dispositivos móveis, principal forma de acesso em lugares públicos e incentivada devido ao QR Code presente nos cartazes.

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

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

Alguns comentários recebidos de forma oral afirmaram que a página de eventos estava pouco atrativa visualmente, sendo necessário que ela tivesse mais elementos que prendessem

Figura 5.20: *Dados obtidos pelo G.A de 01/08 até 30/09*



	196 Porcentagem do total: 100,00% (196)	196 Porcentagem do total: 100,00% (196)
1. (not set)	1	0,51%
2. iOS	3	1,53%
3. Macintosh	25	12,76%
4. Android	38	19,39%
5. Windows	38	19,39%
6. Linux	91	46,43%

Figura 5.21: *Porcentagem de uso por S.O de 01/08 até 30/09.*

a atenção do usuário.

Novamente recebemos comentários pelo próprio formulário do site sobre adicionar a opção de incluir uma foto ao evento. Segue o comentário do Ferdinand Machado: "*Sinto falta de anexo para cartazes. poder enviar uma foto ou cartaz escaneado do evento.*".

5.6 Terceira Iteração

5.6.1 Construção

As maiores críticas recebidas durante a última iteração foram em relação à experiência proporcionada pelo site, que não estava atrativa o suficiente. A partir desse *feedback* foi decidido testar a seguinte a hipótese:

- Melhorar a UX do site para aumentar a aceitação dos usuários.

Em seu artigo publicado na *Agile Conference* Beverly May (May, 2012), especialista em UX, discute alguns dos erros que cometeu ao aplicar o método de *Lean Startup*. Dentre eles um dos principais foi negligenciar a UX inicialmente.

Como recomendação ela aconselha investir numa boa experiência de usuário sempre construindo protótipos e *wireframes*[6] antes de implementar para testar as modificações visuais. Berverly também enaltece a importância de realizar testes constantes e sempre levar em consideração os *feedbacks* dos usuários, positivos e negativos, um conceito também presente no ciclo de Construir-Medir-Aprender e base para o Desenvolvimento de Clientes.

No caso do USP Eventos foi feito um *wireframe* (figura 5.22) da página principal de Eventos com as modificações realizadas na iteração atual.

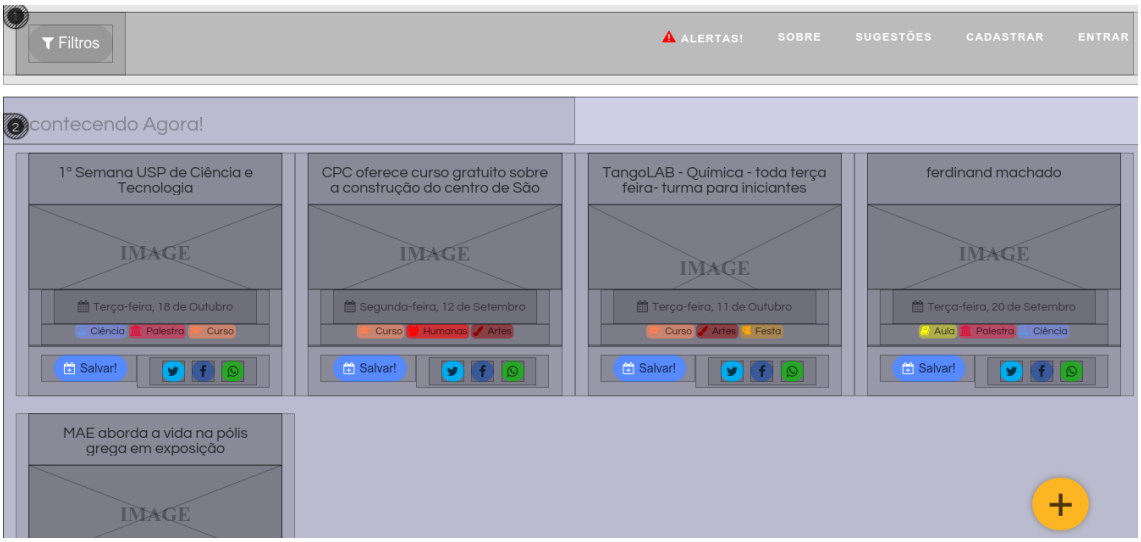


Figura 5.22: *Wireframe da versão modificada durante a Terceira Iteração*

Para contornar a falta de conhecimento na área de *User Experience* foram feitas pesquisas utilizando a bibliografia disponível e também entrevistas com profissionais da área.

Isto posto, para melhorar a exibição de informação dentro do site, centralizamos as principais modificações de UX na página de Eventos, seguindo alguns preceitos apresentados por Steve Krug em seu livro "Don't make me think" (Krug, 2000), dentre eles:

- Eliminar distrações desnecessárias: eliminar espaços em branco e textos que possam distrair, poluir visualmente a página ou criar algum tipo de ruído na informação exibida;
- Criar hierarquias visuais claras: dar mais destaque para as informações importantes e aproximar visualmente elementos que possuam ligações lógicas entre eles, como nome do evento e sua data.
- Tirar vantagens de convenções: utilizar *layouts* já consolidados de sites semelhantes para criar uma identificação na forma de navegar do usuário.

Além disso, para eliminar o excesso de texto presente na exibição de eventos e incluir mais informações visuais foi implementada uma opção de realizar *upload* de uma imagem para o evento que seria exibida tanto na página principal de eventos como na página de informações gerais.

As modificações principais feitas foram (figura 5.23):

- Diminuir os espaços em branco dentro do *thumbnail* de Eventos, deixando-o mais compacto;

[6] Um wireframe web é uma ilustração semelhante do layout de elementos fundamentais na interface. , Fonte: https://pt.wikipedia.org/wiki/Website_wireframe Acesso em: 6 out. 2016.

- Modificação nas cores e tamanho para dar destaque ao título, criando uma hierarquia visual a partir dele com as suas informações contidas no interior do *thumbnail*;
- Eliminação da exibição do local e data de término do evento para diminuir a poluição visual;
- Presença de uma imagem identificadora no evento ao centro do thumbnail e em sua página de exibição;
- O botão de "Salvar" deixou de estar localizado ao centro para ficar associado com os outros botões de compartilhamento na parte inferior do *thumbnail*.

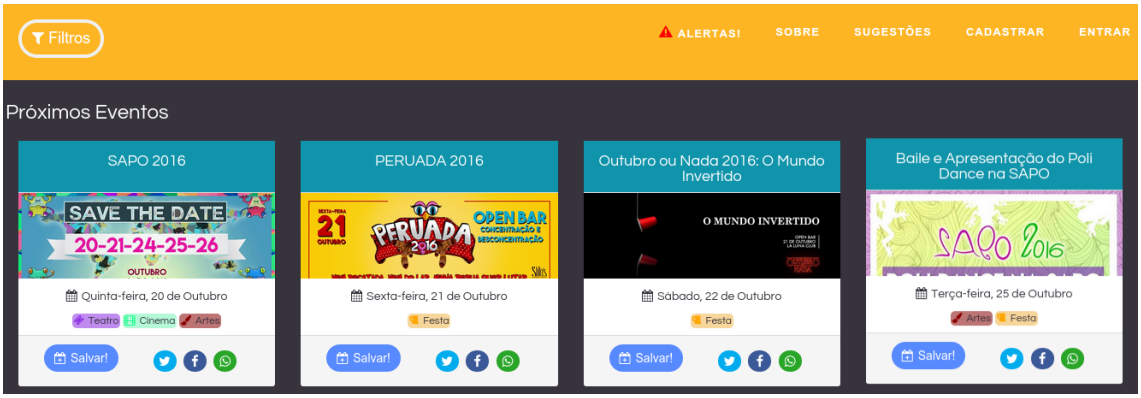


Figura 5.23: *Página de Eventos após modificações*

Victor Krug também afirma em seu livro que os usuários costumam criar "mapas mentais" de navegação, sendo importante manter as convenções com o intuito de aproveitar-se dessa similaridade.

Tomando como um exemplo de convenção o Sympla (www.sympla.com.br) uma plataforma completa para venda de ingressos e, inscrições e gestão de eventos, foi observado que desde o começo nosso fluxo de navegação estava bastante próximo à convenção estabelecida: uma página principal de Eventos contendo toda uma listagem de eventos sendo, que cada evento direciona para sua própria página específica.

Para aproximar ainda mais o fluxo de navegação foi incluída também na página inicial uma listagem com os "Próximos Eventos" assim como o Sympla faz com seus "Eventos em Destaque".

Além disso foram implementados alguns efeitos visuais para chamar atenção do usuário:

- Ao passar o mouse sobre evento, uma animação de salto do *thumbnail* era rapidamente exibida;
- Ao carregar ou atualizar uma listagem de eventos, seu carregamento era feito por meio de uma animação iniciada lateralmente;

A maior dificuldade em realizar o *upload* de imagens foi o seu local de armazenamento, pois o Heroku não permite salvar arquivos em seu servidor, apenas armazenar em *cache* durante a sessão, então foi feita a opção de criar uma conta no Dropbox habilitada para receber imagens de aplicativos e integrá-la com a aplicação USP Eventos.

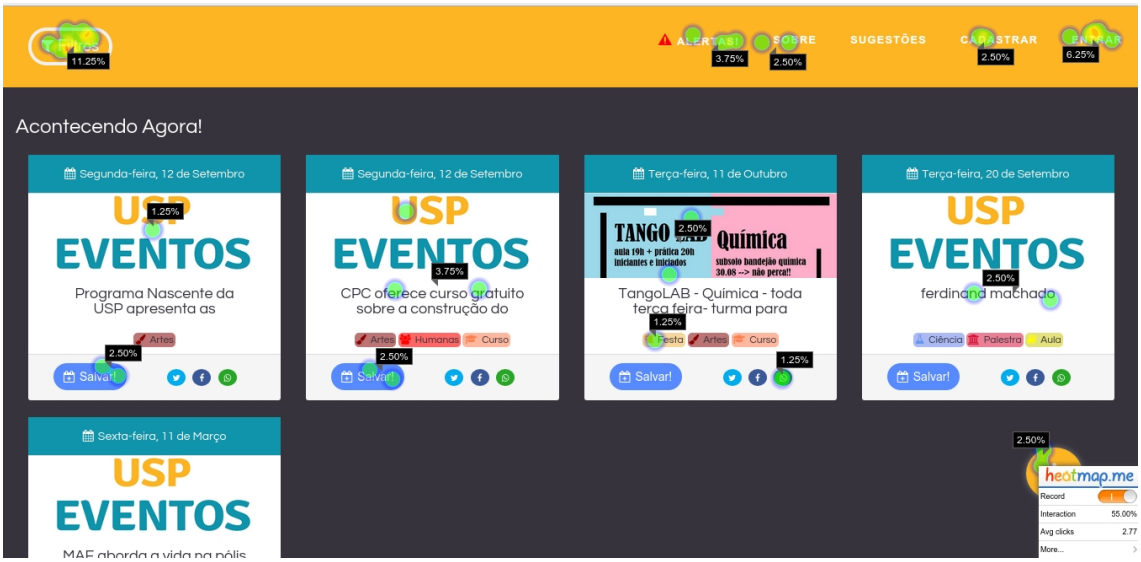


Figura 5.24: *Página principal de Eventos com o Mapa de Calor ativado*

.

5.6.2 Métricas

Nessa iteração foi adicionado um Mapa de Calor (figura 5.24) para medir os cliques de mouse na página principal de eventos.

Foi interessante observar que o Filtro estava sendo bastante utilizado, tanto que em dado momento atingiu 40% dos cliques na página (figura 5.25).

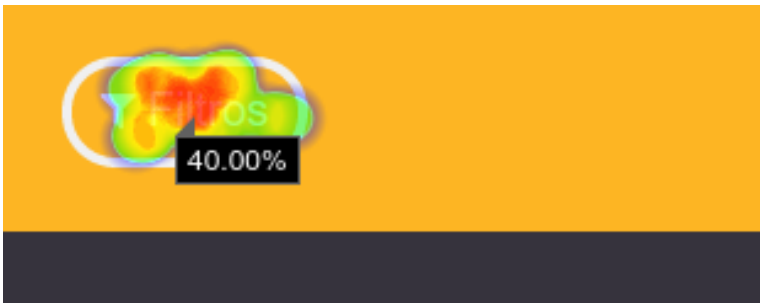


Figura 5.25: *Botão de Filtro com 40% dos cliques da página*

.

Outro ponto interessante observado é que muitos usuários também clicavam nas imagens dos eventos para acessar suas páginas de informações individuais, mostrando que adicionar uma imagem para captar a atenção do usuário trouxe resultados.

O enunciado de cada listagem foi clicado repetidas vezes pelos usuários, o que pode significar que ele esteja sendo confundido com um *link* (figura 5.26).

Figura 5.26: *Nome da Listagem: possível confusão com um link*

.

Analisando os dados (figura 5.27) obtidos pelo Google Analytics no período de 27/09 até 28/10 é possível observar uma diminuição na taxa de rejeição no site para 21,64%, e um

aumento considerável no número de páginas visitadas e duração média por sessão.

Figura 5.27: *Dados obtidos pelo G.A no período de 27/09 até 28/10*

.

Além disso o número de novos visitantes ficou em 47,2%, mostrando que existe um número considerável de usuários que retornaram ao site, o que pode justificar um investimento maior em divulgação.

5.6.3 Aprendizado

Com o intuito de realizar uma coleta de dados mais direta com os candidatos, foi criado um Formulário utilizando o TypeForm e conversas com pessoas que utilizavam a plataforma.

Ao responder à pergunta "Qual ou quais foram os maiores pontos positivos na sua opinião?" a interface do site foi elogiada diversas vezes, mostrando que as modificações foram de fato bem aceitas. A seguir estão listadas algumas das respostas:

- "Interface bem fácil e intuitiva";
- "Organizar visualmente a informação dos eventos e a utilização de *labels*.";
- "Intuitivo em sua maioria, bonito, fácil de entender como funciona";
- "Mostrar eventos de diversos temas (não só festas, por exemplo). Mas também, a opção de selecionar os assuntos de sua preferência, ao fazer o cadastro. A interface é simples e clara, acho que atende aos objetivos e permite uma visualização rápida, podendo rolar até o mês desejável rapidamente.";
- "Reunir todos os eventos do campus em um só lugar.";
- "Gostei da preocupação em se fazer um site responsivo, já que, por ter inclusive uma seção Acontecendo Agora, é de se esperar que o acesso por meios móveis seja maior.";

Também foi feita uma pergunta sobre os pontos negativos e foi observado que alguns usuários acreditavam que os filtros não estivessem funcionando.

Conversando pessoalmente com esses usuários foi identificada uma confusão na forma como o filtro é aplicado. Como o filtro é aplicado somente nas listagens de "Eventos Acontecendo Agora" e "Próximos Eventos", esses usuários não percebiam a filtragem estava sendo realizada.

Ainda sobre os filtros, aparentemente não ficou claro para os usuários que ao preencher suas preferências durante o cadastro, isso iria criar uma listagem personalizada com o nome de "Principais Escolhas".

Alguns comentários sobre os filtros:

- "No meu computador o site demora a responder e alguns filtros não funcionaram.";
- "Tentei usar o filtro e não funcionou mto (sic) bem.".

A página de Alertas também pareceu deslocada. Muitos não entenderam sua função e comentaram que o seu *layout* atual está ruim.

Segue uma compilação das sugestões de funcionalidades recebidas:

- Filtros por data (dias da semana, próximos dias, etc);
- Integrar com o Google Agenda;
- Integrar com os Eventos do Facebook (poder importar eventos do facebook para a plataforma);
- Presença de um calendário;
- Campo de busca por extenso;
- Mapa com a localização dos Eventos;
- Sistema para disparar alertas ou lembretes para determinado evento;
- Opção de cadastros oficiais, como uma certificação de que aquele perfil pertece a um Centro Acadêmico oficialmente por exemplo;
- *Feed* de Notícias;
- Sistema de Avaliação para Eventos mais esperado;
- Vídeos e fotos pós evento.

Algumas respostas sobre sugestões para a plataforma:

- "Talvez criar mais uma separação "eventos nos próximos 7 dias" ou "eventos dessa semana", visualização de mapa?";
- "Senti falta de uma visualização dos eventos em um calendário.";
- "Podia linkar (sic) com os eventos do Facebook e já salvar nos seus eventos de lá.";
- "Talvez, um campo de busca para que fosse possível escolher um mês ou fim de semana específico...É rápido ir rolando a tela, mas para eventos a longo prazo, a barra de busca cumpriria melhor a função.".

Ao todo 28 pessoas responderam ao questionário, sendo que todas acreditam que uma plataforma centralizando Eventos na USP é de fato útil e apenas 3 não gostaram do USP Eventos.

Dentre as críticas, das 3 pessoas que não gostaram os seguintes pontos foram levantados:

- Acreditam que poucas pessoas hoje em dia usariam um site e talvez uma iniciativa dentro do Facebook fosse mais útil;
- Ficaram incomodadas com a paleta de cores;
- Gostariam de poder visualizar os eventos na página inicial.

A abordagem utilizando um questionário e conversas diretas trouxe uma quantidade muito maior de informação sobre a plataforma do que nas iterações anteriores, mostrando que o Desenvolvimento do Cliente é realmente útil para direcionar o desenvolvimento do Sistema.

Utilizar um Mapa de Calor foi um acréscimo importante na coleta de dados pois foi possível perceber detalhes na forma como o usuário utiliza a plataforma, tornando-se mais uma opção para realizar a validação da hipótese inicial.

A divulgação ainda parece não estar atingindo um grande número de pessoas resultando em um baixo volume de acessos.

5.7 Últimas Atualizações

As últimas alterações feitas no site antes da conclusão do projeto foram:

- Login e Cadastro com o Gmail: analisando a boa receptividade da integração com redes sociais foi decidido expandir as opções de *login*.
- *Cache*: uma das reclamações recebidas foi a lentidão do site devido ao fato da aplicação não realizar o *cache* das imagens de eventos, sendo necessários recarregá-las sempre. Implementando a opção de *cache* fornecida pelo próprio Rails foi possível diminuir de forma considerável o tempo de carregamento da página de eventos.

Capítulo 6

Conclusões

Um dos fatores principais para o *Lean Startup* ter sido escolhido para direcionar a pesquisa e a elaboração do USP eventos foi a importância que o cliente tem nesse processo. A fase inicial - de levantamento de interesse por meio de enquete com os alunos - constatou que nem sempre a concepção sobre determinado projeto ou ideia do ponto de visto dos desenvolvedores é, de fato, uma necessidade para o público-alvo.

Nos 3 ciclos de Construir-Medir-Aprender foi possível observar as reações dos usuários e por diversas vezes um comentário ou crítica serviu para modificar uma funcionalidade ou incluir uma nova. Isso demonstra a importância de manter o desenvolvimento sempre em contato com o usuário final, para se obter um software que de fato cumpra com sua proposta e seja útil ao usuário. O desenvolvimento em ciclos também mostrou a importância que existe em obter um aprendizado válido para de fato entender a receptividade e necessidade das atualizações.

Um desenvolvimento pautado em testes por vezes impediu que um *"bug"* fosse colocado em produção, que apesar de exigir um tempo maior de trabalho, a longo prazo, os testes automatizados geraram ganho em produtividade e eficiência. A integração contínua é uma ferramenta poderosa para situar todos os integrantes do projeto sobre seu estado atual, além de garantir a confiabilidade do sistema.

De uma maneira geral foi bastante positivo o resultado final da aplicação. Conseguimos entregar um software sólido, no entanto, apesar de ter sido possível obter *feedbacks* suficientes para melhoria contínua do sistema, infelizmente não tivemos uma adoção de usuários tão grande quanto esperada. Talvez com maior divulgação um público maior pudesse ter aderido e se cadastrado na aplicação.

Com o aprendizado adquirido fica clara a importância do tempo dedicado à divulgação. No começo do projeto hesitamos em realizar uma divulgação maior esperando o projeto estar mais concreto. Contudo, dada a importância que teve o *feedback* dos usuários, hoje seriam espalhados cartazes desde a primeira versão do projeto além de realizar-se mais questionários e conversas com os usuários, dependendo menos do envio de sugestões.

No início do projeto havia a intenção de manter *sprints* de desenvolvimento bem definidos, com prazos claros. No entanto com o passar do tempo devido a problemas de horário e disponibilidade tais *sprints* ficaram muito irregulares e mal definidos. Olhando em retrospecto isso quebrou um pouco o ritmo de desenvolvimento e atrasou a progressão do projeto como um todo. Hoje insistiríamos em realizar sprints bem definidos.

Capítulo 7

Próximos Passos

O sistema ainda tem muito espaço para evolução. Durante a última interação foram recebidos uma série de *feedbacks* e opções para continuar o desenvolvimento do software:

- Integrar o sistema com o Google Agenda para notificar usuários dos seus eventos de interesse;
- Criar uma página contendo um Feed de notícias com as novidades ocorrendo no Campus;
- Opção de Login com Instagram, Twitter e outras redes sociais;
- Remodelar a página de Alertas com um *layout* mais atrativo

Referências Bibliográficas

Blank(2003) Steve Blank. *The Four steps to Epiphany*. KS Ranch, quarta edição. Citado na pág. 3, 7

Blank(2015) Steve Blank. Construir, medir, aprender? entenda as formas de validar seu negócio, 2015. URL https://endeavor.org.br/construir-medir-aprender/. Citado na pág. 7

Documentation(2016) Ruby Documentation. Sobre o ruby, 2016. URL https://www.ruby-lang.org/pt/about/. Citado na pág. 17

Filho(2014) Francisco Barreto Costa Pimentel Filho. Um estudo da adoção das práticas de lean startup, business model canvas e desenvolvimento de clientes para startups. Dissertação de Mestrado, Universidade Federal de Pernambuco. Citado na pág. 29, 30

Junk(2000) W. S. Junk. The dynamic balance between cost, schedule, features, and quality in software development projects. *Computer Science Dept., University of Idaho*. Citado na pág. 4

Krug(2000) Steve Krug. *Don't Make Me Think*. New Riders Press, terceira edição. Citado na pág. 42

Larman(2004) Craig Larman. *Agile and Iterative Development: A Manager's Guide*. Addison Wesley, segunda edição. Citado na pág. 10

May(2012) Beverly May. Applying lean startup: An experience report lean lean ux by a ux veteran: Lessons learned in creating launching a complex consumer app. *Agile Conference (AGILE), 2012*. Citado na pág. 41

Morrice(2015) Gavin Morrice. Why you should build your web startup using ruby on rails, 2015. URL https://blog.katanacode.com/why-you-should-build-your-web-startup-using-ruby-on-rails-8104c2226c6a#.qgpihasir. Citado na pág. 19

Paternoster(2014) Nicolo Paternoster. Software development in startup companies: A systematic mapping study. *Information and Software Technology*. Citado na pág. 3

Pressman(2011) Roger S. Pressman. *Livro - Engenharia de Software - Uma Abordagem Profissional*. Mc Graw Hill, sétima edição. Citado na pág. 14

Ries(2011) Eric Ries. *The Lean Startup*. Crown Business (USA), segunda edição. Citado na pág. 4, 6

Udovychenko(2016) Lilia Udovychenko. Why startups use ruby on rails, 2016. URL http://mlsdev.com/en/blog/61-why-startups-use-ruby-on-rails. Citado na pág. 19, 20