INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

ELOMAR DE FRANÇA COSTA E SOUZA

DESIGN QUIPS: AUXILIANDO O PROCESSO DE DESENVOLVIMENTO WEB COM DADOS ESTATÍSTICOS

Natal - RN
2013

ELOMAR DE FRANÇA COSTA E SOUZA

DESIGN QUIPS: AUXILIANDO O PROCESSO DE DESENVOLVIMENTO WEB COM DADOS ESTATÍSTICOS

Monografia apresentada à Banca Examinadora do Trabalho de Conclusão do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Leonardo Ataide Minora

Natal - RN
2013

ELOMAR DE FRANÇA COSTA E SOUZA

DESIGN QUIPS: UM ESTUDO DE CASO EM DESENVOLVIMENTO ÁGIL UTILIZANDO RUBY ON RAILS

Aprovado em: ___/___/___

BANCA EXAMINADORA

_____

Leonardo Ataide Minora
Instituto Federal do Rio Grande do Norte

_____

Fellipe Araújo Aleixo
Instituto Federal do Rio Grande do Norte

_____

Gilbert Azevedo da Silva
Instituto Federal do Rio Grande do Norte

AGRADECIMENTOS

Agradeço, antes de qualquer outra coisa, à Diretoria Acadêmica de Gestão e Tecnologia da Informática (antiga GEINF), que através de seus professores, funcionários, e estrutura física, que fez do IFRN uma segunda casa. Em especial agradeço ao prof. Leonardo Minora, que vem me ajudando desde os tempos de +web; e a dona Maria, que sempre me recebeu com um sorriso.

Agradeço também ao prof. Marcelo Camilo, que como um pai me apoiou durante toda a minha carreira no IFRN. É difícil descrever o quanto ganhei participando dos programas de intercâmbio da Fulbright e da CAPES, e nada disso teria sido possível sem a ajuda do prof. Marcelo.

Agradeço ainda a todos os alunos com quem tive contato enquanto aluno de TADS, batendo cabeça juntos pra resolver listas de exercício, atacando o dragão que é um PDS, ou conversando entre aulas.

“I learned this, at least, by my experiment: that if one advances confidently in the direction of his dreams, and endeavors to live the life which he has imagined, he will meet with a success unexpected in common hours”
Henry David Thoreau, Walden, 1855

RESUMO

Designers tomam decisões no dia a dia baseados em sua intuição, e não com o suporte evidência concreta. Foi observando essa realidade que a consultoria de design ZURB, na California, decidiu desenvolver uma ferramenta que permitisse a designers consultarem rapidamente estatísticas e tendências sobre o desenvolvimento web para suportar e justificar suas decisões frente a clientes. Design Quips, como veio a se chamar essa ferramenta, foi desenvolvida entre Maio e Agosto de 2013, usando um processo de desenvolvimento ágil adaptado para a realidade da ZURB usando a plataforma Ruby on Rails. Esse trabalho trata do desenvolvimento dessa solução, concentrando-se nas práticas, papéis e etapas adotados no processo de software utilizado.

Palavras-chave: Desenvolvimento Ágil, Processo de Software, Ruby on Rails.

ABSTRACT

Designers make decisions in their day to day work based on their intuition, and not on the support of concrete evidence. It was by observing this reality that the design consulting firm ZURB, in California, decided to develop a tool that allowed designers to research quickly statistics and trends on web development in order to support and justify their decisions to clients. Design Quips, the name given to the proposed tool, was developed between May and August of 2013, using an agile development process adapted for ZURB's reality and on the Ruby on Rails platform. This document deals with the development of this solution, focusing on the practices, roles, and stages adopted on the software process utilized on the project.

Keywords: Agile Development, Software Process, Ruby on Rails.

LISTA DE ILUSTRAÇÕES

Figura 1. Ambiente principal de trabalho 8
Figura 2. Sala de reunião e trabalho individual 9
Figura 3. Diagrama de funcionamento do MVC clássico 16
Figura 4. Diagrama de funcionamento do MVC como implementado no Ruby on Rails 17
Figura 5. Menu off-canvas no estado inicial e no estado visível 19
Figura 6. Design respondendo a uma mudança na resolução do dispositivo 20
Figura 7. Página inicial 22
Figura 8. Página de detalhes de um dado 23
Figura 9. Página inicial para administradores 24
Figura 10. Listagem de categorias 24
Figura 11. Cadastro de dado individual 25
Figura 12. Cadastro na lista de emails 26
Figura 13. Exemplo de propagandas exibidas no site 27
Figura 14. Protótipo de página inicial desenvolvido na concepção 28
Figura 15. Protótipo de página de detalhes de um dado desenvolvido na concepção 29
Figura 16. Comentários do time no protótipo inicial 30
Figura 17. Barra de filtragem adaptada para dispositivos móveis, estado inicial e estado visível 32
Figura 18. Segunda versão da página inicial 34
Figura 19. Terceira versão da página inicial 35

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CEO	*Chief Executive Officer*
CSS	*Cascading Style Sheets*
CSV	*Comma Separated Values*
DRY	*Don’t Repeat Yourself*
HTML	*HyperText Markup Language*
LSI	*Latent Semantic Indexing*
MVC	Modelo-Visão-Controle
ORM	*Object-Relational Mapper*
TCC	Trabalho de conclusão de curso
URL	*Universal Resource Locator*
XP	*eXtreme Programming*

SUMÁRIO

RESUMO **6**

ABSTRACT **7**

LISTA DE ILUSTRAÇÕES **8**

LISTA DE ABREVIATURAS E SIGLAS **9**

1. INTRODUÇÃO **1**

1.1. Contexto e Motivação do Trabalho 1

1.2. Objetivos 2

1.2.1. Objetivo Geral 2

1.2.2. Objetivos Específicos 2

1.3. Metodologia 2

1.4. Estrutura do trabalho 3

2. Processo de desenvolvimento de software **4**

2.1. Processo de Desenvolvimento 4

2.1.1. Processos Ágeis 4

2.1.2. Processo de Desenvolvimento ZURB 5

2.2. Tecnologias Utilizadas 14

2.2.1. Ruby on Rails 14

2.2.2. Foundation Framework 18

3. DESENVOLVIMENTO DO DESIGN QUIPS **21**

3.1. Introdução ao Design Quips 21

3.2. Desenvolvimento do Projeto 27

3.2.1. Concepção 27

3.2.2. Desenvolvimento 29

3.2.3. Lançamento 33

3.2.4. Acompanhamento 34

4. CONCLUSÃO **36**

4.1. Trabalhos Futuros 36

REFERÊNCIAS BIBLIOGRÁFICAS **37**

1. INTRODUÇÃO

Entre 2012 e 2013 participei de um intercâmbio oferecido através do programa Ciência sem Fronteiras. O objetivo principal do programa era cursar disciplinas acadêmicas nos EUA, mas também havia um componente de estágio de trabalho obrigatório. Tal estágio deveria acontecer no período de férias, entre Maio e Agosto de 2013, e durar de 8 a 12 semanas. Meu interesse pelo Vale do Silício, região no estado da Califórnia conhecida como pólo tecnológico nos EUA, me levou a procurar empresas na área que tivessem vagas abertas para estagiários em desenvolvimento de software. Depois de um processo de seleção de cerca de três semanas, em Março de 2013 fui aceito pela ZURB, uma agência de consultoria em design. Em Maio de 2013, logo após o fim do semestre letivo, comecei o estágio.

Durante o estágio fui um dos responsáveis pelo desenvolvimento Design Quips, que é uma ferramenta criada para auxiliar designers em seu processo de trabalho. Esse trabalho apresenta as motivações para a criação dessa ferramenta e documenta o seu desenvolvimento, dos pontos de vista técnico e de processo.

1.1. Contexto e Motivação do Trabalho

A empresa ZURB constatou que designers “frequentemente usam intuição para criar” (ZURB, 2013). Apesar de não haver nada inerentemente errado com essa abordagem, “algumas vezes decisões [de design] precisam de mais suporte que instinto ou até dados analíticos” (ZURB, 2013). Na experiência da ZURB, nem sempre os clientes concordam com decisões de design, que precisam então ser justificadas e ancoradas em evidência. Nesses casos é necessário buscar exteriormente validação através de pesquisas de comportamento e tendências de mercado.

O Design Quips surgiu nesse contexto, motivado pela necessidade de designers de suportarem suas decisões com dados concretos e de facilitar o acesso a informações estatísticas relacionadas ao desenvolvimento web.

1.2. Objetivos

1.2.1. Objetivo Geral

O objetivo desse trabalho é documentar o desenvolvimento do projeto Design Quips, do ponto de vista tecnológico e de processo. O objetivo geral do Design Quips foi especificado pela ZURB quando da concepção do projeto.

1.2.2. Objetivos Específicos

Fazem parte dos objetivos específicos deste trabalho:

- Mapear requisitos funcionais a partir dos objetivos gerais pré-estabelecidos do Design Quips;
- Criar protótipos estáticos de tela no framework Foundation usando as linguagens HTML e CSS;
- Criar protótipos de funcionalidades na plataforma Ruby on Rails usando as linguagens HTML, CSS e Ruby;
- Desenvolver o código-fonte Ruby on Rails de implementação do software Design Quips;
- E lançar o Design Quips publicamente num prazo de três meses do início do desenvolvimento.

1.3. Metodologia

Design Quips foi desenvolvido durante o estágio de 3 meses na empresa ZURB. Antes do início do estágio houve uma curta etapa de concepção, onde designers da empresa levantaram requisitos e produziram wireframes iniciais de uma possível solução.

A essa etapa seguiu-se o desenvolvimento, seguindo um processo ágil centrado em design baseado em princípios do *Extreme Programming* (XP). Tal processo era caracterizado por etapas de tamanho variável, integração contínua, e comunicação frequente entre o time.

Após o desenvolvimento veio a etapa de lançamento, curta e com atividades próprias; e com o projeto lançado foi iniciada a etapa de acompanhamento, que consiste de acompanhar o uso do projeto e aperfeiçoa-lo continuamente.

Por fim, foi produzido esse trabalho, documentando as fases desse processo, sua realização concreta durante o projeto, e as tecnologias utilizadas em seu desenvolvimento.

1.4. Estrutura do trabalho

Este TCC se divide em quatro capítulos: introdução, referencial teórico, desenvolvimento do projeto, e conclusões. O capítulo introdutório apresenta o projeto em si. No referencial teórico são documentados o processo utilizado na ZURB, os processos nos quais se baseia, os funcionamento da empresa, os papéis, práticas e etapas do processo; e as tecnologias utilizadas durante o projeto. O capítulo do desenvolvimento do projeto apresenta a solução como foi entregue ao fim do projeto e traz a concretização das etapas do processo de desenvolvimento, e a aplicação de técnicas e ferramentas tecnológicas no desenvolvimento do Design Quips. A conclusão traz os reflexões sobre os resultados obtidos pelo projeto e sugere temas que poderiam dar continuidade a esse trabalho.

2. PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Esse capítulo visa explicar, de forma sucinta, o processo e tecnologias utilizados no desenvolvimento do Design Quips. O conhecimento de tais assuntos é necessário para que se possa compreender o desenvolvimento do projeto. A seção de “Processo de Desenvolvimento” trata do conceito de processos ágeis e suas principais características, e aborda as práticas e etapas do processo utilizado para desenvolvimento de software na ZURB. A seção de “Tecnologias Utilizadas” trata principalmente do *framework* Ruby on Rails, plataforma na qual o projeto foi desenvolvido, e do Foundation Framework, *framework* usado para o desenvolvimento do front-end do Design Quips.

2.1. Processo de Desenvolvimento

Processos de software definem um conjunto distinto de atividades, ações, tarefas, marcos e produtos de trabalho que são necessários para fazer engenharia de software (PRESSMAN, 2006). Tais processos podem ter uma definição formal de seus componentes; ou serem informais e surgirem naturalmente a medida que um projeto avança. Não existe projeto de software “sem processo” — mesmo que não se siga um processo definido, o conjunto de etapas e as práticas seguidas num projeto constituem um processo.

O processo utilizado no desenvolvimento do Design Quips é um processo ágil mas adequado a realidade centrada em design da ZURB. Para entender esse processo é necessário discutir as origens e princípios dos processos ágeis bem como a filosofia e características da empresa ZURB e como elas influenciaram no processo resultante.

2.1.1. Processos Ágeis

Desenvolvimento Ágil é um movimento na área de engenharia de software que surgiu com o intuito de reavaliar as práticas e métodos sendo utilizadas na indústria, para aumentar a produtividade e as chances de sucesso de projetos de software. Os ideais desse movimento estão presentes no Manifesto de Desenvolvimento Ágil de Software e na lista de Princípios do Software Ágil, documentos escritos em 2001 numa reunião de experientes desenvolvedores e líderes de times de desenvolvimento. O Manifesto diz:

> “Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
> - Indivíduos e interações mais que processos e ferramentas;
> - Software em funcionamento mais que documentação abrangente;
> - Colaboração com o cliente mais que negociação de contratos;
> - Responder a mudanças mais que seguir um plano.
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.” (BECK et al., 2001)

Os Princípios do Software Ágil transformam a filosofia do Manifesto Ágil em 4 itens mais concretos: (i) “Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado”, (ii) “Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo”, (iii) “O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face”, e (iv) “Software funcionando é a medida primária de progresso” (BECK et al., 2011).

Processos Ágeis são uma implementação concreta da filosofia ágil, baseando-se em seus valores e princípios. XP e SCRUM são os mais conhecidos exemplos de processos ágeis, e tem em comum entregas constantes, espaço para mudanças no escopo, e participação dos interessados durante todo o desenvolvimento. Tais processos se opões claramente a processos mais tradicionais baseados no modelo de cascata, que se baseiam numa documentação extensiva de todo o software nas etapas iniciais do projeto, fases sequenciais e bem definidas, e uma falta de acompanhamento do projeto pelos seus interessados durante a fase de desenvolvimento (TELES, 2004).

2.1.2. Processo de Desenvolvimento ZURB

Para se compreender o processo de desenvolvimento ZURB é preciso primeiro conhecer a empresa em si. Uma discussão de sua história, atuação e valores é necessária para que fiquem claras as motivações e raciocínios por trás das práticas adotadas.

2.1.2.1. Empresa ZURB

A ZURB se define como “um time unido que designers de produto que ajudam empresas a desenharem melhores sites, serviços e produtos online”. Eles atuam em quatro áreas distintas relacionadas ao design de produtos web: consultoria, software livre, aplicativos web, e educação (ZURB, 2014).

Consultoria é o foco da empresa, que oferece serviços de estratégia de design, design de interação, e design de interface voltados principalmente para *startups* de tecnologia, mas que também serve a clientes como Facebook, Yahoo, McAfee, e Netflix.

Sua atuação na área de software livre consiste principalmente em desenvolver e manter o Foundation Framework, um *framework* responsivo para o desenvolvimento do *front-end* de aplicações web. O Foundation é um dos 20 projetos mais populares na ferramenta de colaboração GitHub (GITHUB, 2014).

Quanto a aplicativos web, a empresa possui uma suite voltada para agências e departamentos de design. A suite inclui as seguintes ferramentas: (i) Influence, uma ferramenta para a criação de apresentações; (ii) Verify, que permite criar pesquisas para coletar e analisar dados sobre telas ou páginas web; (iii) Solifidy, ferramenta para criação de protótipos clicáveis para a validação de fluxos de usuário; e (iv) Notable, ferramenta que permite compartilhar notas e comentários sobre detalhes específicos em telas, rascunhos, ou desenhos. Além de vendidas para o público externo, os funcionários da ZURB utilizam essas ferramentas extensivamente em seus dia a dia de trabalho.

Por fim, a parte focada em educação da empresa é responsável por treinamentos e documentos de referência para *designers.* Os treinamentos são online e voltados para empresas com foco em design web, uso do Foundation Framework, e outros tópicos relacionados a design de produtos. Os documentos de referência são um conjunto de sites que inclui propriedades como o Word, um glossário de termos relacionados a design de produtos, e o Triggers, uma coleção de padrões de comportamento, motivadores psicológicos e influências cognitivas.

Os valores da ZURB são descritos resumidamente nos parágrafos a seguir, que tratam da aplicação de colaboração, comunicação, autonomia e produtividade no modo de trabalho da empresa.

O foco em design na atuação da empresa se reflete também no seu quadro de funcionários. Cerca de dois terços do dos funcionários são *designers* e líderes de *design*, com o terço restante composto por auxiliares administrativos, escritores, e desenvolvedores, representantes comerciais e um líder de desenvolvimento. Outra característica do quadro de funcionários é que não existem gerentes, existindo pouca hierarquia que é representada pelo *Chief Executive Officer* (CEO), o sócio majoritário e superior de todos os funcionários da empresa. Designers trabalham diretamente com os clientes e são responsáveis pela maioria dos projetos, com o suporte de um líder de design e do líder de desenvolvimento, e respondem diretamente ao CEO. A empresa ainda é composta, além de seus funcionários, por alguns (normalmente 3 ou 4) estagiários.

Os contratos dos estagiários usualmente são para as funções de *designer* ou desenvolvedor por períodos de três a seis meses. Cada estagiário recebe um projeto específico, que se torna seu foco principal durante o seu contrato, e para a realização de suas tarefas, ele trabalha em conjunto com o restante do time. Além de seu projeto os estagiários podem ainda se envolver com trabalhos para clientes e outros projetos internos da empresa.

A estrutura horizontal ora apresentada é uma reflexão de um dos principais valores adotados na ZURB, a colaboração. Trabalhar junto é considerado essencial, e muitas das práticas adotadas no processo de trabalho existem para possibilitar e incentivar a colaboração tanto de *designers* entre si quanto entre *designers* e desenvolvedores ou escritores.

Comunicação é um dos outros valores fundamentais para a empresa. Projetos são sempre desenvolvidos num ambiente aberto, onde os clientes tem acesso constante e freqüente ao trabalho dos *designers*. Até os dados financeiros da empresa são compartilhados com os funcionários.

Um terceiro valor que é parte integral do trabalho da ZURB é a autonomia. *Designers* com mais experiência são promovidos a líderes de *design.* Um líder, seja de designer ou desenvolvimento, tem como principal responsabilidade orientar e auxiliar *designers* e desenvolvedores mais do que propriamente controlar e gerenciar o trabalho de um time. *Designers* e desenvolvedores são responsáveis por suas decisões, e tem autonomia e liberdade em como tocar seus projetos.

Produtividade é um outro valor central à forma que se trabalha na ZURB. Existe um foco em lançar projetos e mostrar progresso concreto, originado na ideia de que acompanhar o uso real de um projeto é a forma mais eficiente de aprender sobre ele e poder melhorá-lo. As práticas e ferramentas que permitem lançar um projeto mais rápido são mais valorizadas que aquelas que permitem descrever ou analisar melhor um projeto antes de seu lançamento.

A forma que esses valores — colaboração, comunicação, autonomia e produtividade — influenciam o dia a dia da ZURB ficará mais claro ao serem discutidas as práticas adotadas no processo de desenvolvimento.

Figura 1. Ambiente principal de trabalho

Fonte: Arquivo da empresa, 2013

Um outro fator que influencia o processo de desenvolvimento é a estrutura física da empresa. O escritório é organizado num formato de "*open-plan*", em uma tradução livre plano aberto. Não existem divisórias entre os funcionários ou escritórios individuais. Todos, incluindo o CEO, trabalham em conjuntos de quatro mesas espalhadas ao longo do segundo andar do prédio, como mostrado na Figura 1.

Escritórios de plano aberto são um assunto controverso e pesquisas apontam para uma queda na produtividade em espaços deste tipo, citando fatores como perda de privacidade e aumento nas interrupções (HEDGE, 1982). Tais fatores são balanceados na ZURB pela existência de espaços individuais usados quando funcionários que precisam de mais concentração, como sofás confortáveis e espaços em corredores; e salas a prova de som usadas para pequenas reuniões entre times ou com clientes. O escritório conta ainda com uma mesa de reuniões (Figura 2) que acomoda todos os funcionários, um terraço, e uma cozinha completa. Como é típico em escritórios de tecnologia na região, a cozinha está sempre estocada com diversos lanches e bebidas.

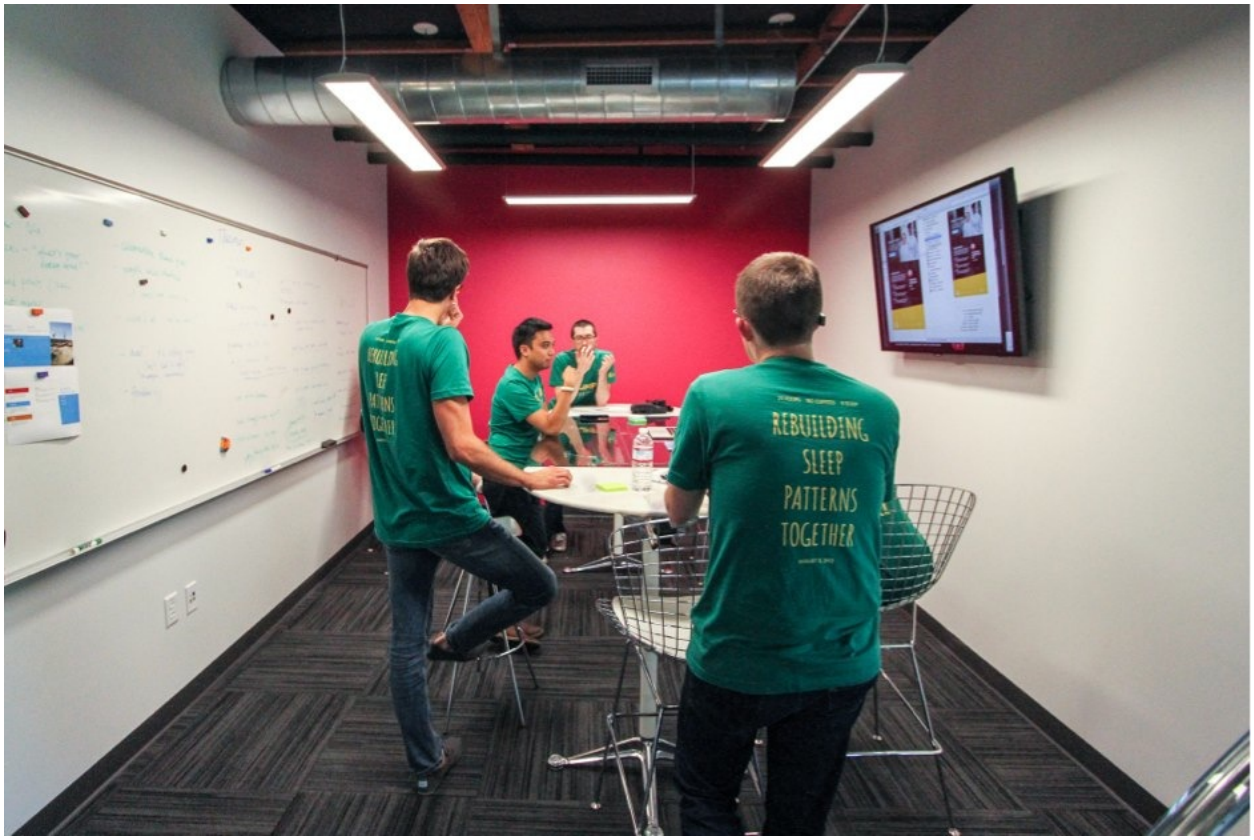

Figura 2. Sala de reunião e trabalho individual

Fonte: Arquivo da empresa, 2013

2.1.2.2. Práticas do Processo de Desenvolvimento de Software da ZURB

As práticas discutidas aqui são descritas da forma que aconteceram no projeto Design Quips, mas em sua maior parte se aplicam a todos os projetos de desenvolvimento da empresa. As práticas mais características do processo são: (i) times multi-disciplinares, (ii) ritmo sustentável, (iii) integração contínua e pequenas

entregas, (iv) equipe integral, (v) propriedade coletiva do código-fonte, (vi) pequenas reuniões diárias, (vii) retrospectivas, e (viii) práticas relacionadas ao convívio na empresa. As ideias por trás das práticas e algumas das práticas em si se aplicam não só a projetos de desenvolvimento de software mas a todas as áreas de atuação da empresa, e envolvem toda a equipe (CEO, *designers,* desenvolvedores, editores). Tais práticas estão descritas resumidamente nos parágrafos abaixo.

Times multi-disciplinares são utilizados em praticamente todos os projetos. Projetos de desenvolvimento contam sempre com, além um desenvolvedor, um *designer* e um editor. Todos trabalham juntos e se comunicam com freqüência. A motivação por trás da decisão de montar times multi-disciplinares é explicada por Vegt e Bunderson:

> “A premissa em que se baseia a motivação ao uso dessas equipes é que, quando representantes de todas as áreas relevantes do conhecimento se reúnem, as decisões e as ações da equipe são mais propensas a abranger toda a gama de perspectivas e problemas que podem afetar o sucesso de um projeto coletiva. Equipes multi-disciplinares são, portanto, uma opção de organização atraente quando indivíduos possuem diferentes informações, conhecimento e experiência que relevantes à um problema complexo.” (VEGT; BUNDERSON, 2005)

Ritmo sustentável é uma das práticas adotadas baseadas no XP. A maioria dos funcionários trabalha num ritmo de 40 horas por semana, e não existe pressão social para trabalhar excessivamente. Exceções acontecem quando prazos de entrega importante se aproximam, e são toleradas apenas por tempo limitado. Exemplo desta situação aconteceu durante o estágio: um funcionário recém-contratado se sentiu sob muita pressão e passou a trabalhar cerca de 10 horas por dia e foi aconselhado a diminuir um pouco suas responsabilidades nos projetos que estava envolvido, bem como a pedir mais ajuda aos seus colegas ao invés de trabalhar em excesso.

Integração contínua e pequenas entregas (*releases*) são duas práticas que se complementam. O código-fonte é compartilhado entre o time ou colocado em produção diversas vezes ao dia, e até funcionalidades em desenvolvimento são disponibilizadas nos repositórios de código-fonte dos projetos para que todos os membros do time tenham acesso.

A prática de equipe integral, pela qual o time de um projeto deve conter um usuário final disponível para responder questões do time, acontece naturalmente já que os projetos lá desenvolvidos são voltados a *designers* ou empresas de *design*.

A prática de propriedade coletiva do código-fonte também vem do XP. Desta forma, todos os desenvolvedores são responsáveis por todas as partes do código-fonte. Não existem partes que são propriedade de um desenvolvedor em particular, e não é preciso pedir permissão antes de alterar algo no projeto. Até o código de *front-end*, feito por e de responsabilidade dos *designers*, pertence a todos no projeto e é alterado por desenvolvedores.

Pequenas reuniões diárias marcam o início de cada expediente, envolvem toda a empresa, e seguem um formato similar ao *Stand Up Meeting* do XP ou ao *Daily Scrum* do Scrum. Cada reunião tem duração aproximada de 15 minutos e acontece periodicamente todo dia antes que se inicie o expediente. A princípio cada funcionário tinha alguns segundos para dizer o que seria seu foco naquele dia de trabalho e que tipo de ajuda precisaria. Quando a equipe esteve muito grande para reuniões desse tipo, mudou-se o formato: o CEO e os líderes de time faziam anúncios relevantes a todos, e depois cada funcionário conversava com três pessoas com quem fosse colaborar naquele dia.

Além da reunião diária, toda as segundas-feiras os desenvolvedores se reuniam para uma retrospectiva da semana anterior e planejamento de alto nível da semana seguinte. Nessa reunião discutiam-se metas alcançadas na última semana, empecilhos ao desenvolvimento de projetos e possíveis soluções, e compartilhavam-se técnicas ou bibliotecas que pudessem servir a todos. *Designers* também faziam uma reunião semanal semelhante.

As práticas relacionadas ao convívio na empresa tinham por objetivo principal fortalecer laços entre os funcionários e estimulá-los criativamente. Defendia-se na empresa que tais práticas aumentavam a produtividade do time a longo prazo. Entre essas práticas sociais, se destacam pequenos exercícios chamados "*Friday 15*," nome que se deve ao fato de que tais exercícios duravam cerca de 15 minutos e aconteciam toda sexta-feira. O conteúdo dos exercícios variava, mas eram quase sempre atividades em grupo com um teor criativo. Como exemplo, houveram competições de avião de papel, desafios de CSS, e construção coletiva de haikus (ZURB, 2013). Todos participam dos exercícios, até mesmo escritores e contadores. Outras práticas sociais

orquestradas pela empresa incluem almoços servidos na empresa três vezes por semana, e atividades sociais que aconteciam em datas comemorativas como um churrasco na semana da independência americana.

2.1.2.3. Etapas do desenvolvimento de um projeto

Um projeto de desenvolvimento na ZURB não segue um processo definido com fases bem definidas e separadas. Ainda assim, é possível perceber diferentes etapas em um projeto, que possuem ritmos próprios e onde conjuntos diferentes de práticas e tarefas são adotadas: concepção, desenvolvimento, lançamento e acompanhamento (Tabela 1). Estas etapas aplicam-se tanto na criação de novos projetos e quanto no lançamento de grandes mudanças em projetos já existentes. Ainda é importante destacar uma peculiaridade do processo da ZURB, não existem "unidades de trabalho" como as iterações do XP ou os *Sprints* do SCRUM. O desenvolvimento é contínuo, e avançam-se etapas de desenvolvimento de acordo com a situação do projeto ao invés de seguindo prazos bem definidos. Os parágrafos seguintes descrevem resumidamente cada etapa.

Tabela 1. Etapas do Processo de Desenvolvimento

Etapa	Envolvidos	Objetivo	Período Médio
Concepção	Idealizador do projeto, *designer*	Gerar um plano inicial do projeto	Uma semana
Desenvolvimento	*Designers,* desenvolvedores	Desenvolver o projeto	Dois meses
Lançamento	*Designers,* desenvolvedores, editor e CEO	Lançar o projeto publicamente	Uma semana
Acompanhamento	Todo o time	Acompanhar e revisar o projeto	Indefinido

A primeira etapa é a de concepção, quando acontece um planejamento de alto nível do projeto. Nessa etapa o idealizador do projeto e designers (um ou dois) discutem e fazem sessões de *brainstorming* para gerar um plano do projeto. Esse plano possibilita o início do desenvolvimento, mas não tem o fim de descrever o projeto de forma detalhada ou fixa. O foco é descrever apenas características essenciais do projeto a ser desenvolvido, com descrições gerais e de alto nível das possíveis

funcionalidades, e pode ser abandonado a qualquer momento. Não há preocupação em manter os planos atualizados ou modificá-los para refletir a realidade do projeto. Usualmente esse plano consiste de esboços de tela feitos em papel. Em projetos mais complexos se desenvolvem esboços em papel e protótipos HTML de algumas das funcionalidades mais importantes. A concepção é a etapa mais curta do projeto, com duração média de uma semana. Apesar de em alguns casos não ter mais que uma sessão de *brainstorming* de duas horas.

Após a concepção inicia o desenvolvimento, fase mais longa de qualquer projeto dentro da ZURB. Essa fase geralmente é liderada pelo *designer* líder do projeto, que prioriza tarefas e cria protótipos de interface. Com estas informações os desenvolvedores implementam funcionalidades (código-fonte). Todos do time interagem com freqüência para discutirem e aperfeiçoarem detalhes das tarefas, e todos estão sempre atualizados no andamento do projeto através das reuniões diárias e semanais. Quando alguma dificuldade é encontrada pede-se ajuda aos líderes de *design* e de engenharia. Quando marcos importantes são atingidos, o time reúne-se com um líder de *design* para apresentar o andamento do projeto e solicitar críticas. Uma vez que as funcionalidades foram desenvolvidas completamente e aperfeiçoadas levando em consideração as críticas de um líder de *design*, considera-se completa a etapa de desenvolvimento e começa a etapa de lançamento.

O início da etapa de lançamento é marcado por uma reunião com o CEO. Nessa reunião o CEO avalia o projeto e apresenta suas críticas, e um cronograma inicial de lançamento é elaborado. O cronograma consiste de datas de lançamento interno, revisão pelo CEO, e lançamento externo. Outros funcionários são adicionados ao projeto conforme necessário para lidar com tarefas relacionadas ao lançamento propriamente dito: um desenvolvedor extra pode ser alocado ao projeto para preparar a infra-estrutura necessária. Por exemplo, um editor para elaborar os textos do site e preparar o anúncio do lançamento. O foco do desenvolvimento passa a ser melhorar a experiência e corrigir possíveis erros de implementação. Na data de lançamento interno o projeto é anunciado para todos os funcionários, que usam e oferecem críticas sobre o projeto.

Pode haver uma ou mais reuniões adicionais com o CEO para acompanhar os ajustes feitos a partir de suas críticas iniciais e dos comentários dos outros funcionários, bem como do andamento das tarefas de lançamento externo. Quando

todas as pendências são resolvidas, o projeto é lançado oficialmente com um anúncio no blog corporativo da empresa.

Depois do lançamento tem início a etapa de acompanhamento, quando acontece do time ser realocado para outros projetos. Apesar desta realização, os desenvolvedores ficam com a responsabilidade do contínuo funcionamento do projeto, e de atualizar o projeto quando houverem mudanças; os escritores iniciam a produzir com regularidade conteúdo para o projeto, se for o caso; e os *designers* podem revisitar o projeto para ajustarem funcionalidades ou fazerem melhoras visuais. Esse processo de acompanhamento e melhoria contínuo não tem data definida para terminar. De certa forma, a etapa continua enquanto o projeto existir como uma propriedade da ZURB.

2.2. Tecnologias Utilizadas

A ZURB prefere utilizar tecnologias que possibilitem fazer protótipos rápidos e facilitem a interação entre *designers* e desenvolvedores. A possibilidade de fazer protótipos rápidos é necessária devido a sua cultura que prioriza colocar software em produção pra melhorá-lo com base no uso real e fazer pequenos experimentos. Além disso, tecnologias que permitem uma prototipação rápida são consideradas mais produtivas, e portanto mais bem aceitas pela ZURB. Facilitar a interação entre designers e desenvolvedores é o outro fator na escolha de uma tecnologia, porque a empresa possui muito mais designers e desenvolvedores, e ambos colaboram frequentemente nos projetos. É preciso que a tecnologia torne possível que designers façam mudanças no projeto sem necessitar da ajuda de desenvolvedores, e que a curva de aprendizado seja pequena o suficiente de forma a permitir designers a compreenderem a organização e estrutura do projeto.

Das tecnologias utilizadas na ZURB, duas são comuns a todos os projetos de desenvolvimento e constituem seu suporte principal: o *framework* de desenvolvimento de aplicações web Ruby on Rails, e o framework de *front-end* responsivo Foundation Framework.

2.2.1. Ruby on Rails

Ruby é uma linguagem dinâmica, código-aberto e tem seu foco em simplicidade e produtividade. Ainda em seu site, é definida como uma linguagem de sintaxe

elegante, leitura natural e fácil escrita. Foi inicialmente desenvolvida pelo desenvolvedor japonês Yukihiro Matsumoto, que até hoje é o principal responsável pela linguagem e chefe de desenvolvimento do MRI (*Matz's Ruby Interpreter*), interpretador Ruby oficial. Foi lançada publicamente em 1995, e é mantida hoje no site http://ruby-lang.org (LINGUAGEM…, 2014).

O Rails é um dentre diversos *frameworks* para desenvolvimento de aplicativos web com a linguagem Ruby. Segundo definição em seu site oficial, Ruby on Rails é um *framework* de desenvolvimento web (gratuito e de código aberto) otimizado para a produtividade sustentável e a diversão do programador, e que permite que se escreva código de forma elegante, favorecendo a convenção ao invés da configuração (MONTEIRO, 2010). Foi desenvolvido pela Basecamp Inc. (anteriormente 37Signals Inc.), em Chicago, EUA, para um aplicativo de gerenciamento de projetos colaborativos chamado Basecamp. Rails foi lançado como um projeto de código aberto em Julho de 2004, e a versão 1.0 veio em Dezembro de 2005. Rails é escrito em Ruby e segue os princípios de desenvolvimento ágil, melhoria da produtividade e aceleração do desenvolvimento (PLEKHANOVA, 2011).

Projetos desenvolvidos utilizando Ruby on Rails seguem uma arquitetura Modelo-Visão-Controle (MVC, do inglês Model-View-Controller). MVC é um padrão de projeto introduzido no Smalltalk-80, e apresentado por Krasner e Pope (1988) no artigo “A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System” (“Uma descrição do paradigma de interface de usuário Modelo-Visão-Controle no sistema Smalltalk-80,” em tradução livre). Segundo eles, o modelo de uma aplicação representa a parte de domínio em um software específico; a visão lida com tudo que é interface com o usuário e por conseguinte com a representação dos dados da aplicação; e controle que traduz toda a interligação entre o modelo e a visão, além de definir o controle de navegação entre as diversas interfaces com o usuário de uma aplicação. No Rails, o funcionamento de forma resumida é mostrado na Figura 3, onde os modelos representam as tabelas do banco de dados e implementam o padrão arquitetural *Active Record*[1] — ou seja, cada objeto representa uma linha do banco e é responsável por sua própria persistência (FOWLER, 2002); a visão é composta por

[1] No Rails esse padrão é implementado através da biblioteca ActiveRecord, componente ORM (Mapeador Objeto-Relacional, do inglês *Object-Relational Mapper)* que implementa, de forma simplificada e resumida, código-fonte para traduzir os dados de um sistema gerenciador de banco de dados relacional para objetos, e vice-versa.

pseudo código-fonte HTML que permite inserção de trechos de código-fonte Ruby, sendo responsável por gerar o HTML final da interface com o usuário; e o controle é o responsável por tratar as requisições das interfaces com o usuário, ora recuperando objetos do modelo e com eles alimentando a visão, ora recuperando dados da visão e atualizando os objetos do modelo.

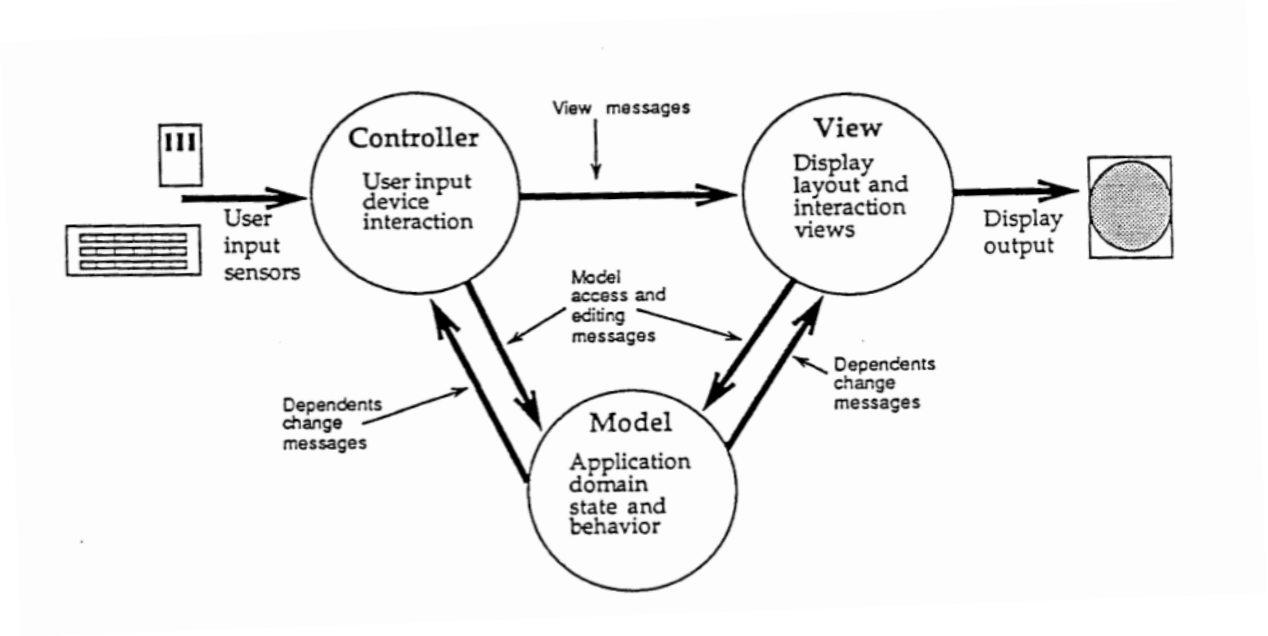

Figura 3. Diagrama de funcionamento do MVC clássico

Fonte: (KRASNER; POPE, 1988)

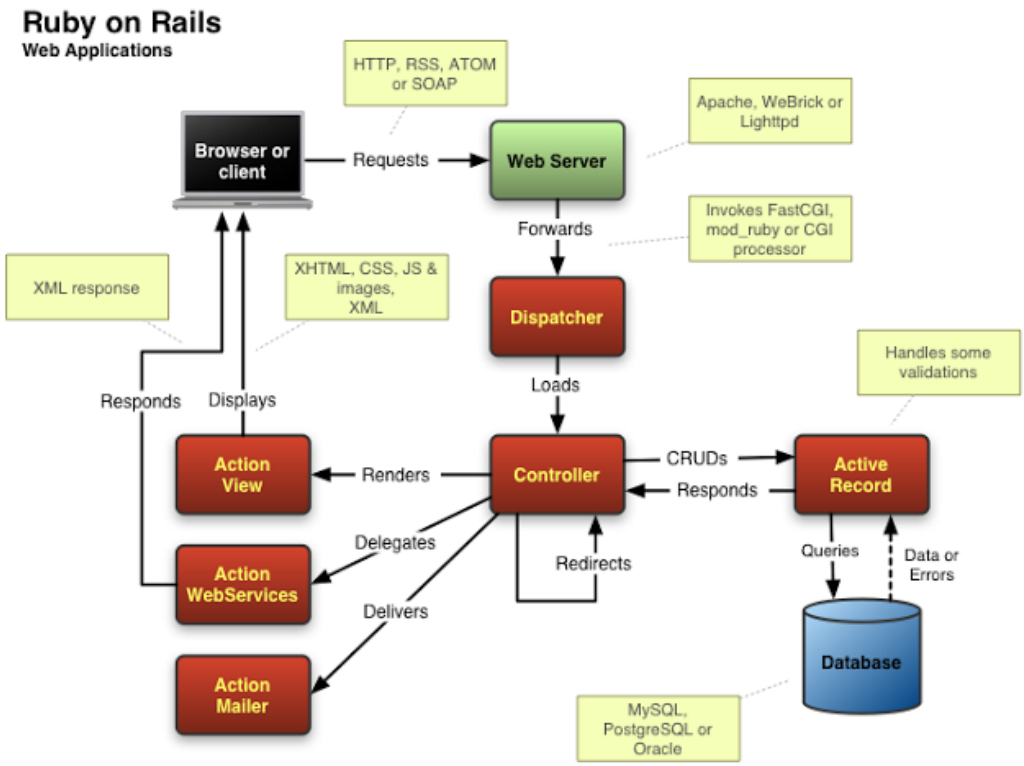

Figura 4. Diagrama de funcionamento do MVC como implementado no Ruby on Rails

Fonte: (NIWATORI…, 2007)

Além do MVC, dois outros princípios são centrais a arquitetura de aplicações Rails: convenção sobre configuração e DRY (do inglês *Don't Repeat Yourself,* com tradução livre para português "não se repita"). Convenção sobre configuração é a ideia de que deve-se seguir padrões do *framework* para decisões comuns ao invés de ter que configurá-las. Esse princípio se manifesta, por exemplo, na convenção de que toda tabela deve ter uma chave primária chamada "id," e que portanto não é necessário configurar uma; e na convenção que o conjunto de visões de cada controlador encontra-se numa pasta padrão, e que portanto não é necessário configurar onde encontrar cada visão. É possível não usar as convenções do Rails, contudo seguí-las resulta em menos código-fonte, mais consistência, e mais produtividade. DRY é o princípio de que um conceito deve ser implementado em um e apenas um lugar. Por exemplo, os atributos de um modelo são definidos apenas como campos de suas tabelas no banco — não é preciso repetir a definição no código-fonte do modelo. Para auxiliar o programador a não ter que repetir, o Rails fornece ainda uma estrutura que facilita compartilhar código-fonte entre várias partes do sistema, com cada parte do

sistema contando com pacotes específicos onde o código-fonte comum pode ser adicionado e que são automaticamente injetados em todos os componentes.

Parte da produtividade promovida pelo Ruby on Rails vem desses princípios do MVC e organização citados, da automatização de tarefas comuns, e de componentes e bibliotecas reusáveis trazidas pelo *framework*. Outra parte vem de uma gama de bibliotecas de código-fonte livre que complementam suas funcionalidades e facilitam o desenvolvimento de novos projetos. Existem bibliotecas para automatização de instalação, integração com serviços de terceiros, autenticação, autorização, e vários outros funcionalidades comuns à aplicações web. Uma busca por projetos Ruby no repositórios de projetos de código livre GitHub resulta em mais de 500 mil projetos, um indicativo de quão ativa é a comunidade do Ruby on Rails (GITHUB, 2014).

2.2.2. Foundation Framework

Foundation é um *framework* para o desenvolvimento de *front-end* de aplicações web. Segundo o site oficial (http://foundation.zurb.com), Foundation teve suas origens em 2008 nos guias de estilo da ZURB, que eram usados em todo projeto. Considerando que poderiam avançar com os guias, a ZURB decidiu que precisava de um *framework* que permitisse fazer protótipos rapidamente. A partir de seus CSS globais, *plugins* jQuery, elementos comuns e melhores práticas, criou o Foundation, cuja primeira versão foi lançada em 2011 (ZURB, 2014).

Foundation fornece, entre outras coisa, um sistema de grade, navegação, botões e *plugins*. O sistema de grade divide o layout da página em 12 colunas, e permite controlar o posicionamento de elementos através de diretivas de tamanho e posicionamento em relação a grade. A grade do Foundation é responsiva e funciona em praticamente qualquer dispositivo que possa acessar a internet, além de suportar aninhamento, deslocamentos, apresentações específicas para determinados dispositivos, entre outras funcionalidades.

Para navegação entre interfaces com o usuário, o Foundation fornece componentes de menu *off-canvas*[2] (Figura 5), paginação, menus globais e menus locais. Os botões do Foundation são versáteis, com tamanho, prioridade, e estado

[2] *off-canvas* é um estilo de menu popular em aplicativos móveis onde o menu parece estar escondido fora da tela, e aparece quando se clica em um botão específico.

customizáveis. Os *plugins* incluídos fornecem funcionalidades como validação de formulários, galeria de imagens, e exibição rotativa de imagens.

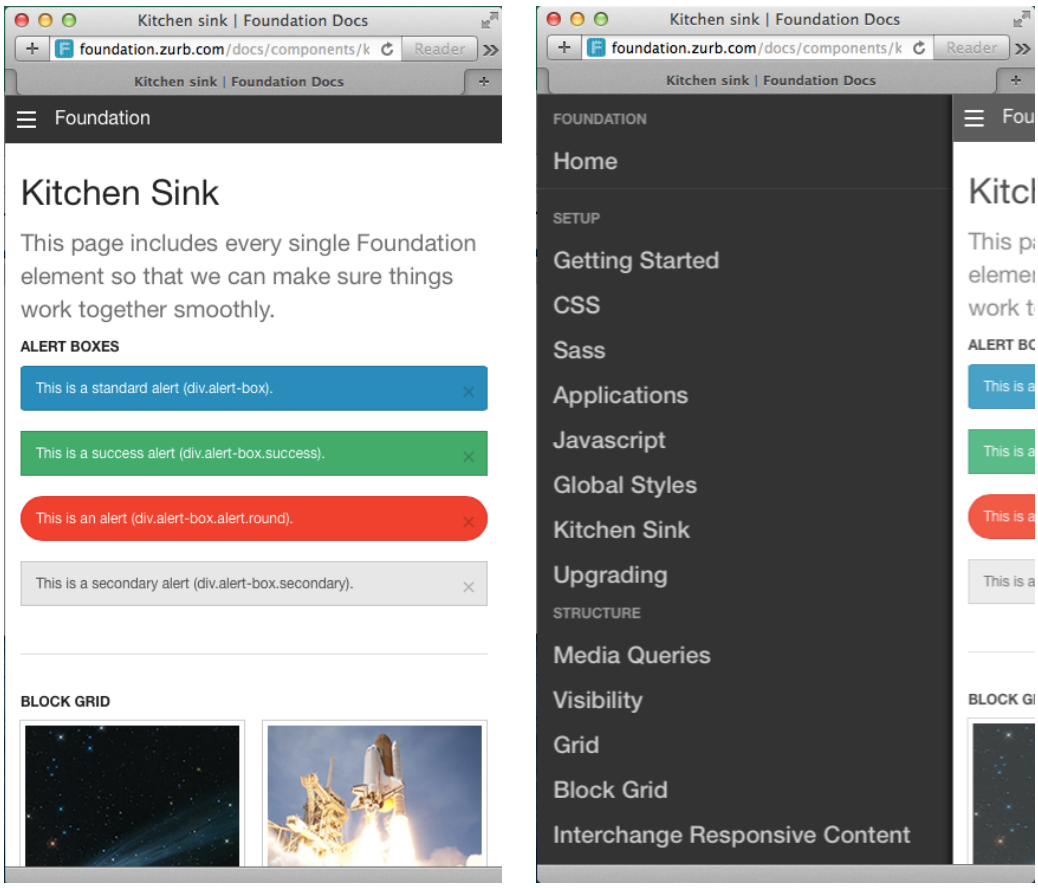

Figura 5. Menu off-canvas no estado inicial e no estado visível

Fonte: Foundation Kitchen Sink, 2014

A prática de desenvolvimento de front-end defendida pelo Foundation é o design responsivo. Esse termo foi cunhado por Ethan Marcotte (2010) num texto na publicação *A List Apart*, quando juntou três técnicas existentes — layout de grade flexível, imagens flexíveis, e *media queries* do CSS — para criar layouts que não assumem uma resolução fixa e ao invés disso se adaptam ao tamanho da tela do dispositivo cliente. O resultado do uso dessas técnicas pode ser visto na Figura 6. Essa abordagem surgiu como uma resposta ao crescente número de diferentes dispositivos capazes de acessar a internet, como smartphones e consoles de vídeo game. Apenas para ressaltar o crescente número de dispositivos móveis que usam a internet, entre março de 2009 e maio de 2013 foi contabilizado o tráfego oriundo de dispositivos móveis correspondeu a 15% do tráfego global da internet (MEEKER; WU, 2013).

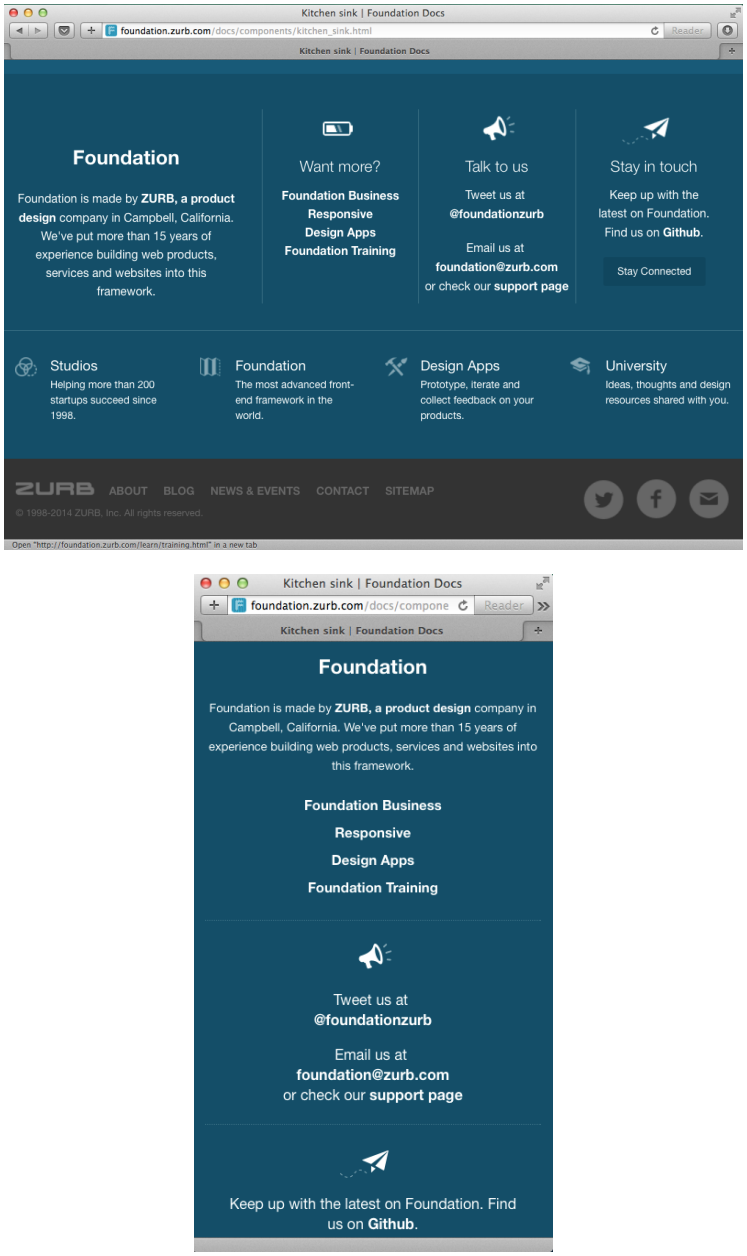

Figura 6. Design respondendo a uma mudança na resolução do dispositivo

Fonte: Foundation Kitchen Sink, 2014

De forma resumida, o Foundation permite o desenvolvimento de *front-end* de uma aplicação através de uma estrutura de layout baseado numa grade responsiva e um conjunto de componentes reusáveis.

3. DESENVOLVIMENTO DO DESIGN QUIPS

Uma vez apresentados o processo e as tecnologias utilizadas durante o projeto Design Quips, apresentam-se agora o resultado obtido e a aplicação concreta do processo. A apresentação do resultado traz os objetivos visados com a criação do Design Quips e como sua implementação atinge estes objetivos. A aplicação do processo traz uma descrição detalhada do que aconteceu no desenvolvimento do projeto em cada uma das fases descritas.

3.1. Introdução ao Design Quips

Design Quips surgiu a partir de uma realidade sobre o trabalho de designers percebida pela ZURB: a maior parte de decisões relacionadas a design era tomada com base em intuição, sem suporte concreto. Isso muitas vezes tornava difícil convencer clientes, que não conseguiam ver vantagens concretas ou entender o raciocínio por trás de cada decisão.

O Design Quips é, como a solução imaginada para tal problema, um repositório de fatos e estatísticas sobre tendências web, dados demográficos, uso de dispositivos para acesso a internet, e qualquer outra informação relacionada ao desenvolvimento web. Ele permite que designers busquem facilmente dados que justifiquem suas escolhas. Como repositório de informações atualizadas, serve ainda como fonte de aprendizado sobre o estado da web e sua evolução. Os objetivos específicos do projeto eram:

- Fornecer dados estatísticos sobre demografia de usuários de tecnologia, tendências de mercado, e outros assuntos relacionados ao trabalho de um designer;
- Fornecer ainda referências e fontes para cada dado, possibilitando uma pesquisa mais aprofundada sobre o assunto;
- Apresentar tais dados de forma simples, permitindo uma visualização rápida e tornando a coleta de dados prática para o designer;
- Permitir a busca e filtragem de dados por vários critérios, para que o designer possa encontrar facilmente o subconjunto de dados da coleção relevante para seu trabalho;

- E, por fim, permitir que editores e escritores da ZURB possam adicionar novos dados a coleção.

Os primeiros objetivos são atingidos pela parte pública do sistema. A página inicial lista os dados existentes, e permite filtrar informações relacionadas a uma certa característica demográfica, plataforma, ou dispositivo, e ainda buscar por texto (Figura 7). Tal mecanismo permite de buscas simples, como qualquer informação relacionada a "Facebook," até buscas mais complexas, como apenas informações relacionadas a mulheres jovens, sobre a plataforma iOS, de dispositivos móveis, e que contenham o texto "crescimento." A listagem é atualizada automaticamente quando os filtros são alterados, e ao invés de paginação usa uma espécie de lista infinita: um botão "Ver mais" traz mais resultados enquanto houverem resultados disponíveis. A API Javascript *pushState* permite que essas funcionalidades sejam implementadas sem quebrar o funcionamento do histórico do navegador (PILGRIM, 2010). A página de detalhes de um dado permite ainda ver a origem da informação, onde o usuário pode se aprofundar no que é mencionado, e funcionalidades acessórias como uma uma listagem de dados relacionados e uma lista de categorias (Figura 8).

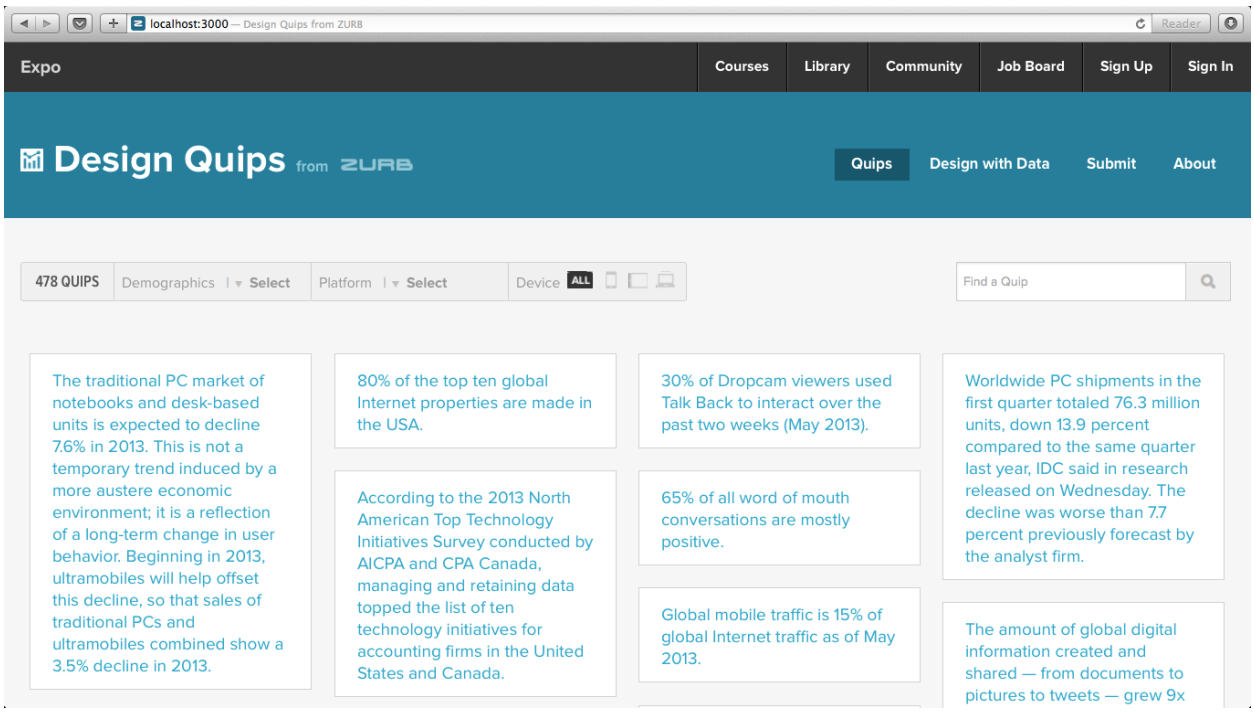

Figura 7. Página inicial

Fonte: Design Quips, Agosto de 2013

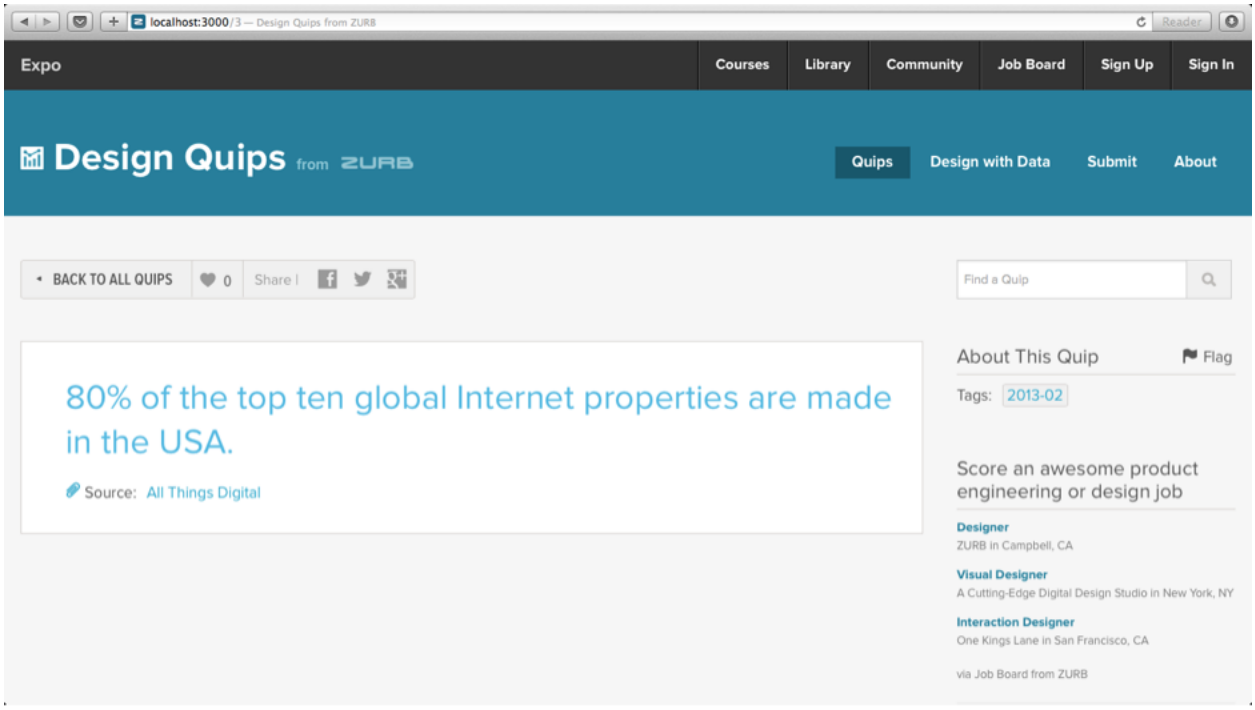

Figura 8. Página de detalhes de um dado

Fonte: Design Quips, Agosto de 2013

O objetivo de permitir que editores e escritores da ZURB possam adicionar novos dados a coleção é atingido pela parte administrativa da aplicação. Para cuidar da autenticação e autorização o Design Quips integra-se ao sistema Platform, um sistema da própria ZURB que funciona como um provedor OAuth e centraliza as credenciais de usuários. O Platform permite ainda autenticação via Facebook, Twitter ou Google+. O componente central da parte administrativa é a listagem de dados (Figura 9). Essa listagem destaca dados que possuem algum problema, como não terem categorias ou não informarem o período a que se referem, facilitando o trabalho dos editores de corrigirem dados falhos. Editores podem ainda ver o uso de cada categoria no sistema, funcionalidade que auxilia o trabalho de consolidar categorias e eliminar as que caíram em desuso (Figura 10). O cadastro de novos dados pode ser feito através da importação de um arquivo CSV contendo os campos conteúdo, período a que o dado se refere, URL da fonte, título da fonte, e lista de categorias (Tabela 2); ou através de um formulário de cadastro (Figura 11). O formulário é otimizado para o fluxo de trabalho comum dos editores, permitindo analisar diversas fontes e cadastrar vários dados de cada sem repetir dados comuns e sem precisar sair da página de cadastro.

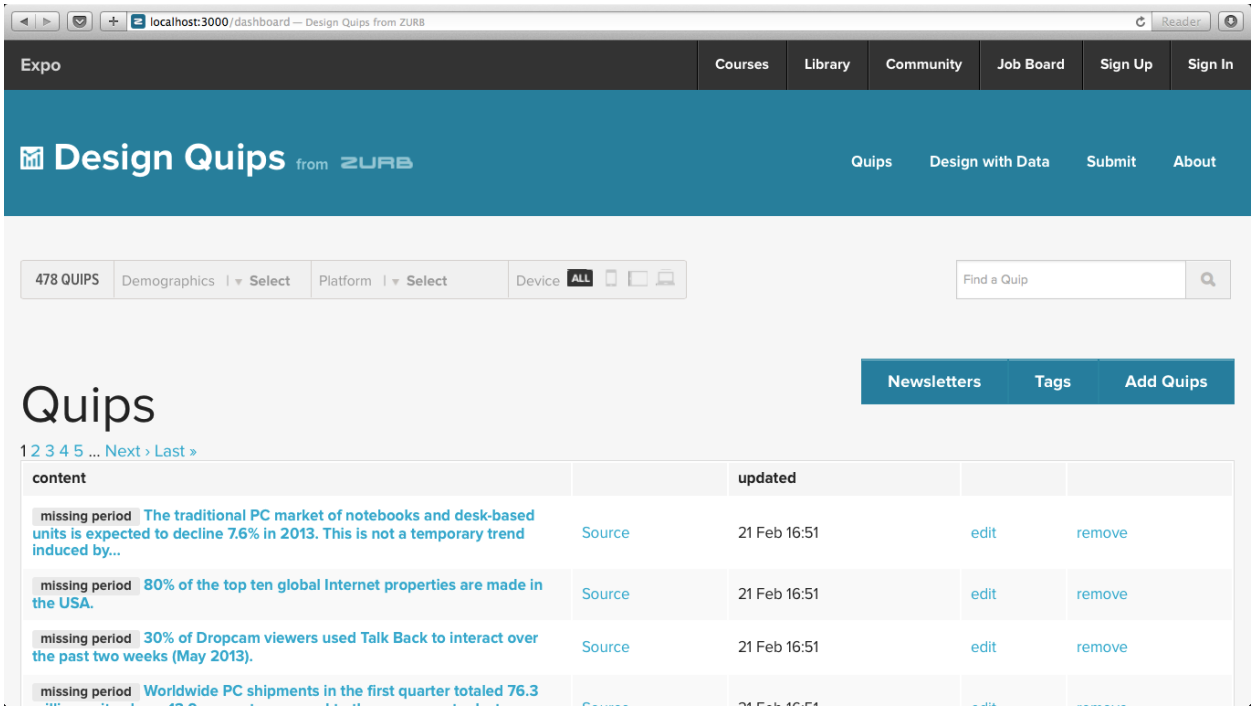

Figura 9. Página inicial para administradores

Fonte: Design Quips, Agosto de 2013

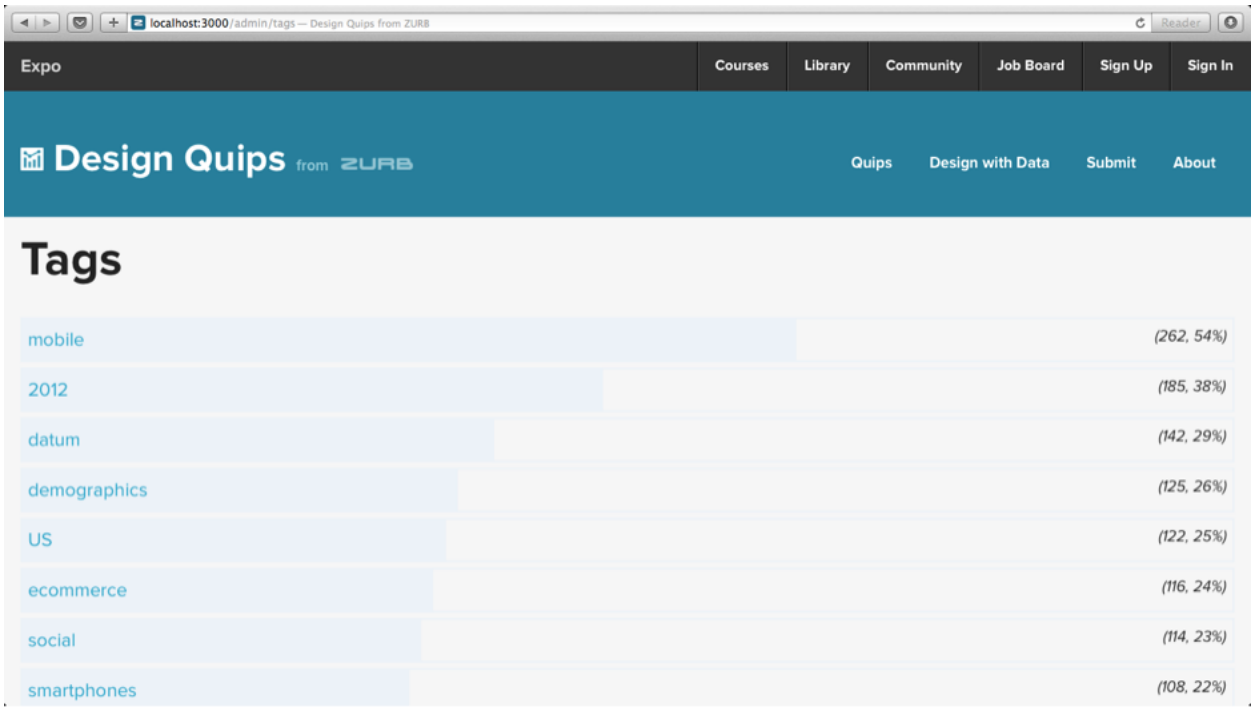

Figura 10. Listagem de categorias

Fonte: Design Quips, Agosto de 2013

Tabela 2. Campos da importação via arquivo CSV

Campo	Descrição	Exemplo
Conteúdo	Conteúdo textual da informação	“Mais que metade de todos os adultos que possuem celular usam seus celulares para acessar a internet”
Período	Intervalo de tempo a que o dado se refere	01/04/2012 - 30/04/2012
URL da fonte	URL do artigo de onde a informação foi retirada	http://bit.ly/1hsDt5V
Título da fonte	Título do artigo de onde a informação foi retirada	“Maioria dos adultos acessa a internet de seus celulares”

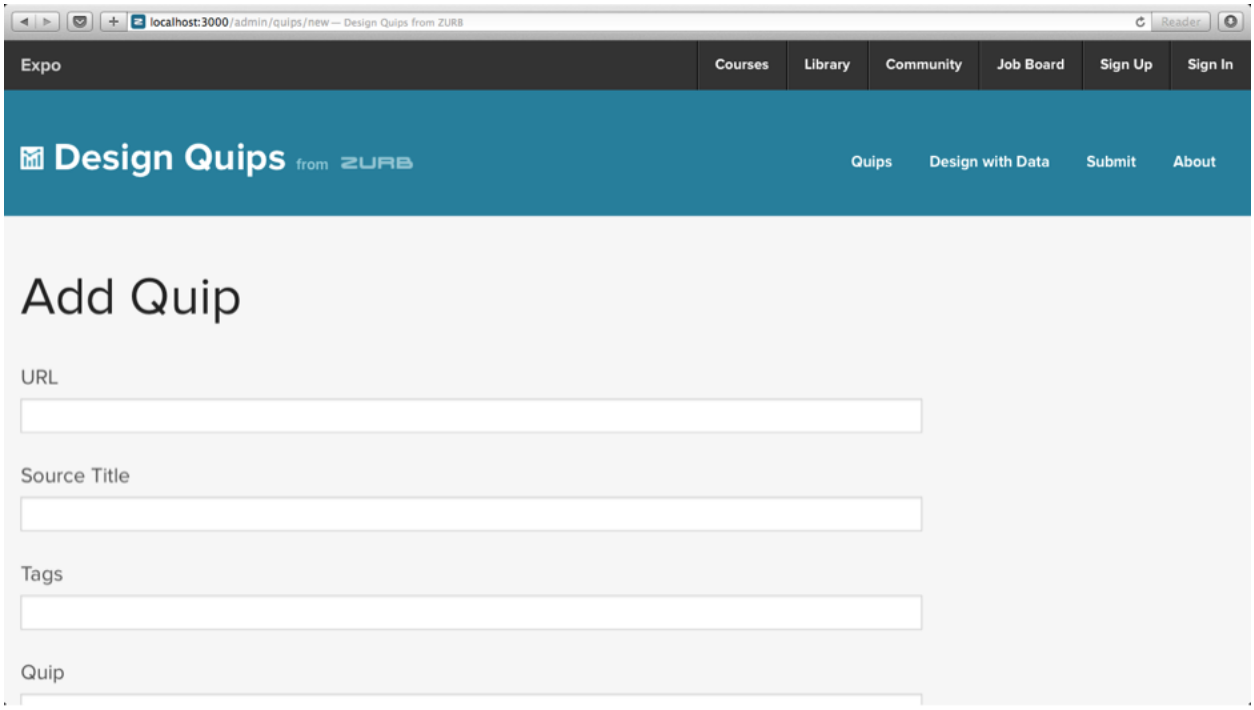

Figura 11. Cadastro de dado individual

Fonte: Design Quips, Agosto de 2013

Além das funcionalidades implementadas para atingir os objetivos principais do projeto, foram implementadas duas funcionalidades complementares presentes em todas as propriedades da ZURB: o cadastro numa lista de email (Figura 12),

gerenciada em outro sistema; e a exibição de propagandas dos classificados de empregos e de parceiros da ZURB (Figura 13).

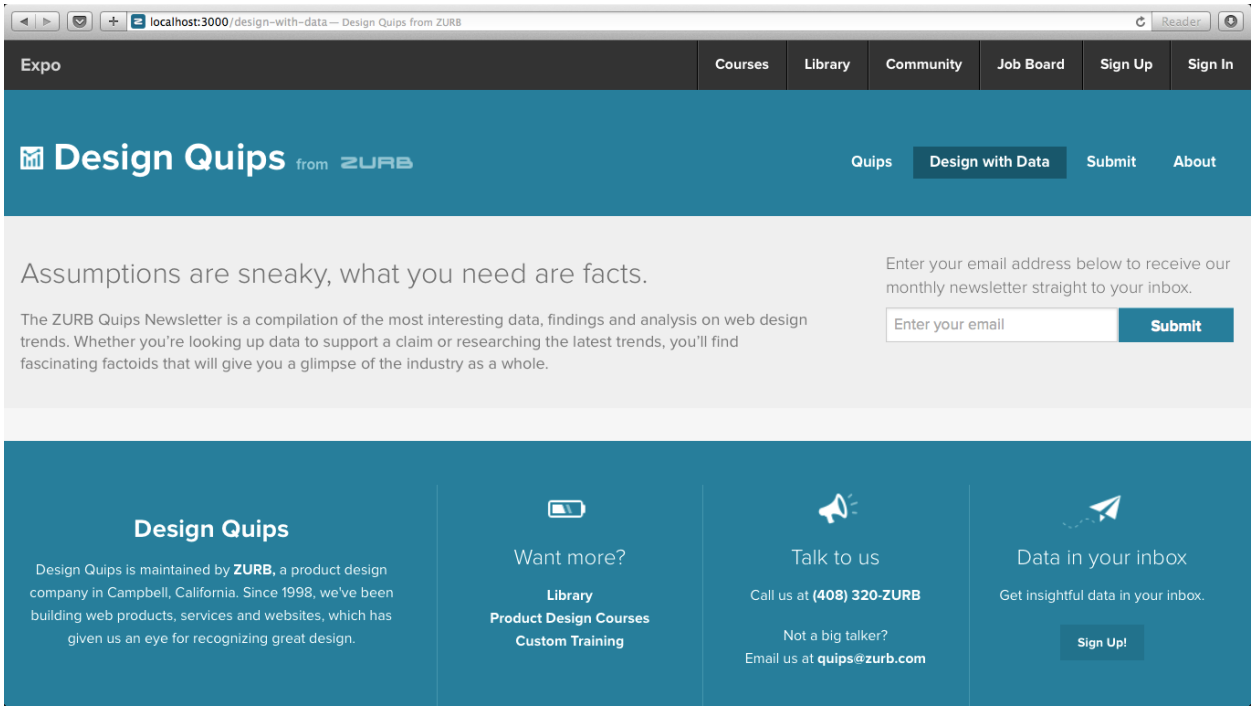

Figura 12. Cadastro na lista de emails

Fonte: Design Quips, Agosto de 2013

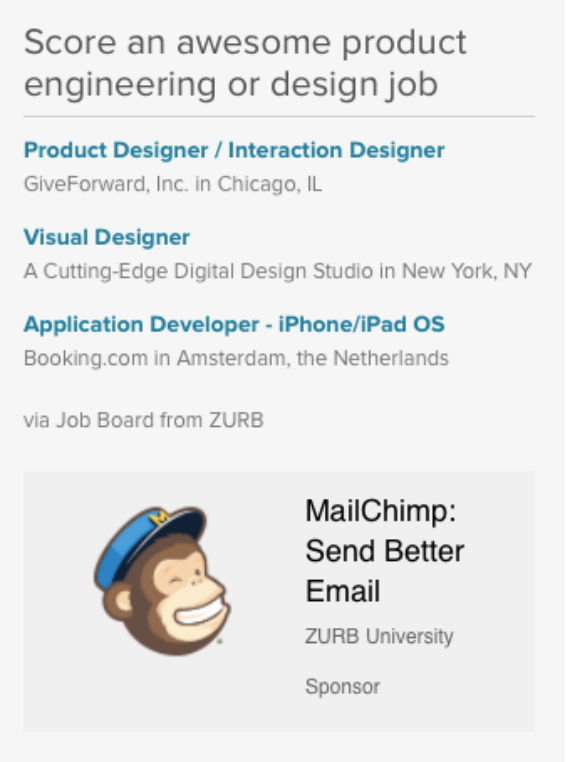

Figura 13. Exemplo de propagandas exibidas no site

Fonte: Design Quips, Agosto de 2013

O Design Quips, como lançado ao fim do projeto, cumpriu os objetivos a que se propôs. A próxima seção descreve como se chegou a esse resultado, através de uma aplicação concreta das etapas e práticas do processo ZURB discutidas anteriormente.

3.2. Desenvolvimento do Projeto

O desenvolvimento do projeto se deu nas quatro etapas discutidas no referencial teórico: a concepção, que antecedeu o início do estágio; o desenvolvimento, que ocupou a maior parte dos três meses de estágio; o lançamento, nas semanas finais da participação no projeto; e o acompanhamento, fase que começou após o fim do estágio da ZURB e continua em andamento.

3.2.1. Concepção

Quando do início do estágio a concepção do Design Quips já havia sido feita. No primeiro dia de trabalho houve uma reunião com o líder de desenvolvimento e o designer responsável pelo Design Quips, onde foi apresentado os objetivos do projeto e o que havia sido produzido na fase de concepção: protótipos HTML estáticos da

listagem de dados (Figura 14) e dos detalhes de um dado (Figura 15), e um documento contendo cerca de 500 dados que haviam sido coletados pelo time nos meses anteriores.

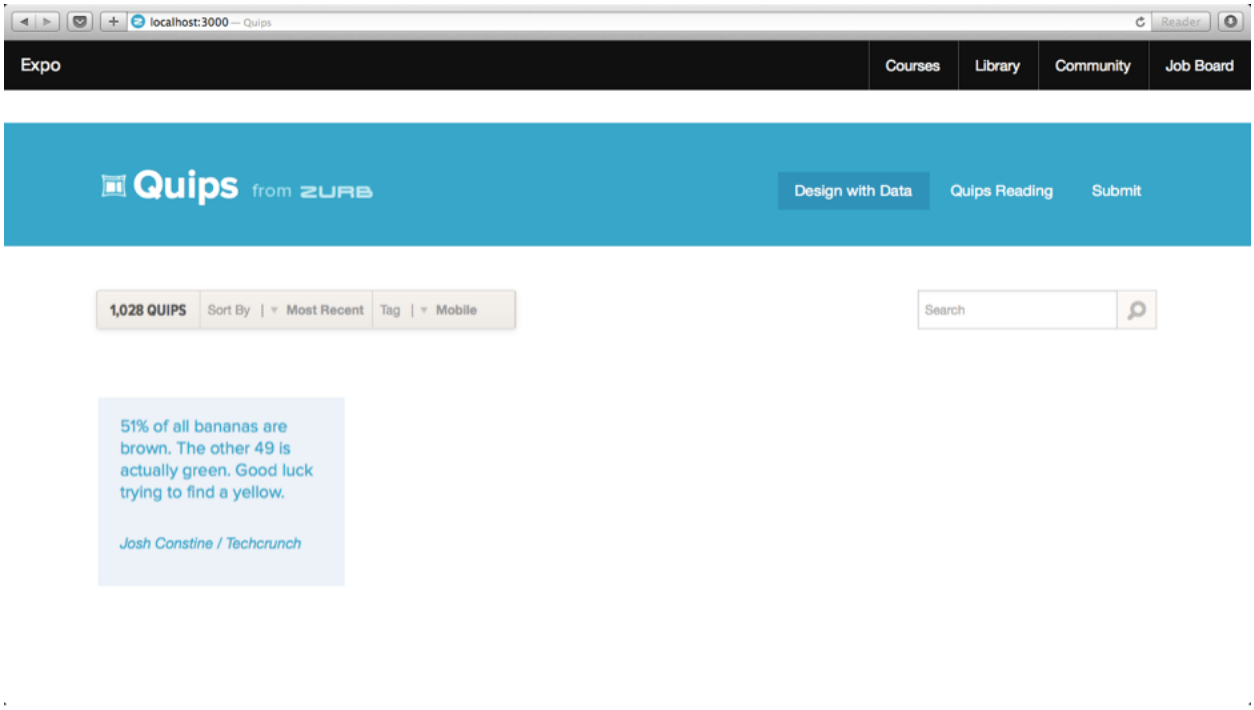

Figura 14. Protótipo de página inicial desenvolvido na concepção

Fonte: Design Quips, Maio de 2013

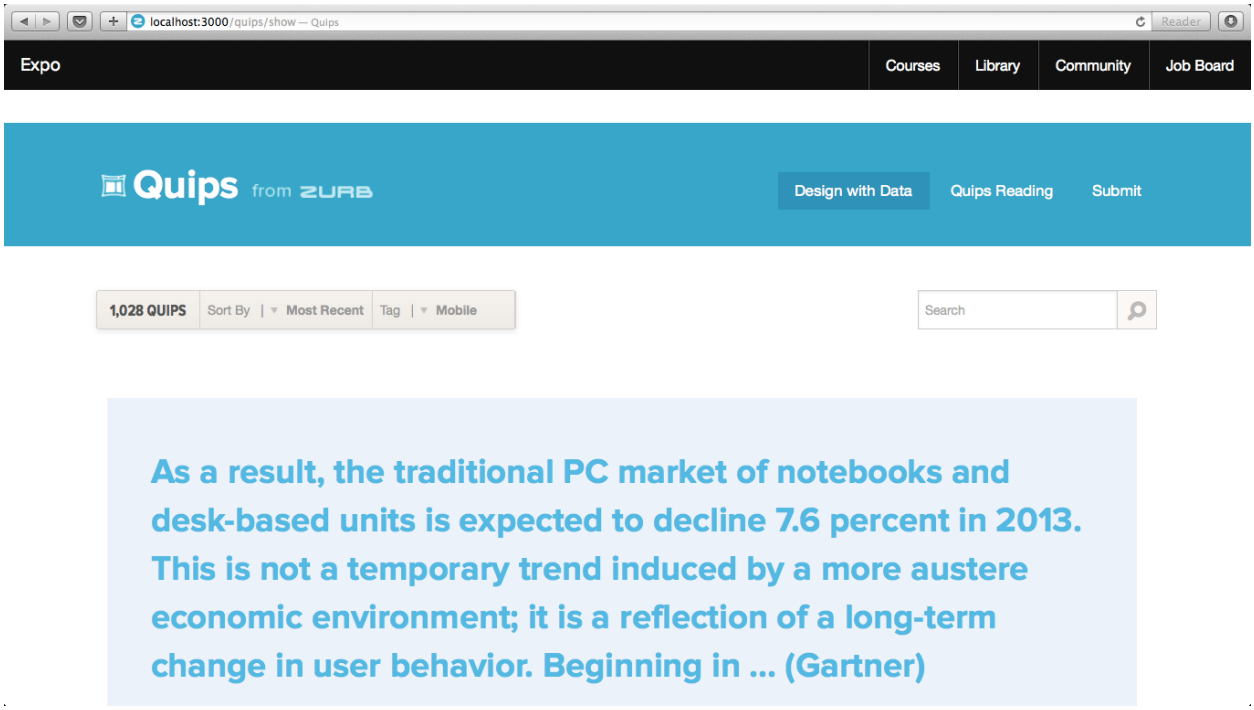

Figura 15. Protótipo de página de detalhes de um dado desenvolvido na concepção

Fonte: Design Quips, Maio de 2013

Foi ainda apresentado o time do projeto, que consistia de um designer, um editor de conteúdo, e eu como desenvolvedor. O líder de desenvolvimento também participaria do projeto como supervisor de estágio, e uma líder de design auxiliaria e revisaria o trabalho do designer.

3.2.2. Desenvolvimento

De posse dos protótipos estáticos e da lista de dados, a meta para a primeira semana de desenvolvimento era implementar um aplicativo básico que listasse e exibisse dados e permitisse importar pelo menos uma parte dos dados existentes. Uma vez que os protótipos estáticos foram implementados numa aplicação Rails básica, foi possível coletar críticas sobre o design de um outro editor de conteúdo da ZURB e do CEO. As críticas ao longo do projeto foram coletadas através da ferramenta Notable (Figura 16).

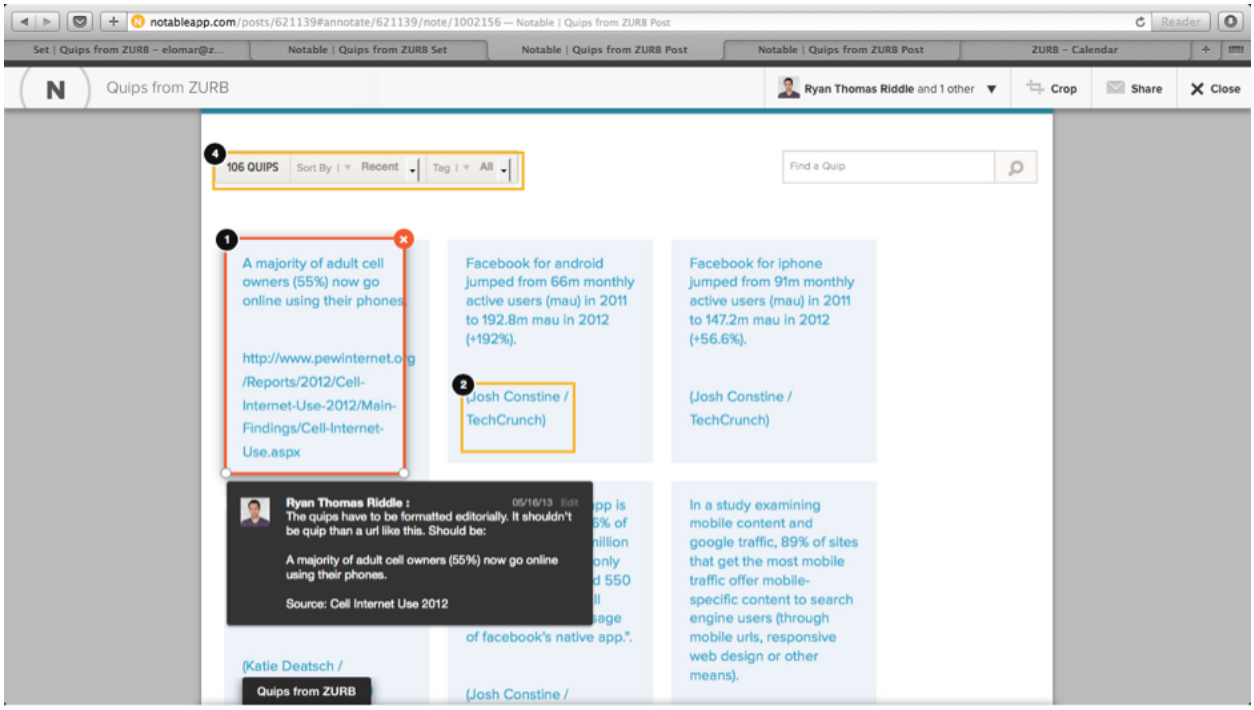

Figura 16. Comentários do time no protótipo inicial

Fonte: Notable, Maio de 2013

Durante o restante da primeira semana o trabalho foi em incorporar as críticas apresentadas e na versão inicial de outras funcionalidades. Algumas das funcionalidades iniciadas nessa semana foram um console de administração dos dados, filtragem por categorias, compartilhamento em redes sociais, e busca textual. A busca textual foi implementada utilizando Sphinx, um “servidor de busca textual de código aberto, desenvolvido desde o começo com performance, relevância (qualidade da busca), e simplicidade de integração em mente. É escrito em C++ e funciona em Linux (RedHat, Ubuntu, etc), Windows, MacOS, Solaris, FreeBSD, e alguns outros sistemas” (SPHINX, 2014).

Em uma reunião do time no início a segunda semana, o editor explicou que seu trabalho de encontrar novos dados consistia em uma vez por semana fazer três buscas específicas no google e extrair dados a partir dos primeiros resultados. As buscas eram “resultados pesquisa web design,” resultados de imagens para “infográfico web design,” e arquivos PDF com os termos “resultados pesquisa web design.” Surgiu então a ideia de automatizar esse processo, e a maior parte da segunda semana foi usada para criar uma ferramenta que fizesse esse processo automaticamente. A ferramenta criada executa automaticamente, uma vez por semana, buscas utilizando a API do Google, e cria um dado com cada resultado com o título e URL da fonte

devidamente preenchidos. Esses dados criados automaticamente eram marcados como “rascunho,” e só apareciam na listagem pública depois de atualizados e aprovados pelo editor. A ferramenta utilizada ainda um algoritmo LSI (Indexador Semântico Latente, do inglês *Latent Semantic Indexer*) para sugerir categorias para os dados importados. Segundo o artigo que introduziu a técnica, a abordagem LSI "tira vantagem da estrutura de ordem superior implícita na associação de termos com documentos (‘estrutura semântica’) para melhorar a detecção de documentos relevantes com base em termos encontrados em consultas” (DEERWESTER et al., 1980). A implementação usada foi a da biblioteca Ruby Classifier (CARLSON; FAYRAM II, 2005). O algoritmo foi treinado inicialmente com os dados coletados até a fase de concepção, e era periodicamente re-treinado com os novos dados publicados pelo editor. Além disso, na segunda semana foi implementada a importação através de arquivo CSV e uma listagem básica das categorias para ser usada pelo editor.

A terceira semana foi dedicada a ajustes visuais e pequenas alterações na exibição de dados e nas ferramentas de busca e filtragem. Fez-se ainda uma reflexão sobre a ferramenta de importação automática de dados a partir de buscas no Google. Apesar de parecer prática na teoria, na realidade tal ferramenta não mostrou-se útil: a maioria dos resultados trazidos era de má qualidade e tinha que ser removido, e a ferramenta não lidava bem com o fato de que cada resultado de busca possuía vários dados e não um só. Essa ferramenta foi removida, e substituída por um cadastro melhorado que possibilitava ao editor cadastrar vários dados sem sair da página e sem repetir informações de fonte ou categoria para dados de um mesmo artigo.

Na terceira semana houve uma reunião com o líder de desenvolvimento e um dos desenvolvedores da ZURB que apresentaram o Platform, sistema integrado de autenticação e autorização. Na época cada propriedade da ZURB possuía seu próprio cadastro de usuários. O Design Quips foi o primeiro projeto a utilizar o cadastro unificado, e seu sucesso nessa integração abriu espaço para que o cadastro unificado fosse estendido a todas as outras ferramentas ZURB. O Platform é um provedor que suporta o protocolo OAuth 2, “um framework de autorização que permite a uma aplicação de terceiros obter acesso limitado a um serviço HTTP, em nome do proprietário de um recurso orquestrando uma interação de aprovação entre o proprietário do recurso e o serviço HTTP” (HARDT, 2012). O Design Quips delega autenticação ao Platform, que permite a um usuário se autenticar via email e senha, ou

via as redes sociais Facebook, Twitter, e Google+. Platform informa ainda aos clientes se o papel do usuário autenticado é de usuário comum ou administrador, informação que foi suficiente para implementar autorização no Design Quips.

A quarta semana foi dedicada a integração com o Platform e a preparação para a implantação do Design Quips na estrutura de servidores da ZURB. Essa preparação foi em sua maior parte feita por um desenvolvedor do time que tem acesso aos servidores e que conhece a infra-estrutura existente.

O tema da sexta semana foi ajustar o site para funcionar corretamente em dispositivos móveis. O site foi desenvolvido seguindo princípios de design responsivo, e o layout se ajustava de acordo com a resolução do cliente. As maiores mudanças implementadas nessa semana para acomodar dispositivos móveis foram uma barra de filtragem que usava elementos *select* ao invés de ícones (Figura 17), já que os ícones ficavam muito pequenos e difíceis de clicar em resoluções baixas; e um menu que usava a técnica "*off-canvas.*"

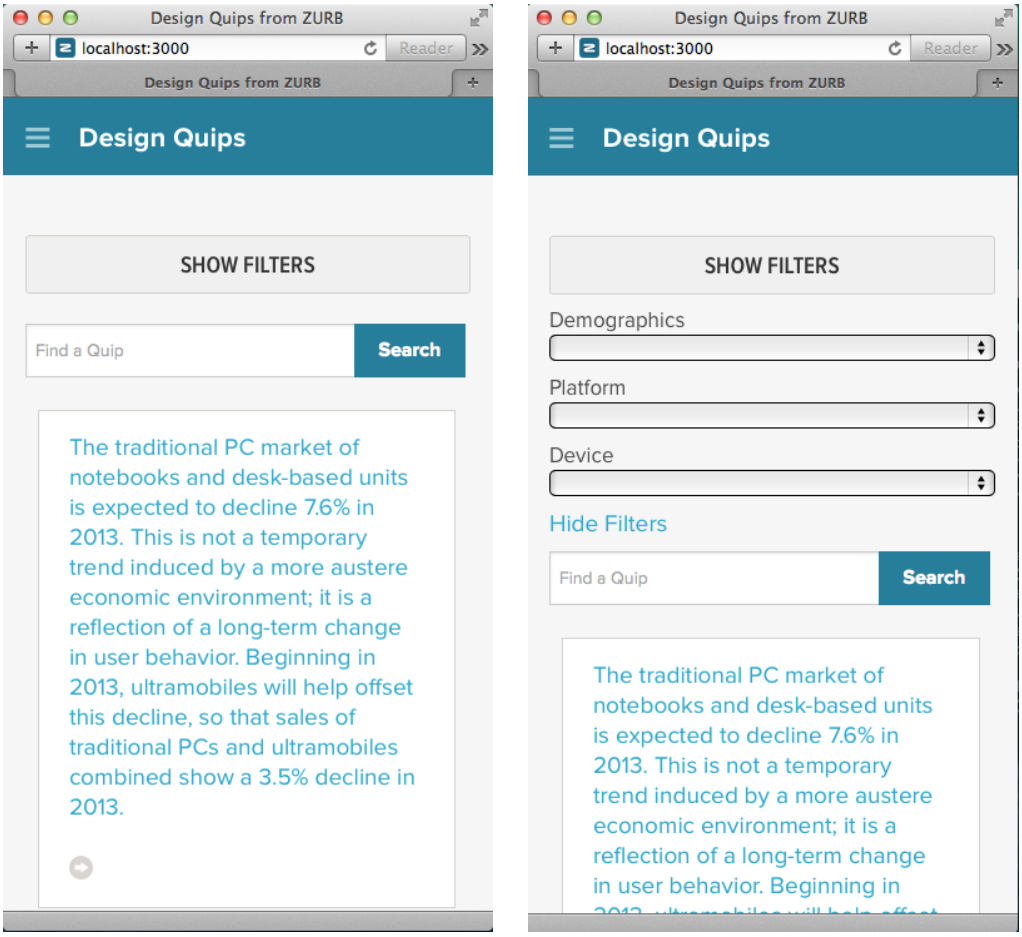

Figura 17. Barra de filtragem adaptada para dispositivos móveis, estado inicial e estado visível

Fonte: Design Quips, Junho de 2013

Nas duas semanas seguintes foram feitos ajustes nas ferramentas administrativas a serem usadas pelo administrador, corrigida a integração com o Platform, adicionada a exibição de propagandas, e criadas as páginas de conteúdo do site. Foram duas páginas de conteúdo: uma sobre o projeto Design Quips, que explica a motivação por trás do projeto e como ele podia ser útil a usuários; e uma página que permitia se inscrever na lista de email do Design Quips que mensalmente envia novos dados e artigos relacionados ao conteúdo do site.

Com essas funcionalidades implementadas, deu-se por terminada a etapa de desenvolvimento e iniciou-se a o processo lançamento.

3.2.3. Lançamento

A primeira tarefa da etapa de lançamento foi revisar cuidadosamente todas as funcionalidades implementadas para corrigir quaisquer falhas existentes, garantir que o design estava consistente, e checar que o sistema funcionava como devido no ambiente de produção. Uma vez feitas todas as correções que o time encontrou, o projeto foi apresentado primeiro a líder de design e depois ao CEO, que fizeram novas críticas e sugeriram novas mudanças. Nessa etapa as mudanças sugeridas eram correções e ajustes a funcionalidades existentes, e não novas funcionalidades.

Depois de incorporadas as críticas, foi feito o processo de garantia de qualidade. Tal processo consistiu em testar o site em diversos plataformas desktops e móveis. Design Quips foi testado nos navegadores Firefox, Safari, Chrome e Internet Explorer rodando em desktops, e nos dispositivos móveis iPhone 3GS e 4S, Nexus 4 e 7, Galaxy S3, Microsoft Surface, um smartphone rodando Windows Phone e um Blackberry, e as falhas encontradas foram corrigidas.

Com o site pronto e testado em diversos dispositivos, foi feita uma nova reunião com o CEO. Nessa reunião foi definida a data de lançamento e criada uma lista de tarefas do lançamento. A lista era composta por tarefas necessárias ao lançamento do Design Quips, e não mudanças no projeto em si. Entre essas tarefas estavam preparar uma apresentação que seria internamente para que todos da empresa conhecessem melhor o projeto, escrever um artigo para ser publicado no blog da empresa quando do lançamento oficial, entrar em contato com outras publicações que podiam se interessar em cobrir o lançamento, e adicionar o Design Quips às ferramentas de monitoramento usadas na ZURB.

Os itens dessa lista foram feitos ao longo da semana, pelo time e por outros membros da ZURB, e o site foi lançado dia 1 de agosto com um anúncio no blog oficial da empresa e em todas os perfis da ZURB em redes sociais.

3.2.4. Acompanhamento

Depois do lançamento começou a etapa de acompanhamento, que consiste em acompanhar o uso do site, periodicamente reavaliar seus objetivos, e propor e implementar mudanças. A minha participação no projeto findou com o lançamento, mas é possível perceber pela evolução do Design Quips após o lançamento que o projeto continua sendo acompanhado e desenvolvido. As figuras 18 e 19 mostram a evolução do projeto 3 e 6 meses após o lançamento, respectivamente.

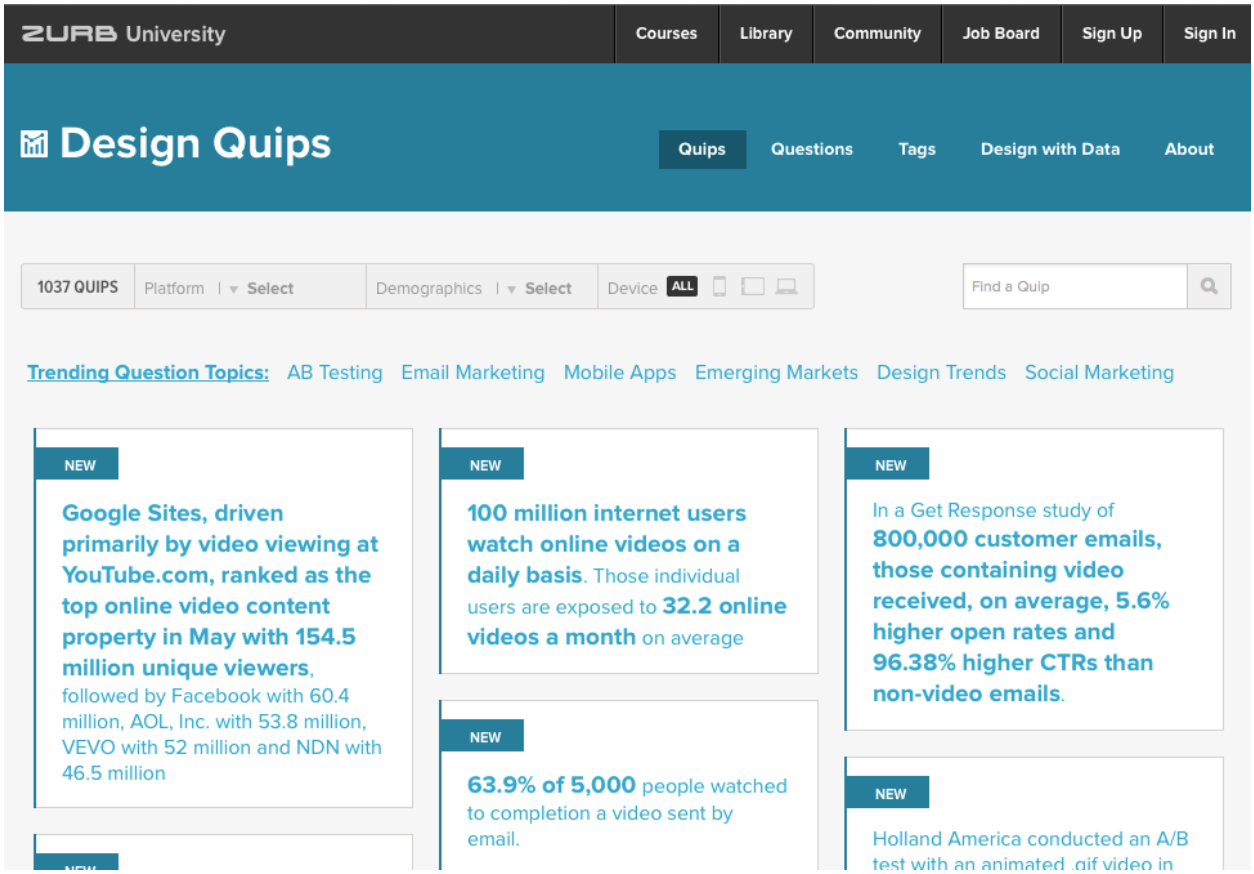

Figura 18. Segunda versão da página inicial

Fonte: Design Quips, Novembro de 2013

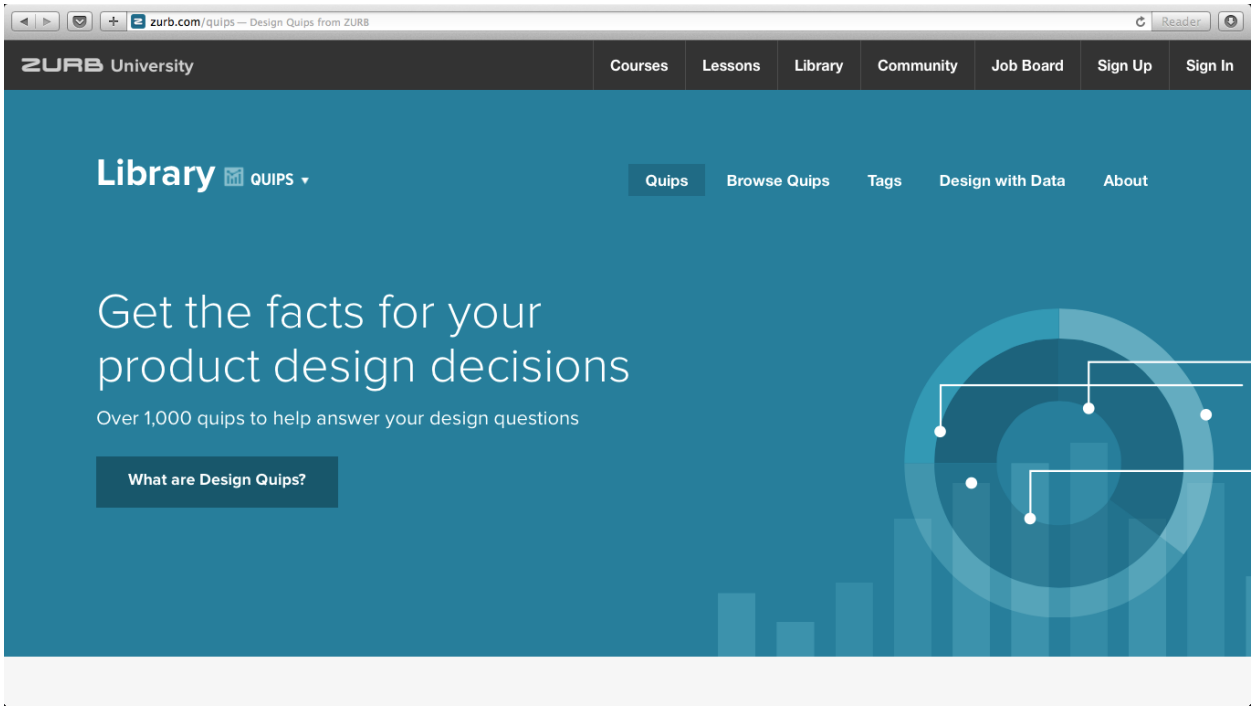

Figura 19. Terceira versão da página inicial

Fonte: Design Quips, Fevereiro de 2014

4. CONCLUSÃO

Seguindo o processo ágil da empresa ZURB foi possível desenvolver e lançar o software Design Quips, atingindo o objetivo geral a que o projeto se propôs. Os objetivos específicos também foram atingidos, com graus variados de sucesso.

O objetivo de mapear requisitos funcionais a partir dos objetivos gerais que haviam sido pré-estabelecidos foi atingido, e foi criado um conjunto de requisitos que atende as necessidades dos usuários do sistema. Houveram alguns problemas no cumprimento desse objetivo, no entanto, na forma de desperdício e retrabalho causados pelo mapeamento e implementação de requisitos que acabaram sendo removidos antes que o projeto fosse lançado.

Os objetivos de criar protótipos estáticos com o framework Foundation e protótipos dinâmicos na plataforma Ruby on Rails também foram atingidos, e sem maiores complicações. O desenvolvimento andou no prazo estipulado, e os protótipos cumpriram seus papéis de orientar o time na definição detalhada dos requisitos.

Por fim, o projeto também cumpriu com o objetivo proposto de desenvolver o código-fonte Ruby on Rails de implementação do software Design Quips e lançá-lo publicamente dentro do período estipulado de três meses, sem que fosse necessário dedicar mais recursos ao projeto ou diminuir seu escopo.

Pode-se dizer, a partir do cumprimento dos objetivos propostos, que o projeto foi realizado com sucesso: Design Quips foi idealizado, desenvolvido e lançado dentro do escopo, custo e prazo previstos inicialmente.

4.1. Trabalhos Futuros

Pode-se dar continuidade a esse trabalho detalhando atividades realizadas depois do lançamento do projeto, na etapa de acompanhamento. Trabalhos futuros podem abordar o que se aprendeu com o uso do sistema pelo público, as motivações por trás das atualizações no projeto, e as mudanças no ritmo e na forma de trabalho que marcam a etapa de acompanhamento.

REFERÊNCIAS BIBLIOGRÁFICAS

BECK, Kent et al. **Manifesto for Agile Software Development.** 2001. Disponível em: <http://agilemanifesto.org>. Acesso em: 13 fev. 2014.

BECK, Kent et al. **Principles behind the Agile Manifesto.** 2001. Disponível em: <http://agilemanifesto.org/principles.html>. Acesso em: 13 fev. 2014.

CARLSON, Lucas; FAYRAM II, David . **Ruby Classifier:** Bayesian and LSI classification library. 2005. Disponível em: <http://classifier.rubyforge.org>. Acesso em: 21 jan. 2014.

DEERWESTER, Scott et al. Indexing by Latent Semantic Analysis. **Journal Of The American Society For Information Science.** New York, p. 391-407. set. 1990.

FOWLER, Martin. **Patterns of Enterprise Application Architecture.** Boston: Addison-wesley Professional, 2012.

GITHUB (San Francisco). **Search.** 2014. Disponível em: <https://github.com/search?p=2&q=stars:>1&s=stars&type=Repositories>. Acesso em: 16 fev. 2014.

GITHUB (San Francisco). **Search:** Language: Ruby. 2014. Disponível em: <https://github.com/search?q=forks:>-1&type=Repositories&ref=advsearch&l=Ruby>. Acesso em: 21 fev. 2014.

HARDT, Dick (Ed.). **RFC 6749:** The OAuth 2.0 Authorization Framework. 2012. Disponível em: <http://tools.ietf.org/html/rfc6749>. Acesso em: 11 fev. 2014.

HEDGE, Alan. The Open-Plan Office: A Systematic Investigation of Employee Reactions to Their Work Environment. **Environment And Behavior.** Salt Lake City, p. 519-542. set. 1982.

KRASNER, Glenn E.; POPE, Stephen T.. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 system. **Journal Of Object Oriented Programming.** Mountain View, p. 26-49. abr. 1988.

LINGUAGEM de Programação Ruby. 2014. Disponível em: <https://www.ruby-lang.org/pt>. Acesso em: 05 fev. 2014.

MARCOTTE, Ethan. **Responsive Web Design.** 2010. Disponível em: <http://alistapart.com/article/responsive-web-design/>. Acesso em: 11 jan. 2014.

MEEKER, Mary; WU, Liang. **2013 Internet Trends.** 2013. Disponível em: <http://www.kpcb.com/file/kpcb-internet-trends-2013>. Acesso em: 7 jan. 2014.

MONTEIRO, Júlio. **Ruby on Rails Brasil.** 2014. Disponível em: <http://www.rubyonrails.com.br>. Acesso em: 13 fev. 2014.

NIWATORI - Álbuns da web do Picasa: Ruby. 2007. Disponível em: <https://picasaweb.google.com/Dikiwinky/Ruby#5116531304417868130>. Acesso em: 13 mar. 2014.

PILGRIM, Mark. **HTML5:** Up and Running. Sebastopol: O'reilly Media, 2010. 222 p.

PLEKHANOVA, Julia. Evaluating web development frameworks: Django, Ruby on Rails and CakePHP. **The Ibit Report.** Philadelphia, p. 4-17. mar. 2011.

PRESSMAN, Roger S.. **Engenharia de Software.** 6. ed. Porto Alegre: Makron Books, 2006.

SPHINX TECHNOLOGIES INC (EUA). **About:** Sphinx. 2014. Disponível em: <http://sphinxsearch.com/about/sphinx/>. Acesso em: 21 jan. 2014.

TELES, Vinícius Manhães. **Extreme Programming:** Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. Rio de Janeiro: Novatec, 2004. 320 p.

VEGT, Gerben S. van Der; BUNDERSON, J. Stuart. Learning and performance in multidisciplinary teams: The importance of collective team identification. **Academy Of Management Journal.** Briarcliff Manor, p. 532-547. jun. 2005.

ZURB (Campbell). **Data is Fun for Nerds: Introducing Design Quips.** 2013. Disponível em: <http://zurb.com/article/1233/data-is-fun-for-nerds-introducing-design->. Acesso em: 1 ago. 2013.

ZURB (Campbell). **Foundation:** The most advanced responsive front-end framework in the world.. 2014. Disponível em: <http://foundation.zurb.com>. Acesso em: 23 fev. 2014.

ZURB (Campbell). **Friday 15 from ZURB.** 2013. Disponível em: <http://zurb.com/friday15>. Acesso em: 09 fev. 2014.

ZURB (Campbell). **ZURB Studios:** Designing Sparks Together. 2014. Disponível em: <http://zurb.com/studios>. Acesso em: 7 fev. 2014.