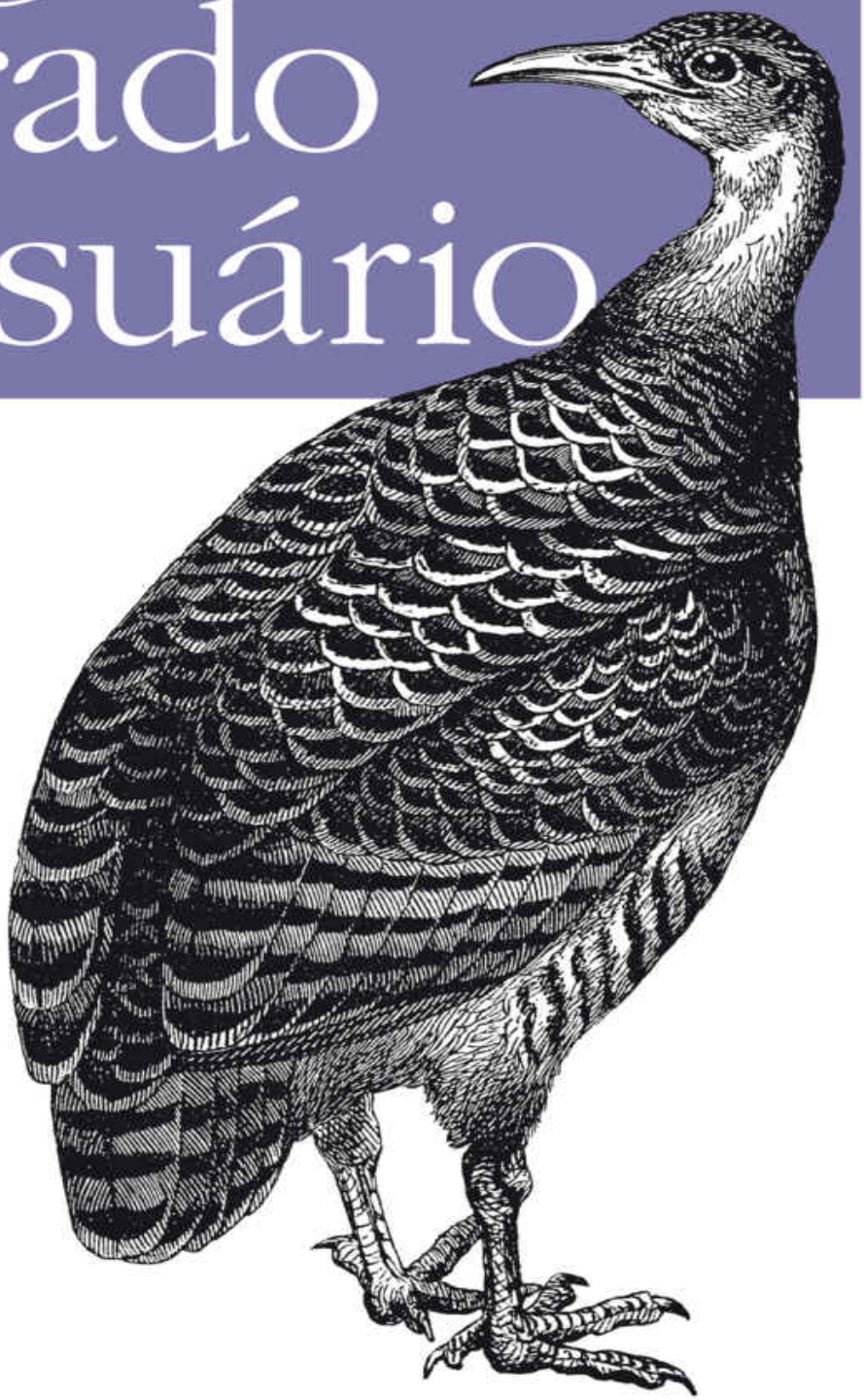
Um guia para o desenvolvimento de aplicativos amigáveis

Design centrado no usuário



O'REILLY®
novatec

Travis Lowdermilk

Um guia para o desenvolvimento de aplicativos amigáveis

Design Centrado no Usuário

Travis Lowdermilk

O'REILLY®
Novatec
São Paulo | 2019

Tradução em português autorizada da edição em inglês da obra User Centered Design, ISBN 9781449359805 © 2013 Travis Lowdermilk. Esta tradução é publicada e vendida com a permissão da O'Reilly Media, Inc., detentora de todos os direitos para publicação e venda desta obra.

Editor: Rubens Prates
Tradução: Lúcia Ayako Kinoshita
Revisão técnica: Aurelio Jargas
Revisão gramatical: Marta Almeida de Sá
Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-772-5

Histórico de edições impressas:

Setembro/2018 Terceira reimpressão
Setembro/2017 Segunda reimpressão
Maio/2015 Primeira reimpressão
Junho/2013 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Para minha esposa — obrigado por me incentivar a sonhar.

Para meus dois meninos — obrigado por serem a razão de eu continuar sonhando.

Para meu irmão — obrigado por me dizer quando é hora de acordar.

Sumário

Prefácio

Capítulo 1 ■ Nosso mundo mudou

Capítulo 2 ■ O que é Design Centrado no Usuário?

- DCU não é usabilidade
- DCU não é subjetivo
- DCU não é apenas design
- DCU não significa perda de tempo ou de dinheiro
- DCU não é relatório de bug
- DCU não é uma distração

Capítulo 3 ■ Trabalhando com usuários

- E se eu não tiver acesso aos usuários?
  - Saber quando ouvir e quando não ouvir os usuários
- Lidando com diferentes tipos de usuários
  - O informante exagerado
  - O obcecado por controle
  - O Advogado do Diabo
- Lidando com a negatividade

Capítulo 4 ■ Planejar

- Como sei qual é o plano adequado para mim?
- Criando uma definição de missão para a equipe
- Definindo seu projeto
- Coletando requisitos de usuário
- Criando requisitos funcionais
- Documentando modelos de dados e de fluxo de trabalho
- Documentando protótipos

Revisando sua documentação

Capítulo 5 ■ Criando um manifesto pessoal

Exercitando restrições
Construindo uma narrativa
Criando personas
Criando cenários

Capítulo 6 ■ Criatividade e experiência de usuário

Ter metas de experiência de usuário
Criatividade exige coragem e trabalho árduo
Pegue um lápis
Liberdade de criação
Entendendo sua meta
Roube (quero dizer, empreste) de outros
Criatividade exige questionamento

Capítulo 7 ■ Princípios de design

Princípio da proximidade (Princípio da Gestalt)
Visibilidade, feedback visual e proeminência visual
Hierarquia
Modelos mentais e metáforas
Revelação progressiva
Consistência
Disponibilidade e restrições
Confirmação
A Lei de Hick
Lei de Fitt

Capítulo 8 ■ Reunindo feedback

Quantos usuários serão necessários?
Pesquisas
Conduzindo entrevistas
Análise de tarefas

Avaliação heurística
Criação de storyboards
Usando protótipos
Testes A/B

Capítulo 9 ■ Estudos de usabilidade
O que são estudos de usabilidade?
Criando um plano de testes
Introdução
Garantias
Diretrizes para testes
Tarefas
Conclusão
Agradecimentos
De que você precisará
Cronômetro
Bloco de anotações
Ambiente
Planilhas ou banco de dados
Câmeras e gravadores de áudio
Conduzindo o estudo
Não hesite em praticar
Organizando seus resultados

Capítulo 10 ■ Você nunca termina
É impossível fazer certo da primeira vez
Esteja preparado para reiniciar
Últimas reflexões

Capítulo 11 ■ Outros recursos
Twitter
Ferramentas para prototipagem
Sites

Apêndice A ■ Exemplo de template para projetos

Template

Template de Projeto Ciclo de Vida do Desenvolvimento de Software

Dan Welks

Apêndice B ■ Referências

Prefácio

Este livro é adequado para mim?

Espero realmente que seja. Vamos ver se consigo ajudar a definir as expectativas.

Nos vários anos em que venho desenvolvendo aplicativos, tenho investido bastante tempo na tentativa de entender os usuários. Sou obcecado por descobrir o que os compele: O que os motiva? O que os frustra? O que os faz escolher um aplicativo no lugar de outro? O que posso fazer para que eles optem pelos *meus* aplicativos?

Com base em várias discussões que já tive com desenvolvedores de todo o mundo, posso seguramente assumir que não estou sozinho. Desse modo, decidi escrever este livro para ajudar os desenvolvedores a entender melhor seus usuários. Este livro não é destinado aos profissionais da área de experiência de usuário (UX, ou User Experience) ou aos designers profissionais. Pelo contrário, meu objetivo é ajudar desenvolvedores a compreender as práticas fundamentais do design centrado no usuário, da usabilidade e da experiência de usuário.

Esta discussão deverá ser sua plataforma de lançamento para o mundo dos *experts* em usabilidade. Você conhecerá as motivações, as terminologias e as estratégias para julgar o sucesso (ou o fracasso) de um aplicativo. Minha esperança é que, com esse conhecimento, você tenha muito mais autoconfiança para começar a estudar os usuários de maneira mais significativa.

A indústria da interação humano-computador é vasta e inclui décadas de pesquisas científicas. Não seria possível abranger todos os aspectos daquilo que, atualmente, é conhecido como usabilidade. No entanto este livro está repleto de ótimos exemplos

(que também são práticos) para ajudá-lo a dar o primeiro passo.

Com este livro, você aprenderá a:

- implementar práticas de design centrado no usuário e de usabilidade;
- lidar com diferentes tipos de usuários e suas personalidades únicas;
- criar uma visão que será essencial ao sucesso de seu aplicativo;
- criar um plano que o ajudará a navegar pelo processo de desenvolvimento e evitará erros caros;
- estimular a criatividade e criar aplicativos atraentes usando princípios comuns de design;
- reunir feedback e tomar decisões de design que estejam fundamentadas.

Durante a nossa discussão, compartilharei ferramentas e processos que considero úteis em meu próprio trabalho. Embora as diversas tecnologias, as histórias e os exemplos usados neste livro possam se tornar ultrapassados ou obsoletos, as lições que aprenderemos com eles permanecerão atuais.

Talvez você seja um desenvolvedor solitário, criando aplicativos móveis para uma base enorme de consumidores. Talvez esteja trabalhando em uma equipe de desenvolvimento pequena, criando aplicativos de linha de negócios para sua organização. Ou, quem sabe, você tenha começado a programar como um *hobby*, na esperança de fazer disso uma profissão de período integral. Muitos de nós não temos acesso a um profissional de UX ou a um designer em nossa equipe. Precisamos nos virar por conta própria. Embora o valor da UX e de suas tecnologias associadas estejam se tornando cada vez mais evidentes, muitas organizações ainda não estão preparadas para investir em cargos de período integral.

Tampouco é o caso de se tratar somente de desenvolvedores em empresas. Em nosso programa semanal na Internet, conversei com

muitos desenvolvedores que estão criando aplicativos sem nenhum treinamento formal em design ou em usabilidade. Com muita frequência, eles aprendem o suficiente para sobreviver, mas continuam a lutar com os fundamentos sobre como criar uma experiência de usuário que seja ótima.

Em qualquer uma dessas situações, as informações contidas neste livro ajudarão você a desenvolver aplicativos melhores, fortalecendo suas habilidades de observação e de design.

O livro está dividido nos seguintes conceitos:

O que é design centrado no usuário?

Para começar, faremos uma discussão sobre os relacionamentos e os erros conceituais comuns associados à usabilidade, ao design centrado no usuário e à experiência de usuário.

Trabalhando com usuários

Discutiremos estratégias para obter o máximo de seus usuários.

Ter um plano

Desenvolver um aplicativo de sucesso (*qualquer artefato* de sucesso, para dizer a verdade) exige um planejamento cuidadoso. Abordaremos os passos críticos que devem ser incluídos em seu processo de desenvolvimento. Esses itens o ajudarão a implementar a obtenção de feedback do usuário de maneira eficiente.

Criando um manifesto pessoal

Um aspecto torna-se evidente quando conversamos com desenvolvedores e designers de sucesso. Cada um deles tem uma visão bem clara sobre o que querem realizar com seus aplicativos. Procuraremos descobrir por que ter uma visão é a chave para a criação de um produto significativo.

Criatividade e experiência de usuário

É preciso ter *insights* criativos para gerar continuamente ideias que

atendam às necessidades dos usuários. Discutiremos sobre maneiras de estimular sua própria criatividade e ter inspiração.

Princípios de design

Felizmente, para nós, há muitos princípios para nos guiar em direção a designs que comprovadamente funcionam. Discutiremos alguns dos princípios de design mais populares que podem ser utilizados em seus aplicativos para melhorar incrivelmente a usabilidade.

Reunindo feedback

Coletar dados significativos dos usuários é crucial para o design centrado no usuário. Discutiremos os diferentes métodos empregados pelos pesquisadores para resolver problemas com a experiência de usuário.

Estudos de usabilidade

Observar os usuários quando estão usando seus aplicativos consiste em um dos processos mais importantes da pesquisa relacionada com a usabilidade. Discutiremos sobre as várias ferramentas necessárias para conduzir seus próprios estudos.

Convenções usadas neste livro

As seguintes convenções tipográficas são usadas neste livro:

Itálico

Indica termos novos, URLs, endereços de email, nomes e extensões de arquivos.

Monoespaçado

Usada para listagens de programas, assim como dentro de parágrafos, para se referir a elementos de programas, como nomes de variáveis ou de funções, bancos de dados, tipos de dados, variáveis de ambiente, comandos e palavras-chave.

Monoespaçado em negrito

Mostra comandos ou outros textos que devam ser digitados literalmente pelo usuário.

Monoespaçado em itálico

Mostra o texto que deve ser substituído por valores fornecidos pelo usuário ou determinados pelo contexto.

Este ícone significa uma dica, uma sugestão ou uma observação em geral.

Uso de exemplos de código de acordo com a política da O'Reilly

Este livro está aqui para ajudá-lo a fazer seu trabalho. De modo geral, se este livro incluir exemplos de código, você pode usar o código em seus programas e em sua documentação. Você não precisa nos contatar para pedir permissão, a menos que esteja reproduzindo uma parte significativa do código. Por exemplo, escrever um programa que use diversas partes de código deste livro não requer permissão. Porém vender ou distribuir um CD-ROM de exemplos de livros da O'Reilly requer permissão. Responder a uma pergunta mencionando este livro e citar o código de exemplo não requer permissão. Em contrapartida, incluir uma quantidade significativa de código de exemplos deste livro na documentação de seu produto requer permissão.

Agradecemos, mas não exigimos, atribuição. Uma atribuição geralmente inclui o título, o autor, a editora e o ISBN. Por exemplo: "*User-Centered Design* por Travis Lowdermilk (O'Reilly). Copyright 2013 Travis Lowdermilk, 978-1-449-35980-5."

Se você achar que o seu uso dos exemplos de código está além do razoável ou da permissão concedida, sinta-se à vontade em nos contatar em *permissions@oreilly.com*.

Como entrar em contato conosco

Envie seus comentários e suas dúvidas sobre este livro à editora escrevendo para: *novatec@novatec.com.br*.

Temos uma página web para este livro na qual incluímos erratas, exemplos e quaisquer outras informações adicionais.

- Página da edição em português
  http://www.novatec.com.br/catalogo/7522366_designcentrado
- Página da edição original em inglês
  http://shop.oreilly.com/product/0636920028741.do

Para obter mais informações sobre os livros da Novatec, acesse nosso site em: *http://www.novatec.com.br*.

Agradecimentos

Pessoas que me ajudaram a escrever este livro

Estas pessoas foram gentis o bastante para passar algum tempo comigo, o que permitiu que eu compartilhasse seus conhecimentos com vocês. Como escrevi neste livro, para ser ótimo, devemos seguir pessoas ótimas. Em minha opinião, essas são algumas das melhores:

Julian Walker

Engenheiro líder na FiftyThree e criador do Paper. Se quiser saber mais sobre o que Julian faz, siga-o no Twitter em @julianwa.

Jeff Weir

Designer de UX na Microsoft, já trabalhou nas equipes de Windows e Live Labs. Você pode encontrar apresentações feitas por Jeff no Channel 9 (*http://msdn.channel9.com*), site de vídeos para desenvolvedores, da Microsoft.

Billy Hollis

Um desenvolvedor e evangelizador que promove a importância das boas práticas de usabilidade. Billy é bastante conhecido na

comunidade de desenvolvedores para .NET da Microsoft e tem sua própria empresa de consultoria, a Next Version Systems.

Robby Ingebretsen

Um designer de UX e fundador da Pixel Lab, uma empresa de design e estratégia para software, localizada em Seattle. Você pode saber tudo sobre Robby no Twitter em @ingebretsen ou em seu blog pessoal (*http:// thinkpixellab.com*).

Mark e Lisa

Este livro não estaria em suas mãos se não fossem a orientação e a autêntica genialidade desses dois. Vai, Blue Demons!

Mary Treseler e a família na O'Reilly Media

Este livro mostra que realmente o mercado de desenvolvimento de aplicativos está mudando. Saudações a toda a equipe da O'Reilly Media por ajudar, a mim e a outras pessoas, a compreender a importância de uma excelente usabilidade e um ótimo design. A O'Reilly continua provando que é um verdadeiro guia de suporte aos desenvolvedores, ajudando-os a permanecer à frente nesse cenário em constantes mudanças.

Mary, obrigado por ser tão simpática e por me fazer sentir como um verdadeiro autor, algo que soa muito estranho quando digo em voz alta.

Amanda, seu olhar crítico e sua sabedoria tornaram este livro muito melhor do que algum dia eu poderia imaginar. Muito obrigado.

Agradeço aos vários revisores que ofereceram suas opiniões criteriosas e suas ideias.

Pessoas que me ajudaram na vida

Meus pais

Kim, Deborah, Kathi, Joe, David e John. Obrigado por todo o amor e pelo apoio. Este livro é uma realização tanto minha quanto de

vocês.

Meus irmãos

Ao melhor grupo de irmãos que existe: Ryan, Brandon e Brett — obrigado por sempre cuidarem de mim.

Minha irmã

Hope, você tem uma seção só sua porque é minha irmã favorita. Abraços.

Meus bons amigos

JC, Daniel, Matt e Travis (e suas esposas e seus filhos também)! V-Town, caras! Uma das minhas melhores amigas, Margery Godfrey. Mantive minha promessa.

Meus colegas de trabalho

Toda a equipe do Kaweah Delta Health Care District: Dave, Nick, Steven, Eli, Mark, Anita e Tim – obrigado por suportarem minhas incessantes divagações. A maior parte do que está neste livro é resultado da paciência e do feedback ativo de vocês.

Meus meninos

Noah e Jackson, permitam que este livro seja um testemunho do poder do trabalho árduo e da determinação. Vocês sempre estão em meu coração e em meus pensamentos.

Minha esposa

Jackie, como tudo em minha vida, este livro começa e termina com você. Obrigado por todo o seu trabalho árduo e pelo apoio. Tudo isso não teria sido possível sem você.

CAPÍTULO 1
Nosso mundo mudou

“Ficar parado é a maneira mais rápida de retroceder em um mundo em rápida mudança.”

– Lauren Bacall

Em 9 de janeiro de 2007, um homem caminhou silenciosamente por um palco e mudou o curso da história da tecnologia. Ele anunciou que sua empresa estava para lançar um produto que mudaria para sempre o modo como nos comunicamos.

Então, de maneira dramática, ele mostrou um telefone no qual ele e sua empresa vinham trabalhando havia mais de cinco anos. Jornalistas capturaram freneticamente as imagens do dispositivo, enviando-as rapidamente a todos os cantos do mundo. O homem demonstrou como era possível dar um zoom em imagens simplesmente com um gesto de apertar os dedos e navegar pela biblioteca de músicas arrastando um único dedo pela tela. Ele percorreu vários aplicativos: um bloco de notas, um calendário, uma bússola e mapas detalhados. Ninguém tinha visto nada parecido antes. O telefone parecia ser um produto de ficção científica. Porém era bastante real e tão pequeno que podia caber em seu bolso.

Naquela época, eu trabalhava como programador web em um hospital infantil. Lembro-me de estar sentado em minha escrivaninha assistindo à apresentação em um blog ao vivo, esperando o que parecia ser uma eternidade para ver as imagens serem transmitidas em meu computador. Assim que vi a primeira imagem do iPhone, tive a sensação de estar testemunhando algo

importante. Naquele momento, ainda não tinha me dado conta da extensão do impacto do iPhone em nosso mercado; porém, como desenvolvedor, pude ver que o nível havia sido elevado. Sabia que os dias em que era possível se safar com uma interface de usuário (UI, ou User Interface) desordenada e layouts confusos estavam contados.

Meus usuários iriam esperar muito mais.

Não seria suficiente que meus aplicativos tivessem tempos de carga rápidos ou um rol de facilidades. Meus usuários iriam querer o iPhone. Não somente o produto, especificamente, mas o que ele representava. Era intuitivo, minimalista e cativante; e agora meus usuários tinham um excelente exemplo de como tudo deveria funcionar. Novas formas de interação passaram a fazer parte de conversações, e termos como Multi-Touch e NUI (Natural User Interface, ou Interface Natural de Usuário) instantaneamente tornaram-se parte da língua franca dos desenvolvedores.

Um ano depois, a Apple inaugurou a App Store para o iPhone, gerando uma explosão no desenvolvimento de aplicativos. Os desenvolvedores começaram a competir em um mercado saturado, no qual os usuários tinham milhares de opções e, na maioria dos casos, *centenas* de milhares. Empresas como Google, Microsoft, Facebook e Amazon também estavam expandindo suas extensas plataformas de desenvolvimento.

Atualmente, mais e mais consumidores estão adquirindo esses produtos e serviços. Eles passaram a contar com eles e levá-los a seus locais de trabalho. Os departamentos de TI não estão mais controlando seus ambientes, disponibilizando telefones e computadores; a expectativa é de que todos esses dispositivos simplesmente funcionem na rede corporativa do usuário. Sendo assim, o nível também foi elevado para o desenvolvedor que trabalha em empresas. Usuários corporativos esperam produtos como portais para empresas e aplicativos de linha de negócios que

sejam criteriosamente projetados e cativantes, exatamente como os produtos que usam em casa.

Portanto, como desenvolvedores, como lidamos com tudo isso?

Eu tenho um pressuposto simples. Para criar produtos que os usuários amem, é necessário incluir os usuários no processo de criação desses produtos. Admito que muitos poderiam apontar para Steve Jobs como a antítese do que estou sugerindo. Em um artigo de maio de 1998 no *Bloomberg Business Week*, Jobs fez sua famosa declaração:

> É realmente complicado fazer design de produtos utilizando grupos de foco. Muitas vezes, as pessoas não sabem o que elas querem até que você lhes mostre.

Embora parte desse sentimento possa ser verdadeira, acho que devemos ser honestos conosco. Jobs tinha uma habilidade única para entender o que os usuários queriam, e muitos de nós não possuímos essa habilidade.

> Sempre procuramos estar na intersecção entre a tecnologia e as artes liberais para que pudéssemos ter o melhor dos dois mundos e criar produtos extremamente avançados do ponto de vista tecnológico, mas que também fossem intuitivos, fáceis de usar, divertidos, de modo que eles realmente se adequassem aos usuários – os usuários não teriam de ir até os produtos; eles iriam até o usuário.

Não creio que Jobs pudesse ter criado produtos que estivessem na “intersecção entre a tecnologia e as artes liberais” sem compreender o amplo espectro das necessidades dos usuários. Não podemos criar produtos que “vão até o usuário” se não estivermos dispostos a ir até os usuários. Embora a Apple possa ter uma compreensão intuitiva do comportamento humano, muitos desenvolvedores não a têm.

No entanto, acredito que essa compreensão possa ser adquirida com o tempo, e a melhor maneira de adquiri-la é investindo tempo junto aos usuários.

Ao coletar *feedback* e observar seu comportamento, podemos obter esclarecimentos valiosos para criar aplicativos que os usuários irão amar. Qualquer pessoa envolvida no processo de criação de um aplicativo (não apenas os designers) deveria tentar compreender

quais são as necessidades dos usuários para determinar o propósito de um aplicativo. Isso envolve muito mais do que o design da parte gráfica, o código ou a funcionalidade. É toda a equipe (ou somente você) trabalhando continuamente para entender o usuário. Nem todos os problemas de nossos usuários podem ser solucionados por meio de códigos, por mais que eu desejasse; sendo assim, os desenvolvedores devem assumir uma abordagem mais holística.

Essa noção parece pertencer ao senso comum, mas continuo impressionado com a quantidade de desenvolvedores que ainda não investe tempo nisso.

A maior parte de minhas experiências é proveniente do trabalho em um ambiente hospitalar comunitário. É um mundo diferente e único em relação a outros ambientes de desenvolvimento de software; no entanto ainda encontro ali os mesmos tipos de desafios. Em um ambiente hospitalar, os usuários são tratados da mesma maneira que os clientes. Eles fazem uma solicitação por nossos serviços; nós nos reunimos com eles e traçamos nosso plano para ajudar e, então, entregamos um produto (cruzem os dedos!) com base no cronograma acordado.

Pudemos melhorar nosso processo ao implementar as práticas de design centrado no usuário apresentadas neste livro. Ao focar na usabilidade, economizamos tempo e criamos aplicativos que atendem às necessidades de nossos usuários. Embora nosso ambiente de desenvolvimento possa ser diferente do seu, você perceberá que as práticas detalhadas neste livro podem ser modificadas para atender às suas necessidades ou circunstâncias.

Este livro não é uma edição volumosa sobre a história ou sobre o quadro atual da usabilidade. Seu propósito é ser uma coleção de ferramentas e métodos sensatos que você pode começar a implementar hoje mesmo. Essa não é uma fórmula mágica que, quando aplicada, gerará um aplicativo perfeito. O ideal é que você chegue ao final de nossa discussão com suas próprias visões e

ideias sobre como melhorar seu processo de desenvolvimento e reconquistar seus usuários.

Sendo, eu mesmo, um desenvolvedor, tenho ciência de que estamos em um mundo de frameworks, linguagens de codificação e ferramentas de edição incríveis em constantes mudanças. Acrescentar mais passos ao ciclo de vida de desenvolvimento pode parecer assustador.

Contudo os métodos descritos neste livro são essenciais para a criação de um processo de desenvolvimento focado e eficiente. Esses passos, na realidade, irão fazer você economizar tempo e evitar que seus projetos se voltem para a direção errada.

Sei que alguns desenvolvedores avaliam um livro pelo número de páginas, mas este livro foi feito para ser pequeno. Dei o melhor de mim para fazer uma apresentação geral, para que você possa começar rapidamente. Não se esqueça de dar uma olhada na seção “A versão resumida”, no final de cada capítulo. Ela contém uma lista de itens que sintetiza os principais conceitos de cada seção.

Esta é uma época emocionante para um desenvolvedor! Há tantas maneiras de enriquecer a vida das pessoas... Nós temos a capacidade de encantá-las e mudar a maneira que elas interagem com o mundo e umas com as outras. É uma responsabilidade única e desafiadora.

Após a nossa discussão, espero que você tenha um desejo maior ainda de explorar a comunidade de experiência de usuário. Não se esqueça de dar uma olhada no capítulo 11 para ter acesso aos links úteis aos líderes relevantes da indústria e também às publicações e aos produtos que o ajudarão ao longo do caminho.

Vamos começar!

CAPÍTULO 2
O que é Design Centrado no Usuário?

“Facilidade de uso pode ser invisível, mas sua ausência certamente não é.”

– IBM

Acho que o pressuposto mais comum e equivocado que existe, especialmente dentro da comunidade de desenvolvedores, consiste em afirmar que a prática da usabilidade é subjetiva. Esses desenvolvedores acreditam que decisões sobre usabilidade são arbitrárias e podem ser tomadas simplesmente pela aplicação de suas preferências pessoais. Além do mais, muitas dessas decisões são tomadas por motivos que não têm nada a ver com os usuários. É melhor acreditar que o comunicado de hoje do CEO vai parar na página principal do portal da empresa. Quem se importa com o fato de ele ter sido escrito em verde-limão e ter uma pimenta dançante na parte superior? Desse modo, se você estiver desenvolvendo um aplicativo com uma equipe ou em um ambiente corporativo, poderá se sentir desafiado quando tentar implementar um design centrado no usuário.

Talvez você vá se sentir como se fosse o único membro da equipe a se importar com a experiência de usuário. Seus colegas ou companheiros poderão revirar os olhos quando você falar sobre a importância de um bom layout e design. Sei que essa jornada poderá ser longa e solitária, mas não é necessário que seja assim. Há maneiras de disseminar conhecimentos sólidos, centrados no usuário, para desarmar até mesmo seus críticos mais exasperados.

Uma das formas de fazer isso é educar sua equipe ou orientar sua empresa a respeito da importância do design centrado no usuário. Para isso, será necessário entender o que é o design centrado no usuário; e, mais importante ainda, o que ele não é.

DCU não é usabilidade

Sei que meu uso intercambiável de design centrado no usuário e usabilidade pode causar confusão. A usabilidade, também referenciada como fatores humanos, corresponde ao estudo de como os seres humanos se relacionam com qualquer produto. As práticas de usabilidade poderiam ser implementadas em tudo, de uma torradeira a uma maçaneta, ou até mesmo à embalagem de ambos.

A interação humano-computador (IHC) está baseada na usabilidade, mas foca no modo como os seres humanos se relacionam com os produtos ligados à computação.

O design centrado no usuário (DCU) surgiu da IHC e consiste em uma metodologia de design de software para desenvolvedores e designers. Essencialmente, ele os ajuda a criar aplicativos que atendam às necessidades de seus usuários.

Embora isso possa estar um pouco simplificado demais, a figura 2.1 mostra um diagrama que ajudará você a entender a relação entre essas metodologias.

É razoável dizer que a prática do design centrado no usuário garante que sua aplicação mantenha uma boa usabilidade. É esta a questão principal! Ao colocar os usuários no centro de seu processo de desenvolvimento, você eliminará a ambiguidade e chegará ao ponto central de suas necessidades.

Além do mais, há a questão da experiência do usuário (UX, ou User Experience). UX é um termo usado frequentemente para sintetizar toda a experiência com um produto de software. Ela não engloba somente as funcionalidades, mas também o quanto um

aplicativo é cativante e agradável de ser usado. A UX de um aplicativo é maior do que a soma de suas partes.

O design centrado no usuário pode ser implementado para garantir que seu aplicativo proporcione uma ótima experiência de usuário.

Figura 2.1 – A relação entre usabilidade, IHC, DCU e UX.

DCU não é subjetivo

Toda a disciplina da usabilidade e todas as suas metodologias subjacentes representam um conglomerado de várias disciplinas científicas. Por meio da utilização de ergonomia, psicologia, antropologia e de vários outros campos, a usabilidade está fundamentada em conhecimento científico. Ela está longe de ser uma forma de raciocínio subjetiva ou uma conjectura.

O processo de design centrado no usuário funciona *contra* pressupostos subjetivos acerca do comportamento dos usuários. Ele

exige provas de que suas decisões de design são eficazes. Se o design centrado no usuário for usado corretamente, seu aplicativo terá como resultado usuários ativamente engajados. Sendo assim, qualquer decisão de design que leve em conta observar e ouvir os usuários não será baseada em caprichos ou em preferências pessoais.

Como diz o ditado, “os números não mentem”. A prática do design centrado no usuário baseia-se em dados para fundamentar suas decisões de design. Uma maneira de fazer isso é realizando estudos de usabilidade (capítulo 9). Ao observar diretamente os usuários, eliminamos os pressupostos e provamos estatisticamente o que está acontecendo de verdade. Isso nos dá uma fundação mais estável para direcionar nosso desenvolvimento.

Com efeito, os dados coletados por meio do processo de design centrado no usuário devem fazer com que fique mais difícil argumentar contra as alterações necessárias ao seu aplicativo.

DCU não é apenas design

Provavelmente, esse é o equívoco mais comum a respeito do design centrado no usuário. Algumas pessoas (e eu me deparo com isso principalmente entre nossos amigos desenvolvedores) acreditam que os praticantes do design centrado no usuário estão focados somente em estética ou em fazer com que tudo pareça mais bonito. Embora possa ser importante em um aplicativo, a estética não representa todo o cenário.

Focar no usuário é mais do que simplesmente discutir sobre como será a aparência dos componentes ou criar animações rápidas e transições suaves. O design centrado no usuário permite que possamos examinar o quanto um aplicativo é eficiente para atingir o propósito para o qual foi concebido. Como mostrado na figura 2.2, é possível ter um aplicativo incrivelmente bonito, cuja usabilidade seja um pesadelo.

Figura 2.2 – Os designers dessa bicicleta deveriam ter conduzido um estudo de usabilidade!

É claro que o inverso também pode ser verdadeiro. Um estudo de usabilidade pode identificar falhas na interface de usuário (UI, ou User Interface) de seu aplicativo que dificultem a conclusão das tarefas. Nesse caso, a UI de seu aplicativo desempenha um papel bastante significativo para alcançar o sucesso; no entanto seria um erro fazer com que ela fosse o único elemento a ser focado.

DCU não significa perda de tempo ou de dinheiro

Dedicar tempo a práticas de design centrado no usuário que sejam adequadas pode ser uma tarefa difícil. A própria natureza do DCU exige reflexão e observação. Vamos encarar os fatos: se você estiver gastando tempo de desenvolvimento refletindo sobre opções de design, pode parecer que você não está fazendo progressos. Além disso, se sua pesquisa de usabilidade revelar problemas de design, você poderá acabar tendo de descartar esforços anteriores. Isso pode dar a impressão de que você esteja retrocedendo!

O design centrado no usuário exige que perguntemos aos usuários

sobre o que eles não gostam em nossos aplicativos. Às vezes, não queremos ouvir suas críticas ou assumimos que sabemos o que eles dirão. Estar disposto a receber *feedback* significa estar disposto a ouvir reclamações, e ninguém gosta de saber que está fazendo um trabalho horrível. É claro que Sally só estava sendo "construtiva" quando disse que seu aplicativo "não valia nada".

Para evitar esse tipo de crítica, ignoramos nossos usuários e os excluímos. Focamos em finalizar nosso código, na esperança de que todo o resto se resolva.

Olhe, eu compreendo isso. Como desenvolvedores, o conjunto de conhecimentos com os quais devemos nos familiarizar está sempre se expandindo. Novas tecnologias emergem, os padrões dos dispositivos mudam e novos frameworks para codificação surgem diariamente. Em um minuto, você passa a entender todo um conjunto complexo de APIs e, no próximo, está lendo um artigo em um blog sobre como ele está ultrapassado, e que uma novíssima API está sendo usada no lugar!

Entrar no negócio de programação é concordar, para sempre, que você estará aprendendo constantemente; e à medida que as tecnologias se tornam mais complicadas, elas exigirão mais de nosso tempo e investimentos. Com todos esses desafios, a tentação é mergulhar bem fundo em nosso código e deixar de lado qualquer "distração", como, por exemplo, os testes de usabilidade.

Billy Hollis, um desenvolvedor e evangelizador que promove a importância das práticas de usabilidade, afirma que essa resistência constitui um desafio enorme em nosso mercado. Ele sugere que a comunidade de desenvolvedores perde líderes valiosos na área de usabilidade porque essas pessoas não conseguem equilibrar o aprendizado de novas técnicas de codificação com o tempo gasto junto aos usuários. Elas acabam tendo de optar por um ou pelo outro. Sendo assim, a comunidade está cheia de simples desenvolvedores de código dispostos a focar somente no

aprendizado da próxima API:

> Acho que esse é um dos motivos pelos quais percebemos uma resistência tão grande [por parte dos desenvolvedores] em relação ao design. As mesmas pessoas que sobreviveram no ecossistema de desenvolvedores são aquelas que amam tanto codificar que excluem todo o resto.

A questão acerca do design centrado no usuário é que não é necessário escolher entre um e outro. Envolver o feedback dos usuários em seu processo de desenvolvimento pode ser um "ambos/e" muito poderoso. Além do mais, não é preciso ser um *expert* em experiência de usuário para implementar bons princípios de usabilidade.

Hollis compara o processo com aprender a esquiar:

> Quando você sai para esquiar, não está tentando aprender para se tornar um esquiador olímpico. Está tentando aprender a chegar até o final da colina sem cair.

Precisamos deixar para trás o raciocínio segundo o qual, se não estamos escrevendo código, nosso aplicativo não está progredindo. Devemos aceitar que o tempo gasto com nossos usuários é uma parte necessária do processo de desenvolvimento. É tão necessária quanto aprender a escrever código. Parece que alguns desenvolvedores gastam mais tempo decidindo qual framework irão usar do que como planejam proporcionar valor a seus usuários. É como se esses desenvolvedores assumissem que a ideia de seu aplicativo é valiosa por si só.

Se implementado corretamente, o design centrado no usuário, na realidade, pode fazer você *economizar tempo*. Ao compreender as necessidades dos usuários, você evitará equívocos e erros que podem sair caros. Lembre-se de que refazer sua aplicação porque você não atendeu às expectativas de seus usuários também representa um desperdício de tempo!

Gostaria de salientar que a solução de problemas externos ao código exporia você a novas maneiras de pensamento crítico. Afastar-se de seu código e conduzir uma pesquisa ou um estudo de usabilidade podem permitir que você olhe para o problema por um ângulo diferente. Você poderá descobrir que, ao retornar para seu

código, você estará mais focado e direcionado em relação ao que deve ser melhorado.

Pode ser que você deseje conduzir uma pesquisa com usuários, mas isso não parece ser financeiramente viável. Afinal de contas, tempo é dinheiro; sendo assim, a percepção é de que o tempo gasto em qualquer atividade que não seja codificar é um investimento caro.

Gerar evidências a respeito do retorno sobre o investimento (ROI, ou Return on Investment) no design centrado no usuário está fora do escopo de nossa discussão; todavia, gostaria de estimulá-lo a pensar na usabilidade como uma maneira de evitar a *perda* de dinheiro. Resolver bugs no aplicativo depois de lançado e dar suporte aos usuários por meio de um fluxo de tarefas confuso e desestruturado também exige um comprometimento financeiro significativo.

Em um artigo publicado na revista *interactions*, Arnold Lund apresenta um argumento semelhante:

> Uma abordagem alternativa consiste em ver os testes de usabilidade como parte do programa de gerenciamento de qualidade de software e justificá-los por meio da redução de custos que, de outra maneira, incorreriam, caso problemas de usabilidade não fossem eliminados mais cedo no desenvolvimento. Esses incluem os custos de suporte para softwares já instalados e os custos para correção de software, uma vez que já estão instalados.

Se você estiver tendo problemas para convencer a gerência (ou a você mesmo) das vantagens financeiras de adotar as metodologias de design centrado no usuário, considere fazer uma argumentação financeira por outro ângulo, como, por exemplo, o de evitar custos.

DCU não é relatório de bug

Você pode acreditar que já esteja praticando o design centrado no usuário simplesmente por dar aos usuários a possibilidade de submeter um bug ou um problema que se encontre em seu aplicativo.

Embora isso seja admirável e, certamente, algo que você deva continuar fazendo, não confunda relatório de bugs com pesquisa abrangente de usuário.

Ao olhar continuamente para os feedbacks dos usuários na forma de uma lista de itens a serem corrigidos, você nunca chegará à raiz das necessidades de seus usuários.

Suponha que um usuário submeta um relatório de bug para um recurso que não esteja funcionando corretamente. Você poderá ficar tentado a esmiuçar imediatamente seu código, encontrar a origem do problema e corrigi-lo o mais rápido possível.

Se não gastar tempo para questionar o que o usuário estava tentando fazer quando encontrou o problema, você não obterá nenhum esclarecimento significativo para melhorar seu aplicativo como um todo.

Ao explorar e fazer perguntas que não estejam relacionadas a bugs específicos, você poderá descobrir que os usuários estão tentando usar seu aplicativo de uma maneira que você não tinha percebido. Você poderá considerar a reescrita de alguns recursos para tornar os fluxos de tarefas mais claros ou, melhor ainda, a discussão poderia gerar novas ideias sobre como seu aplicativo poderia proporcionar mais valor.

Certa ocasião, recebi uma solicitação de suporte porque uma usuária estava se deparando com um erro sempre que tentava submeter um registro em minha aplicação. A solicitação continha todos os detalhes técnicos, inclusive a mensagem de erro completa. Eu me achava muito esperto por incluir um relatório de bug automatizado em minha aplicação. Sempre que um usuário encontrava um erro, uma solicitação era gerada automaticamente e enviada para mim. Nada de perder tempo conversando com os usuários! Ledo engano.

Durante horas, esmiucei o código tentando descobrir um erro de sintaxe ou de lógica de programação. Olhei todas as minhas

conexões com o banco de dados e revisei até mesmo os próprios bancos de dados. Revisei a mensagem de erro da solicitação automática diversas vezes.

Por fim, sentindo que não estava indo a lugar nenhum, decidi chamar a usuária. Perguntei o que ela estava tentando fazer. No final das contas, o erro estava sendo causado porque ela estava incluindo alguns caracteres inválidos em um campo de comentário em um dos formulários do aplicativo.

Admito que deveria ter me ocorrido que um usuário poderia inserir esses caracteres especiais e que eu deveria ter condicionado meu código a aceitá-los. Porém o maior problema foi o fato de que ela estava tentando usar o campo de comentários genéricos para documentar informações médicas importantes.

Após passar mais algum tempo com a usuária, percebi que devia criar mais campos no formulário para capturar as informações que ela estava tentando documentar. Meu relatório automatizado de bugs não representava todo o quadro do problema. Se eu tivesse perdido a oportunidade de conversar com a usuária e tivesse somente corrigido o problema no código e prosseguido, não teria tomado conhecimento de sua necessidade de ter documentação adicional.

Desse modo, os usuários teriam continuado a usar o campo de comentários genéricos para documentar informações médicas vitais.

É por isso que a totalidade dos feedbacks de seus usuários não deveria ser composta apenas por uma lista de erros em seu aplicativo. Considere o uso de relatórios de bugs somente como uma maneira de ampliar sua estratégia geral de design centrado no usuário.

DCU não é uma distração

Já aconteceu de você estar em uma reunião ouvindo as solicitações de seus usuários e de repente sua mente vagar para o campo dos

sonhos da solução?

- Devo usar um serviço de web ou devo me conectar diretamente com os dados?
- Será que podemos incluir isso no portal de nossa empresa?
- Que linguagem de programação devo utilizar?
- Aposto que posso desenvolver isso na forma de um módulo para nosso produto principal.
- Adoraria usar isso como uma oportunidade para, finalmente, desenvolver um aplicativo móvel.

Todo esse raciocínio tecnológico é bom. No entanto não tem nada a ver com as necessidades dos clientes, pois ainda não coletamos os requisitos do usuário! O design centrado no usuário nos ajuda a manter o foco nas necessidades principais dos usuários. Ele garante que obteremos informações sólidas antes e impede que tentemos fazer o problema se enquadrar à tecnologia.

Admito que haja restrições tecnológicas verdadeiras com as quais temos de lidar, mas os desenvolvedores, com frequência, cometem o erro de endereçar essas questões previamente. O design centrado no usuário nos ajuda a progredir de forma adequada, partindo dos requisitos do usuário em direção à nossa solução tecnológica. É uma abordagem objetiva que garante que realizaremos as tarefas na ordem correta. Discutiremos mais a respeito desse processo no capítulo 4.

Por enquanto, entenda que a usabilidade não representa uma distração. Na realidade, ela atua contra as distrações ao nos ajudar a focar nos aspectos corretos. Ela insere você em uma lógica adequada para que você possa fazer as perguntas certas e desafiar qualquer noção preconcebida.

Em vez de contemplar a tecnologia, aqui estão algumas perguntas que devíamos fazer antes:

- Qual é a origem da solicitação do usuário? Uma solução técnica

poderia resolver esse problema? Talvez o problema seja de procedimento ou até mesmo político. Quem sabe seja uma questão de processo, de fluxo de tarefas ou uma questão educacional.

- Por que o usuário está confuso em relação a essa mensagem? Como ele interpreta seu significado? Devo explicar de maneira diferente?
- Por que o usuário se perde entre essas duas telas?
- Por que ele não percebeu esse aviso? Ele está simplesmente o ignorando? Em caso afirmativo: por quê?
- Por que o usuário está realizando as tarefas na ordem incorreta? Há uma maneira melhor de organizar o layout para garantir que ele as fará da maneira correta?

Um tema recorrente no estudo de usabilidade é perguntar *por quê*. O design centrado no usuário nos ajuda a permanecer superfocado em entender o comportamento do usuário. É um framework que nos ajuda a descobrir a resposta mais eficaz às suas necessidades.

Ao combinar usabilidade, design centrado no usuário e experiência de usuário, você garantirá uma abordagem mais completa para o desenvolvimento de seu aplicativo. Isso exige foco, determinação e até mesmo um pouco de sacrifício. No entanto ignorar esses aspectos de seu aplicativo, especialmente no mercado sempre competitivo de hoje em dia, representa um desserviço a você mesmo e a seus usuários.

A versão resumida

- O mundo da usabilidade é amplo e foca no estudo das interações humanas com *qualquer* produto.
- A interação humano-computador (IHC) é um subconjunto da usabilidade que foca especificamente nas interações humanas com produtos ligados à *computação*.
- Design centrado no usuário (DCU) é uma metodologia usada

por desenvolvedores e designers para garantir que estão criando produtos que atendem às necessidades dos usuários.

- Experiência de usuário (UX, ou User Experience) é um dos vários focos do DCU. Ela inclui toda a experiência do usuário com o produto, incluindo reações físicas e emocionais.
- O DCU não é subjetivo e, com frequência, baseia-se em dados para fundamentar as decisões de design.
- O DCU envolve muito mais do que criar aplicativos agradáveis do ponto de vista estético. O design desempenha uma função importante; contudo, não representa o único foco.
- O DCU, na realidade, pode promover economia de tempo ao ajudar você a evitar erros que podem custar caro.
- O DCU não representa uma distração que nos impede de terminar o trabalho. Ele garante que iremos focar nos aspectos corretos: atender às necessidades dos usuários usando a solução tecnológica adequada.

CAPÍTULO 3
Trabalhando com usuários

“Reunir-se é um começo. Permanecer junto é um progresso. Trabalhar junto é um sucesso.”

– Henry Ford

Sei que envolver usuários pode ser assustador. Vamos encarar os fatos: os usuários têm uma tendência a perturbar nossos processos de desenvolvimento; eles não entendem o que é necessário para criar um aplicativo; e, na maioria das vezes, não têm a mínima ideia do que estão pedindo. Às vezes, suas solicitações não são realistas nem úteis. Como eles poderiam nos levar a algum tipo de descoberta significativa em um projeto de software?

Está ficando cada vez mais evidente que o trabalho de um desenvolvedor vai além dos domínios da codificação. Devemos estar mais sintonizados com as necessidades dos usuários, e a única maneira de fazer isso é passar algum tempo com eles. Devemos orientar construtivamente nossos usuários para que eles possam nos fornecer (independentemente de eles perceberem ou não) as informações de que necessitamos para criar um aplicativo de sucesso.

Isso exige que ouçamos mais, assumindo uma postura observadora e inquisitiva.

E se eu não tiver acesso aos usuários?

Muitos de vocês poderiam estar lendo isso e pensando consigo mesmo: “Não tenho um grupo de usuários com quem trabalho

diretamente.” Talvez você esteja desenvolvendo algo para o mercado de massa, como um aplicativo para smartphone ou um website. Se for esse o caso, talvez você esteja confuso quando começo a falar sobre conquistar os usuários e trabalhar ativamente junto a eles.

Aqui está o meu conselho: se você não estiver desenvolvendo um aplicativo para um cliente ou um grupo de usuários específico, então aconselho você a encontrar um. Isso pode parecer óbvio, mas já vi muitos desenvolvedores começarem a criar um aplicativo assumindo que, assim que começaram a criá-lo, os usuários surgiram. Esses desenvolvedores investem poucos esforços para entender quem poderiam ser esses usuários e quais são suas necessidades.

Afinal de contas, o aplicativo que você está desenvolvendo deveria servir a *alguém*. O truque consiste em encontrar pessoas que personifiquem quem seja esse alguém. Acho que as redes sociais, como o Twitter e o Facebook, são ótimos meios de encontrar amigos e familiares dispostos a fornecer *feedback* ou responder a perguntas.

Admito que o processo não seja tão simples quanto criar um aplicativo para um cliente ou em um ambiente corporativo, mas os princípios de design centrado no usuário continuam sendo válidos. A chave é garantir que os usuários sejam envolvidos, em algum nível, nas opções de design do software. É a melhor maneira de garantir que você esteja criando um aplicativo que as pessoas querem e do qual necessitam.

Alguns desenvolvedores podem limitar suas interações com os usuários e optam por seguir seus instintos. Até certo ponto, isso é notável, mas não é a maneira mais eficiente de tomar decisões sobre o design de seu aplicativo. Isso não quer dizer que não haja evidências de desenvolvedores que usam sua própria intuição para criar produtos de sucesso. Temos visto inúmeros exemplos de

desenvolvedores que foram pioneiros em softwares e serviços revolucionários simplesmente por terem tido a visão e a intuição para decidir sobre o que era necessário. Eles não precisaram de um grupo de foco ou de um amplo estudo de mercado. Eles simplesmente *sabiam* que o aplicativo deveria ser criado.

Lembrei-me de uma citação de Steve Furtick, pastor da Igreja da Elevação da Carolina do Norte, e acho que é a que melhor qualifica minha resposta. *Não compare o filme editado de alguém com seu vídeo dos bastidores.*

Em outras palavras, é fácil olhar para produtos como Google, Facebook, Twitter, Amazon, Groupon (e isso e aquilo) e pensar que a criação de um software tem a ver somente com uma grande ideia. Acreditamos, erroneamente, que, assim como ocorreu com Newton, uma maçã caiu da árvore e acertou a cabeça desses desenvolvedores, dando início a uma revolução digital, como num passe de mágica. Então vejo desenvolvedores esperando que uma maçã lhes caia na cabeça também e penso comigo mesmo: “Sabe de uma coisa? Se eles simplesmente gastassem algum tempo com as pessoas, provavelmente chegariam mais rapidamente a algum lugar.”

Se virmos somente os pontos altos do aplicativo de alguém, mas não virmos nenhum dos erros, será fácil para nós acreditarmos que o desenvolvedor simplesmente teve uma ótima ideia.

Leonardo da Vinci, por exemplo, tinha um caderno de anotações cheio de esboços e desenhos de sua versão definitiva d’*A Última Ceia*. Não foi o caso de simplesmente sentar-se um dia e pintar sua obra-prima. Ele gastou anos fazendo esboços, apagando e redesenhando diferentes ideias e conceitos, como está representado na figura 3.1. A maioria de nós nem tem conhecimento desses esboços anteriores; tudo de que ouvimos falar diz respeito ao quadro final, adorado por milhões de pessoas.

Então, a questão é esta: não confunda o processo de criar um

aplicativo centrado no usuário com uma linha reta de A até B. Podem ser necessários anos de observação e de estudos para conduzir você até o momento decisivo. Seu processo deverá ser uma linha curva de A até B e depois para E, e de volta para A, e poderá dar tantas voltas que você perderá a noção.

Se estiver disposto a gastar tempo fazendo perguntas, sendo curioso em relação ao mundo que o cerca e observando o comportamento dos usuários, você aumentará as chances de obter conhecimentos. Ao longo do tempo, esses conhecimentos farão expandir sua intuição e, muito possivelmente, conduzirão você a um produto de sucesso.

Figura 3.1 – Desenhos anteriores de Leonardo da Vinci para A Última Ceia.

Saber quando ouvir e quando não ouvir os usuários

Só porque estou sugerindo que devemos ouvir os usuários, isso não significa que devemos ouvir *tudo* o que eles nos dizem. Não é como no comércio em que o cliente sempre tem razão. A maioria dos usuários não faz ideia de como a tecnologia funciona. Eles não sabem o que é ou o que não é possível. Às vezes, suas ideias são

simplesmente malucas; no entanto, se forem cuidadosamente orientados, seu conhecimento pode ser extremamente valioso. No final das contas, precisamos aprender a separar o que devemos conservar do que devemos descartar.

Certa manhã, eu estava andando em direção ao quiosque eletrônico de visitantes que eu tinha pedido para que fosse instalado na recepção do hospital. O quiosque permitia que os visitantes localizassem os pacientes que iriam receber visitas. Também continha a localização de vários pontos de interesse. Vi uma mulher e sua filha usando o quiosque e decidi aproveitar a oportunidade para fazer perguntas à mulher a respeito de sua experiência.

> “Oi, meu nome é Travis. Trabalho aqui neste hospital. Gostaria de saber se você conseguiu usar o quiosque eletrônico adequadamente.”
>
> “Ah, sim! Quero dizer, minha filha teve de me ajudar no começo, mas achei muito legal!”
>
> “Ótimo, você conseguiu localizar devidamente o paciente?”
>
> “Sim. Encontrei meu pai. Você sabe o que seria ótimo? Se os pisos se iluminassem e nos conduzissem até o local onde pudéssemos encontrá-lo!”

Isso poderia chocá-lo, mas eu não seria capaz de criar um sistema que fizesse o piso se iluminar. Francamente, se eu pudesse fazer isso, provavelmente estaria morando em uma praia, juntando meus milhões. Era uma ideia absurda. É evidente que a mulher não tinha a mínima ideia de como funcionava a tecnologia ou do que eu seria capaz de proporcionar.

No entanto ela havia tocado em um problema interessante. É difícil, para os usuários, obter informações de um diretório de computador e traduzi-las para o mundo que os cerca. Em outras palavras, eu apresentava o número do quarto e o andar, mas havia a necessidade de uma indicação mais clara sobre a direção a ser tomada a seguir.

Durante o desenvolvimento, eu já havia antecipado esse problema e criado uma série de animações usando pontos e um mapa do andar. De forma muito parecida com a ideia do piso iluminado da mulher, essas animações, mostradas na figura 3.2, conduziam o

visitante até o elevador mais próximo.

Desse modo, a solicitação absurda dessa mulher havia validado meus pressupostos anteriores e fizeram-me refletir sobre o valor das animações. É claro que o piso não estava sendo iluminado, mas um mapa com pontos animados seria a segunda melhor opção.

O que a maioria dos usuários fornece é uma compreensão de seus próprios fluxos de tarefas. Cabe a você explicar como será possível expandir esses fluxos de tarefas com suas habilidades de programação. Educar os usuários e introduzi-los em seu processo é tarefa sua. Ensine a terminologia correta a eles para que possam explicar suas necessidades de maneira adequada.

Ouvindo-os e orientando-os ativamente, você poderá direcionar os usuários para que forneçam as informações que você estiver procurando. Você deve fazer as perguntas corretas e, caso não esteja conseguindo obter aquilo de que necessita, será preciso perguntar-lhes de maneira diferente. Seja persistente.

Figura 3.2 – Diretório do visitante com pontos animados que indicam o elevador mais próximo.

Certa vez, gastei trinta minutos somente com uma pergunta para garantir que havia recebido as informações corretas. Se um usuário estiver fornecendo informações das quais não necessito, explicarei educadamente por que essas informações não serão úteis. Se um usuário estiver dando explicações sobre algo que não entendo, vou dizer isso a ele. Não escondo o fato de que não tenho a mínima ideia de como seu trabalho ou seu processo funciona. Quando baixo a guarda e admito que não tenho todas as respostas, isso não só afasta de mim a pressão, mas também faz com que eu possa me relacionar melhor com o usuário.

No livro *Observing the User Experience* (Morgan Kaufmann), o autor Mike Kuniavsky descreve essa situação como o *modelo do mestre/aprendiz*. Nesse modelo, você trata o usuário como o mestre-artesão, focado em fornecer todos os detalhes necessários acerca do que ele está fazendo.

Isso é particularmente produtivo em indústrias ou trabalhos com os quais você não esteja familiarizado. Se estivesse criando um aplicativo para agendamentos em uma clínica veterinária, faria sentido acompanhar a recepcionista da clínica como se fosse sua sombra.

Eu me deparo com essa situação no hospital o tempo todo. Por mais que eu desejasse ser um médico, mesmo após todos esses anos, ainda não domino uma terminologia suficiente para estrelar como coadjuvante em *Grey's Anatomy*. Todos os dias, aprendo novos fluxos de trabalho, acrônimos (são tantos acrônimos) ou serviços proporcionados pelo nosso hospital. Não tenho vergonha de admitir que não sei do que um usuário está falando. Por meio de uma série de perguntas, faço-o descrever seu fluxo de trabalho de uma maneira que eu possa compreender.

Acho que os programadores, com frequência, são marcados com um estigma de alguém que sabe de tudo ou que tem uma inteligência acima do normal. Às vezes, os usuários podem se sentir

intimidados. Uma ótima maneira de torná-los mais acessíveis é ser o primeiro a admitir que precisa de ajuda. Isso dispõe os usuários a assumir uma postura mais adequada. São eles quem têm as respostas, e não você. O trabalho deles é ajudá-lo a aprender o que eles fazem para que você possa criar um produto melhor para eles. É assim que o design centrado no usuário funciona!

Lidando com diferentes tipos de usuários

Não há uma única solução para ajudá-lo a obter as informações necessárias de seus usuários. Cada pessoa tem uma maneira diferente de lidar com os problemas. Você deve ser flexível e estar disposto a se adaptar a diferentes abordagens e estilos. A seguir estão descritos alguns dos tipos de personalidade com os quais costumo me deparar.

O informante exagerado

Esses usuários gostam de dar informações... muitas informações. Eles vêm para as reuniões com pilhas de papéis e anotações. Copiam você em emails com longas discussões, não oferecendo nenhum contexto sobre o motivo pelo qual você está sendo incluído. Explicam toda a história de seus processos com vários detalhes e incluem até mesmo uma história sobre as primeiras férias no Havaí. Fornecem cópias de anotações que fizeram em todas as reuniões de que já participaram. Telefonam para falar sobre uma ideia para a versão 2, apesar de você nem ter começado a desenvolver a versão 1. Eles têm boas intenções, mas não entendem que você não consegue lidar com a quantidade de informações que eles estão fornecendo.

O desafio com esse tipo de usuário é não desestimulá-lo. Se o problema for o fato de ele fornecer *informações demais*, na verdade, esse é um ótimo problema para se ter. Sabendo disso, é necessário estabelecer limites para as informações que você vai considerar.

Oriente-o acerca das maneiras adequadas de se comunicar com você.

Se você preferir não receber telefonemas, informe-o de que seria melhor comunicar assuntos não urgentes por email. Se ele continuar telefonando, deixe que a secretária eletrônica atenda. Então, de forma educada e direta, responda às mensagens por email. Essa é uma ótima maneira de orientá-lo a usar as formas de comunicação preferenciais.

Se ele quiser começar a falar sobre a próxima versão, explique a ele a importância de permanecer focado na versão corrente. Esses tipos de usuários têm uma tendência a se adiantar, portanto, traga-os de volta, continuamente, para a tarefa atual.

Acho que fornecer atualizações de *status* por email é uma ótima maneira de lembrá-lo do foco. Eu crio compromissos recorrentes no calendário para garantir o fornecimento contínuo de informações atualizadas aos usuários sobre seus projetos. Essa é uma abordagem mais proativa, por meio da qual faço contato com eles antes que sintam a necessidade de vir me perguntar. Também ajuda a reservar um tempo para parar e refletir sobre o que já foi realizado e o que ainda deve ser feito.

É mais provável que o desejo dos usuários de ajudar simplesmente seja o resultado de sua empolgação. Finalmente, eles estão conseguindo ajuda e querem garantir que você terá tudo de que necessita. Como diz meu sogro, “às vezes, é preciso deixar que eles falem”. Seja paciente com eles e tente ser grato pelo fato de estarem dispostos a dar informações. Lembre-se de que, no final das contas, eles são seus clientes e devem ser respeitados por isso.

O obcecado por controle

Pessoas obcecadas por controle querem estar envolvidas em todas as decisões sobre o projeto. Elas impõem sua presença em reuniões e, com frequência, atrapalham as apresentações ou as

discussões. Reclamam quando não conduzem as atividades ou quando o grupo decide tomar uma direção diferente. Resumindo, esses usuários querem dar ordens. Querem dizer a você como tudo será feito.

De acordo com minha experiência, a necessidade de controle é proveniente de uma sensação de insegurança. Se estiver disposto, tente chegar até o centro daquilo que os preocupa. Você poderá tentar conduzir uma reunião particular e fazer com que eles saibam o que você está tentando realizar. Lembre-os de que seu desejo é criar o melhor aplicativo possível. Dê-lhes exemplos de como eles podem ajudá-lo e lembre-os de que vocês estão na mesma equipe!

Escolha suas batalhas e faça-os tomar decisões sobre aspectos que terão pouco impacto. Além do mais, é melhor apresentar opções antes de deixar que eles decidam. Ao limitar as opções, você poderá minimizar sua influência geral.

Às vezes, eles só querem se sentir incluídos e respeitados. Não se esqueça de pedir suas opiniões durante as reuniões e prossiga educadamente quando a situação começar a sair do controle. Seu trabalho é garantir que todos sejam ouvidos. Não tenha medo de fazer o papel de mediador durante as reuniões, caso alguém esteja querendo assumir o controle.

Com mais frequência do que se espera, esses usuários acabam se transformando em seus melhores aliados porque podem ser profundamente conhecedores. No hospital, temos usuários especiais aos quais chamamos de superusuários. São parceiros com quem contamos para atingir melhores resultados. Se você criar aplicativos para o mercado de massa, pense em desenvolver um programa beta para que alguns de seus usuários mais críticos utilizem versões não liberadas ou versões de teste de seu produto.

Para o superusuário, o papel pode vir com responsabilidades adicionais e maior expectativa de envolvimento. Acho que essa elevação no nível de engajamento funciona muito bem para esse

tipo de personalidade.

Diferentemente do modelo mestre/aprendiz, esse modelo é chamado de *modelo de parceria*. Esses usuários têm a habilidade de examinar seus próprios comportamentos de usabilidade (e dos demais) e fazer avaliações do que possa ser necessário. Em vez de ser apenas outro usuário a ser observado, eles podem, na realidade, ajudar no desenvolvimento e na implementação de sua estratégia de design centrado no usuário. Por estarem mais intimamente envolvidos nos resultados do processo, é possível que eles recuem e se tornem mais um membro da equipe.

O Advogado do Diabo

> “Isso não funcionará.”
>
> “Já tentamos isso.”
>
> “Se você alterar o menu, ninguém mais usará isso!”

Os Advogados do Diabo se esmeram para ter suas próprias ideias, mas dizem euforicamente porque suas ideias não funcionarão. Desempenhar o papel de Advogado do Diabo permite a essas pessoas acabar com sua ideia sem que sejam considerados os vilões. Afinal de contas, não são eles que estão dizendo. Eles estão somente argumentando em nome do Diabo.

No livro *As 10 faces da inovação: as estratégias para turbinar a criatividade* (Campus), o autor Tom Kelley explica como incentivar personalidades mais construtivas em sua equipe para minimizar os efeitos do Advogado do Diabo.

Ao fornecer aos usuários (ou aos membros da equipe) papéis detalhados a desempenhar, você estará dando-lhes autonomia para que participem de uma maneira específica. Cada persona será responsável por um aspecto do processo criativo.

Por exemplo, o papel do Antropólogo é ser responsável pela observação dos comportamentos humanos e por relatar isso ao grupo. O Experimentador testa novas ideias e valida os pressupostos. O Polinizador explora outros mercados e outras

culturas e, então, converte suas descobertas em novas ideias.

Você poderá encontrar pessoas em sua equipe que se enquadram naturalmente nessas personas, ou deverá empregar a mesma pessoa para representar várias personas. A chave para o modelo de Kelley é dar a cada membro um personagem ou um papel específico a ser desempenhado. Cada pessoa terá a responsabilidade de representar sua visão, a partir da perspectiva de seu papel. Sendo assim, faz menos sentido que alguém seja contra uma ideia simplesmente porque está desempenhando o papel de alguém que discorda. Esse papel simplesmente não existe. O Advogado do Diabo torna-se marginalizado pelos pontos de vista mais fortes.

Em seu livro, Kelley detalha os papéis e as responsabilidades de cada uma das dez personas, e, embora não possam eliminar totalmente o Advogado do Diabo, elas certamente ajudarão. Afinal de contas, não é a pessoa que está discordando do Advogado do Diabo; é a persona assumida pela pessoa.

> O Advogado do Diabo pode não desaparecer nunca, mas as dez personas podem mantê-lo em seu lugar. Ou dizer a ele que vá para o inferno.

Portanto, em vez de ouvir “Deixe-me fazer o papel de Advogado do Diabo por um momento; se mudarmos o sistema de menu, nossos usuários ficarão confusos e frustrados”, você ouvirá “Deixe-me fazer o papel de Antropólogo por um momento. Venho observando nossos usuários utilizando o sistema atual de menu, e eles já estão confusos e frustrados!”.

Lidando com a negatividade

Trabalhar com usuários nem sempre é fácil, especialmente se eles quiserem focar somente naquilo que está errado. Dos desenvolvedores, exige-se que sejam resistentes: você não pode estar nesse negócio se não aceitar ser criticado. No entanto temos de admitir que a negatividade abaixa nosso moral e reduz a

motivação.

Nossa função é permanecer otimista. Temos de acreditar que podemos fornecer a solução de que nossos usuários necessitam. Devemos acreditar que a resposta está esperando para ser descoberta e, com persistência suficiente, podemos descobri-la. Isso não é o mesmo que dizer que devemos nos iludir ou não devemos ser realistas, mas acho que precisamos trabalhar ativamente contra uma postura negativa.

Para ser honesto, é muito mais fácil fazer uma lista do que está errado do que uma lista do que está certo. Somos meio que programados para pensar dessa maneira. Como desenvolvedores, parte de nosso trabalho consiste em focar nos erros em nossos códigos e em processos que não estejam funcionando. Procuramos e erradicamos as falhas eliminando bugs. Infelizmente, acabamos aplicando o mesmo foco a todas as situações, confundindo um problema com uma coleção de bugs a serem corrigidos. Desse modo, acabamos focando demais naquilo que não está funcionando.

Em vez disso, poderíamos dedicar um tempo para observar o que está realmente *funcionando*.

Deixe-me esclarecer. Não estou sugerindo que procurar erros seja algo negativo. Devemos estar sempre focados em encontrar erros ou identificar tarefas que não estejam funcionando. Isso é essencial no processo de desenvolvimento de software. Contudo acho que podemos equilibrar nosso tempo para incluir a avaliação de aspectos que estejam funcionando. Há muito que aprender com nossos erros, mas há muito que aprender com nossos sucessos também.

No livro *A guinada: maneiras simples de operar grandes transformações* (Best Business), os autores Chip e Dan Heath referem-se a essa mudança de foco como “encontrar os pontos luminosos”. Essencialmente, eles acreditam que a motivação e as

mudanças resultam de se concentrar naquilo que esteja *funcionando*, e não no que esteja *falhando*:

> Pense em um mundo no qual você experimentasse uma onda de gratidão sempre que acionasse um interruptor e a sala se iluminasse. Pense em um mundo em que, após o marido se esquecer do aniversário de sua esposa, ela lhe desse um beijo e dissesse: “Você se lembrou de meu aniversário treze vezes nos últimos catorze anos! Isso é maravilhoso!”.

E se tentássemos mudar nosso foco? E se gastássemos tempo durante nossas reuniões com a equipe para explorar aspectos que foram um *sucesso*? Ao compreender totalmente nossos sucessos, garantimos que nós os reproduziremos em outras áreas de nossos aplicativos.

Por exemplo, se aprendemos que o layout de um design particular incentivou mais vendas em um site de comércio, como poderíamos aplicar esse conhecimento para aumentar a quantidade de reservas em um site hoteleiro? Há paralelos dos quais podemos tirar vantagem? Em caso afirmativo, quais são eles?

E se estivéssemos obtendo feedbacks positivos a respeito de uma alteração feita recentemente em um mecanismo de comentários em nosso site? Se nós somente os lêssemos rapidamente, sorríssemos e voltássemos à correção de bugs, perderíamos as oportunidades de explorar o que estivesse funcionando para nossos usuários.

Novamente, não estou o incentivando a ser um otimista iludido, que só ouve feedbacks positivos, mas acredito poder afirmar que feedbacks positivos proporcionam o mesmo volume de percepções que feedbacks negativos.

Além do mais, quando você reserva tempo para focar na exploração de aspectos que estão funcionando, você não só se lembra de seus sucessos, mas também se abre para novas perspectivas. Uma mente repleta de positividade simplesmente funciona melhor.

Também devemos tomar cuidado com a negatividade que *nós* trazemos à experiência. Já estive em discussões com

desenvolvedores em que eles reclamavam da notória estupidez dos usuários. O design do aplicativo nunca é o problema. A culpa é sempre dos usuários, e nós os acusamos:

"Eles são simplesmente preguiçosos."

"Não importa o quanto você facilite, eles nunca usarão isso."

Já fui, eu mesmo, vítima desse tipo de raciocínio. É fácil pensar assim.

Lee Ross, um psicólogo de Stanford, define isso como *Erro Fundamental de Atribuição*:

> Essencialmente, o erro está em nossa tendência de atribuir o comportamento das pessoas à *maneira como elas são*, em vez de atribuir à *situação em que elas se encontram*.

O Erro Fundamental de Atribuição pode afetar, de forma adversa, a maneira como nos relacionamos com os usuários. Ela nos deixa cegos em relação ao que estamos fazendo de errado.

Por exemplo, digamos que estejamos frustrados porque nossos usuários continuamente se esquecem de fazer logout de nosso sistema corporativo interno de arquivamento. Já os orientamos inúmeras vezes sobre a importância de fazer logout, e já tentamos até mesmo mudar o botão de logout para vários locais na tela para melhorar a visibilidade. No entanto os usuários continuamente deixam suas sessões abertas, criando problemas quando tentam fazer login em outros locais.

Nesse caso, pode ser fácil culpar os usuários por não clicarem no botão claramente visível de logout. Podemos até mesmo tentar medidas disciplinares ameaçadoras pelo não cumprimento dos procedimentos de logout. É claro que esse tipo de raciocínio está incorreto. É óbvio que o sistema está falhando com o usuário, e não o contrário.

Portanto, devemos explorar o que está impedindo os usuários de fazer logout corretamente. Talvez não tenha nada a ver com nosso aplicativo. Quando visitamos e observamos nossos usuários,

percebemos que seu fluxo de trabalho não leva realmente a lembrá-los de fazer logout.

Descobrimos que os usuários se afastam de seus computadores com a intenção de retornar, mas acabam sendo desviados e não retornam após vários minutos, ou até mesmo horas.

Após perceber essa situação, decidimos criar uma função de autologout que é ativada após trinta minutos de inatividade.

É isso que o design centrado no usuário nos ajuda a fazer. Todo comportamento de usuário (seja positivo ou negativo) é uma reação ao nosso aplicativo. Aprender por que nossos usuários estão reagindo da maneira que estão é nossa tarefa. O Erro Fundamental de Atribuição trabalha contra isso ao introduzir pressuposições negativas sobre nossos usuários e distorcendo nossas percepções sobre o que está realmente acontecendo.

Criar experiências maravilhosas de software exige uma boa dose de automotivação. É necessário ter um desejo profundo de fazer o que é certo. Dê o melhor de si para eliminar linguagem e atitudes negativas. Isso fará com que você e sua equipe permaneçam positivos, focados e sigam adiante.

A versão resumida

- Embora trabalhar com usuários possa ser difícil, eles representam um patrimônio inestimável para a criação de um aplicativo de software bem-sucedido.
- Os usuários nem sempre têm razão. Com frequência, eles são terríveis para descrever ou para compreender o que necessitam. Fazer perguntas continuamente e chegar até a raiz do que eles estão pedindo é tarefa nossa.
- Traga os usuários ao seu processo de desenvolvimento e eduque-os para que usem a terminologia correta. Dê-lhes ferramentas para que expliquem melhor suas necessidades.
- Os usuários têm suas próprias maneiras de abordar os problemas. Aprenda a trabalhar com diferentes tipos de

personalidades para que você possa obter o máximo de seus usuários.

- Trabalhe arduamente no sentido de eliminar linguagem e atitudes negativas. Evite o Erro Fundamental de Atribuição, que consiste na “tendência de explicar o comportamento das pessoas pela *maneira como elas são*, em vez de explicá-lo pela *situação em que elas se encontram*”.

CAPÍTULO 4
Planejar

“Metas são sonhos com prazos.”

— Diana Scharf Hunt

Quando nossa equipe começou a implementar práticas de design centrado no usuário no hospital, tornou-se evidente que devíamos ter um plano. Não poderíamos dizer que iríamos começar a colocar os usuários em primeiro lugar e, então, simplesmente continuar a fazer nosso trabalho. Devíamos ter uma documentação, do princípio ao fim, que nos ajudasse a completar todo o processo corretamente.

Também era necessário comunicar nossa visão a respeito daquilo que estávamos procurando realizar. Não só para o hospital, mas também para nós mesmos: Qual era o nosso propósito? Em geral, o que queríamos realizar como equipe? Como agregaríamos valor a nossos usuários e à organização?

Decidimos que um template era necessário – um que nos ajudasse a preencher os espaços em branco e nos conduzisse, de forma eficiente, pelo processo de design centrado no usuário.

Desenvolver aplicativos de linha de negócios para uma organização exige que você exerça várias funções. Em um instante, você está desenvolvendo um aplicativo para ajudar o departamento de recursos humanos a administrar seus funcionários e, no próximo, está desenvolvendo algo para dar suporte ao departamento financeiro com suas faturas. Ao ter um template a ser consultado, nossa equipe é capaz de manter todos os nossos projetos voltados para a mesma direção.

Para um desenvolvedor que não trabalha em uma empresa, é igualmente importante ter um plano estabelecido antes de começar a codificar. Você também deve investir tempo para pensar no que está tentando realizar com seus aplicativos, mesmo que os esteja criando somente por diversão.

Essencialmente, ter e documentar um plano estratégico garante que você criará aplicativos de uma maneira padronizada. Um plano evita que você se esqueça de recursos importantes ou de solicitações feitas pelos usuários.

Em nossa equipe, temos um processo documentado que começa com um planejamento, prossegue para implementação e testes, e termina com a implantação e a manutenção.

Quando estiver trabalhando com uma equipe, isso será vital, porque, em algum momento, você poderá ser solicitado a ajudar os colegas em seus projetos. Se eles estiverem usando o mesmo processo padronizado para desenvolver suas aplicações, será mais fácil para você analisar suas documentações e começar a trabalhar imediatamente.

Nosso template possui os seguintes componentes principais:

- Definição de missão da equipe
- Detalhes do projeto
- Requisitos de usuário
- Requisitos funcionais
- Diagramas de banco de dados/fluxo de dados
- Imagens de telas de protótipos

No hospital, usamos o template para mostrar aos usuários como funciona o nosso processo. Eles sabem que, quando estiverem trabalhando com um membro de nossa equipe, haverá uma maneira padronizada pela qual realizaremos nossos projetos. Quanto maior a frequência com que nossos usuários trabalharem dentro dessa estrutura, mais familiarizados eles se tornarão com o que é

necessário para desenvolver um aplicativo. De maneira eficiente, o modelo nos ajuda a ajudá-los, e vice-versa!

Há vários termos no mercado para esse tipo de documentação. Nosso template faz parte de um Ciclo de Vida de Desenvolvimento de Software genérico; algumas empresas chamam a isso de Request for Proposal (Solicitação de Proposta) ou Letter of Intent (Carta de Intenções). O conteúdo específico desses documentos está fora do escopo de nossa discussão. Contudo, gostaria de incentivá-lo a dar uma olhada em como outras equipes de desenvolvimento de software documentam seus processos.

Como sei qual é o plano adequado para mim?

Eu compreendo que o template que estou propondo pode não ser adequado ao seu ambiente de trabalho ou ao seu projeto. Tudo bem. O que eu quero que você compreenda é que ter um plano estabelecido antes de começar a codificar é uma ótima maneira de garantir o sucesso.

Comparo isso com a pintura de uma sala. Se estiver procedendo da maneira correta, você não vai simplesmente entrar e começar a espalhar tinta em todas as paredes.

Vamos pensar em todos os passos envolvidos no planejamento da pintura de sua sala de estar:

1. Meça a sala para ter uma ideia da dimensão do espaço e da quantidade de tinta que será necessária.
2. Observe os móveis e os equipamentos de iluminação e estime a quantidade de luz natural que incide na sala.
3. Vá até uma loja de tintas e dê uma olhada nas diversas amostras de cores. Converse com o vendedor e pergunte sobre as diferentes qualidades de cada marca.
4. Traga uma amostra da cor e aplique-a em uma pequena área da

parede. Espere secar e certifique-se de que é a cor que você quer.

5. Remova tudo das paredes, coloque fita crepe nas bordas, conserte qualquer buraco ou defeito na parede e aplique a primeira demão.

Tudo isso vem antes de começar o projeto oficial, que consiste em pintar a sala de estar. Imagine que você assumisse esse mesmo tipo de comprometimento em seus projetos de software antes de começar a codificar. E se você gastasse mais tempo pesquisando em torno do problema e perguntando aos usuários sobre suas necessidades? E se você explorasse o problema desenvolvendo protótipos e criando mockups[1] rápidos ou imagens de tela?

Minha suposição é que você gastaria menos tempo escrevendo códigos desnecessários ou desenvolvendo recursos de que ninguém necessita. É por isso que ter um plano deve ser uma parte crucial de seu processo de design centrado no usuário. É mais do que simplesmente coletar *feedback* dos usuários; é inserir-se em um processo estratégico para garantir que todos os pontos sejam cobertos.

No entanto não estou sugerindo que será algo tão simples quanto escrever um plano e sair escrevendo código. Usando nossa metáfora, desenvolver um aplicativo é como pintar uma sala que muda de formato todos os dias. Com o design centrado no usuário, adquirimos novos conhecimentos continuamente. Alguns desses conhecimentos são tão significativos que mudam o curso do desenvolvimento.

Desenvolvimento de software é um processo dinâmico; é impossível prescrever um fluxo de trabalho específico. Praticar o design centrado no usuário significa que você estará reagindo a descobertas que não haviam sido incluídas em seu plano original. A equipe de design de software da EffectiveUI tem uma ótima maneira de olhar para essa situação:

> Não importa o quanto você acha que compreende o assunto, e não importa o quanto você realmente tenha pensado nos requisitos, ainda há uma boa dose de incerteza nos fatos e nas premissas originais, e uma vastidão imensa do desconhecido continua à sua espera. Assim como em uma batalha, o resultado será determinado tanto pelo que ocorre durante o curso do projeto quanto pelo que ocorreu antes.

Todavia ter consciência disso não significa que não devemos manter um template ou um modelo para os passos que pretendemos dar. No mínimo, você deve criar algum tipo de roteiro para executar seu processo de design centrado no usuário de modo significativo.

Criando uma definição de missão para a equipe

Uma definição de missão para a equipe é uma ótima maneira de definir o tom de seu template. Sem compreender o que sua equipe está tentando realizar, como será possível prosseguir com qualquer projeto? Uma definição de missão tem o poder de lembrar à equipe de seu propósito. Além do mais, ela proporciona uma compreensão fundamental acerca do trabalho que você está tentando realizar. Você pode recorrer à sua definição de missão quando estiver tomando decisões difíceis em um projeto.

Nós decidimos que a definição de missão de nossa equipe deveria ser diferente da definição de missão de nossa organização. Não é que discordássemos da missão do hospital, mas precisávamos de algo mais específico para o que nossa equipe estava tentando realizar. Sendo assim, examinamos a definição de missão da organização e a customizamos para que refletisse melhor o propósito de nossa equipe. Decidimos que nossa equipe seria responsável por:

> Desenvolver soluções simples e inovadoras para permitir que os usuários alcancem os padrões mais elevados no atendimento aos pacientes e nos serviços.

A missão do hospital é prover os mais elevados padrões no atendimento aos pacientes e nos serviços. A missão de nossa

equipe é prover soluções simples e inovadoras para que eles possam cumprir a missão deles.

Ao usar nossa definição de missão quando estamos avaliando nossos aplicativos, podemos nos perguntar:

- Esse aplicativo é simples e fácil de ser usado?
- Esse aplicativo é inovador? Podemos implementar uma tecnologia mais nova para torná-lo melhor ainda?
- Esse aplicativo representa um impedimento ou permite que os funcionários proporcionem o melhor atendimento possível a nossos pacientes?

O template de nosso projeto contém a definição de missão da equipe, bem no começo. É um lembrete do comprometimento da equipe com o hospital e de uns com os outros. Se você trabalhar em uma equipe, em uma organização de qualquer porte, pense em qual deve ser a definição de sua missão.

Definindo seu projeto

Para começar a se voltar na direção correta, você deverá ser capaz de resumir o que está tentando realizar com seu aplicativo: Qual é o propósito? Por que você está desenvolvendo isso? A quem o aplicativo servirá? Como ele agregará valor?

Parece óbvio que seu projeto deva ser capaz de responder a essas perguntas, mas acho válido investir tempo para documentar essas informações. O ato de documentar o que você quer realizar pode ajudar a compreender melhor as necessidades de seus usuários. Também uso esse processo como um marco para que os usuários me deem *feedback*. Algo tão simples quanto o título do projeto pode exercer um efeito dramático no resultado. Um exemplo disso ocorreu em um projeto no qual trabalhei para nossa Equipe de Respostas Rápidas (ERR).

A ERR responde urgentemente a pacientes quando seus sinais

vitais (batimento cardíaco, nível de respiração, pressão sanguínea etc.) estão na “zona de perigo”. Se uma enfermeira da ERR puder responder antes de a situação tornar-se crítica, ela poderá proporcionar um atendimento proativo e salvar a vida de um paciente, assim como aumentar suas chances de ter um resultado clínico de sucesso.

Após as primeiras rodadas de reuniões com os funcionários, decidi que iria oferecer uma solução de banco de dados para que os enfermeiros da ERR documentassem seu processo. Ao coletar informações sobre o que acontece durante uma situação com a ERR, poderíamos procurar maneiras de melhorar o processo e ajudar os enfermeiros a responder no tempo adequado.

Dei o nome Banco de Dados da Equipe de Respostas Rápidas ao projeto.

O cirurgião chefe envolvido no projeto viu o título e me fez uma pergunta:

> “Por que você está chamando esse projeto de Banco de Dados da Equipe de Respostas Rápidas?”
>
> “Bem”, respondi, “um banco de dados permitirá que a equipe documente seu processo. Com esses dados, poderemos gerar relatórios e procurar maneiras de melhorar o tempo de resposta.”
>
> “Isso é ótimo”, disse ele, “no entanto, tenho um problema com isso. Esse projeto deveria incluir mais do que somente a Equipe de Respostas Rápidas. Na verdade, ele deveria se chamar Sistema de Respostas Rápidas. Veja bem, para que possamos responder de forma eficiente, precisamos de um sistema que envolva todos aqueles que fazem atendimento aos pacientes, e não somente a Equipe de Respostas Rápidas. É assim que faremos uma grande diferença.”

A correção do título do projeto permitiu que eu percebesse algo importante. Descobri que, para atender às necessidades dos usuários (e, espero, salvar as vidas dos pacientes), era necessário expandir o escopo do projeto para incluir membros externos à ERR. Eu devia começar a pensar no projeto como um sistema completo, em vez de ser apenas um pequeno banco de dados para uma equipe específica.

Se não tivesse gastado tempo para descrever minha compreensão

das metas do projeto, eu teria cometido um erro de cálculo grosseiro quando começasse o desenvolvimento. Teria criado um aplicativo muito reduzido quanto ao escopo, diminuindo seu impacto.

Nesse caso, o título fez uma enorme diferença na minha compreensão quanto ao projeto.

O template de nosso projeto possui uma seção “Detalhes do projeto” que inclui as subseções a seguir:

- Título
- Descrição
- Lista das pessoas-chave
- Avaliação do impacto

O título do projeto e a descrição devem refletir todo o escopo do projeto. Enquanto o título proporciona uma maneira conveniente de referir-se ao projeto, a descrição deverá sintetizar o que o projeto irá realizar.

A seção sobre pessoas-chave poderá ou não ser necessária, em sua situação. Caso esteja entregando um aplicativo a um grupo específico de pessoas, então aconselharia você a incluir uma lista dos indivíduos que compartilharão com você a responsabilidade pelo projeto. Se o seu projeto for destinado ao mercado de massa, você poderá considerar rever essa seção para referir-se a “Usuários em Potencial”. Essencialmente, você deve reservar um tempo para considerar os tipos de usuários sobre os quais deseja exercer impacto.

Se você tiver pessoas-chave, então a tarefa delas consistirá em fornecer informações necessárias para que você possa criar um aplicativo de sucesso. Ao listar as pessoas-chave, você saberá com quem poderá contar quando tiver perguntas ou quando precisar de esclarecimentos. Isso ajuda a compreender quem está envolvido no processo de design centrado no usuário.

Por fim, a avaliação de impacto descreve o impacto que o

aplicativo exercerá sobre o ambiente no qual será implantado. Em outras palavras, a avaliação deve resumir quem ou o que será afetado como resultado do aplicativo (tanto de forma positiva quanto negativa). É uma ótima maneira de ajudar a manter o foco na recompensa final.

Novamente, isso pode não se aplicar ao seu caso. Em um ambiente de desenvolvimento corporativo, é sempre uma boa ideia considerar como seu aplicativo impactará a organização. Isso ajudará sua equipe a priorizar os projetos.

Se estiver trabalhando por conta própria, você poderá considerar o impacto pessoal do projeto: O que você espera ganhar ao trabalhar nesse aplicativo? Como esse projeto o ajudará a melhorar suas habilidades? O que você espera ganhar com seu trabalho?

Você também poderá considerar rever a avaliação de impacto após o término de cada projeto. Dessa maneira, será possível verificar se o aplicativo teve o impacto que você estava esperando. Considere fazer com que os usuários revisem a avaliação de impacto antes de iniciar o projeto, se possível, para garantir que sua visão esteja de acordo com a deles.

Coletando requisitos de usuário

Coletar requisitos de usuário representa a parte mais importante de sua estratégia centrada no usuário. Os requisitos de usuário estabelecem a base para os demais passos do processo de design centrado no usuário. Sem requisitos de usuário definidos adequadamente, será impossível prosseguir na direção correta.

O processo de coletar requisitos de usuário exige que você tome as solicitações abstratas dos usuários e as converta em necessidades significativas. Documentar essas necessidades exige que você sintetize o que é necessário. Ao mostrar os requisitos a seus usuários, você poderá garantir que entendeu corretamente suas necessidades.

Mostrar aos usuários suas solicitações por escrito é uma ferramenta poderosa. Muitas vezes, já fui corrigido ao mostrar-lhes minha lista resumida contendo seus requisitos. Eles dizem “Sim, vejo que você pensou que era isso que eu precisava, mas, na verdade, eu preciso de algo diferente”. Documentar os requisitos do usuário me fez economizar horas incontáveis de tempo perdido em desenvolvimento e evitar dores de cabeça.

Esclarecendo melhor, requisitos de usuário não são requisitos técnicos. De fato, essa parte do Ciclo de Vida do Desenvolvimento de Software (CVDS) deve evitar soluções tecnológicas. Requisitos de usuário correspondem às necessidades dos usuários, e não uma folha contendo especificações sobre o que você vai entregar.

Coletar requisitos de usuário é tão vital para um projeto bem-sucedido que, com frequência, eu me recuso a escrever uma linha de código sequer antes de ter várias reuniões com usuários para extrair deles os requisitos. Sem uma comunicação adequada e um entendimento entre o desenvolvedor e o usuário, é impossível criar um aplicativo eficiente. Documentar requisitos de usuário é a melhor maneira para incentivar esse tipo de comunicação.

Se não estiver trabalhando com um cliente ou um usuário diretamente, então eu aconselharia você a usar o tempo para refletir sobre o que usuários em potencial exigiriam de seu aplicativo. Colegas de trabalho, familiares e amigos podem ser um excelente recurso para isso. Considere mostrar-lhes seus requisitos e obter *feedback* para verificar se os seus requisitos de usuário correspondem aos deles.

Durante a pós-graduação, trabalhei em um aplicativo para iPhone, utilizado como prova de conceito, para a organização de festinhas, daquelas em que cada um leva um prato. Um dos primeiros passos dados pela equipe foi fazer um *brainstorming* para levantar todos os requisitos de usuário para o planejamento de uma festa. Itens como contatar amigos, organizar uma lista de pratos e manter os

participantes atualizados acerca dos últimos acontecimentos foram requisitos que começaram a surgir. Após ter concluído nosso *brainstorming*, achei que tínhamos coberto razoavelmente bem todos os aspectos.

No entanto, quando resolvemos conversar com nossos amigos e familiares, tomamos conhecimento de diversos requisitos que não havíamos considerado. O fato é que eu planejava esse tipo de festa de maneira muito diferente de alguns de meus amigos. Os aspectos com os quais eu me importava não eram tão importantes para eles, e vice-versa.

Se você já gastou meses desenvolvendo um projeto, somente para descobrir que criou algo de que seus usuários não necessitavam, é hora de considerar a coleta e a documentação dos requisitos de usuário.

Criando requisitos funcionais

Você pode estar confuso quanto à diferença entre requisitos de usuário e requisitos funcionais. Um requisito de usuário refere-se ao que o usuário necessita; um requisito funcional refere-se ao que o aplicativo necessita. Essencialmente, os requisitos funcionais podem ser vistos como as especificações técnicas do projeto. São as funções individuais que você planeja entregar por meio de seu aplicativo para atender à solicitação do usuário.

Nosso aplicativo de organizar festas possuía uma variedade de requisitos de usuário para planejamento desse tipo de festa. Ao coletar requisitos de usuário, era difícil permanecer focado na discussão dos requisitos sem que nos desviássemos na tentativa de encontrar as soluções. Permanecemos focados em compreender quais eram as necessidades dos usuários para planejar uma festa de sucesso. Quando atingimos o estágio de discutir os requisitos funcionais, finalmente pudemos fazer um *brainstorming* sobre como podíamos atender a essas necessidades com nosso aplicativo.

Percebemos que um requisito consistente de usuário para planejamento de uma festa era fazer com que os participantes fossem notificados quando havia algum tipo de alteração. Por exemplo, um participante poderia mudar de ideia e decidir levar uma sobremesa, em vez de levar um prato salgado. Caso duas outras pessoas estivessem planejando levar a sobremesa, seria importante informar essa pessoa para que ela pudesse considerar rever sua opção. Também era importante que os participantes soubessem quem estava planejando comparecer ao evento. Eles queriam ser notificados caso alguém tivesse cancelado ou se não tivesse planos para comparecer.

A equipe analisou diversas soluções técnicas. Também analisamos outros aplicativos e vimos como eles lidavam com situações semelhantes. Perguntamos aos usuários quais tecnologias eles estavam usando no momento para atender às suas necessidades e questionamos a respeito dos prós e dos contras desses serviços.

Por fim, decidimos que as notificações do iOS eram a melhor maneira de responder às necessidades dos usuários. Desse modo, acabamos listando a capacidade do aplicativo de enviar notificações ao iOS como um requisito funcional, conforme mostrado na figura 4.1.

Figura 4.1 – Aplicativo para organizar festas enviando uma notificação do iOS para informar que um novo participante aceitou o convite para a festa.

Em seu template, considere associar cada requisito funcional ao requisito do usuário que estiver sendo atendido.

No hospital, nosso template inclui colunas para que possamos numerar nossos requisitos de usuário e os requisitos funcionais. Isso nos permite fazer associações entre eles. Você poderá ter um requisito de usuário e vários requisitos funcionais que atendam a essa necessidade. Ao enumerar essas listas, fica mais fácil revisá-las e garantir que associamos um requisito funcional a cada requisito de usuário.

Se você estiver trabalhando para um grupo de usuários ou clientes específicos, então a lista dos requisitos funcionais poderá ajudar a comunicar aos usuários o funcionamento de seu processo de desenvolvimento. Ao virem seus requisitos de usuário convertidos

em uma lista de soluções tecnológicas, isso se transforma em uma ferramenta eficiente para explicar seu papel no processo.

Acho que esse processo também ajuda no caso de usuários que fazem solicitações de última hora. Se um usuário mudar de ideia a respeito de suas necessidades, então o processo terá de ser reiniciado. Você deverá documentar essa nova solicitação e convertê-la em um requisito funcional. Esse processo poderá ser usado para explicar por que um tempo extra será necessário para acrescentar a nova solicitação ao projeto. Em mais de uma ocasião, já aconteceu de usuários decidirem contra a adição de um novo recurso, após terem visto como isso afetaria a entrega de outros requisitos funcionais.

Documentando modelos de dados e de fluxo de trabalho

Se o aplicativo que você está desenvolvendo baseia-se em um conjunto ou em vários conjuntos de dados, incluir diagramas que mostrem como os dados serão estruturados, como eles serão obtidos e transferidos, é uma boa ideia. Ferramentas como o Microsoft Visio oferecem templates para a criação de diagramas de bancos de dados e de fluxos de dados.

Um *diagrama de banco de dados* que represente a estrutura e a organização de seu banco de dados pode ser uma ferramenta útil. Ele permite ver quais elementos de dados estarão disponíveis e como você poderá acessá-los. Eu costumo manter uma cópia impressa do meu diagrama de banco de dados em minha mesa. Sei que esse processo é um pouco analógico, mas acho mais fácil do que vasculhar meu software de banco de dados para procurar um campo em particular. Obviamente, se você tiver um banco de dados extremamente grande, imprimir um diagrama não será uma opção. Se esse for o caso, você poderá considerar um diagrama modificado que foque nos elementos que você planeja utilizar em seu aplicativo.

Outra ferramenta útil consiste no *diagrama de fluxo de trabalho* ou *diagrama de fluxo de dados*. Eu uso diagramas de fluxo de trabalho em projetos que possuem uma sequência complicada de passos para realizar uma ação ou quando várias pessoas estão envolvidas para alcançar um resultado. Diagramas de fluxo de trabalho permitem considerar todos os passos necessários para que um usuário ou grupo de usuários complete uma ação.

Pode ser necessário algum tempo para criá-lo, mas o diagrama evitará que você tenha de administrar todo o fluxo de trabalho do aplicativo somente contando com a memória. Mesmo algo tão simples quanto uma máquina automática de vendas poderá ter vários caminhos e condições a serem considerados, conforme representado na figura 4.2. É fácil deixar um passo de lado ou esquecer-se de um caminho crítico.

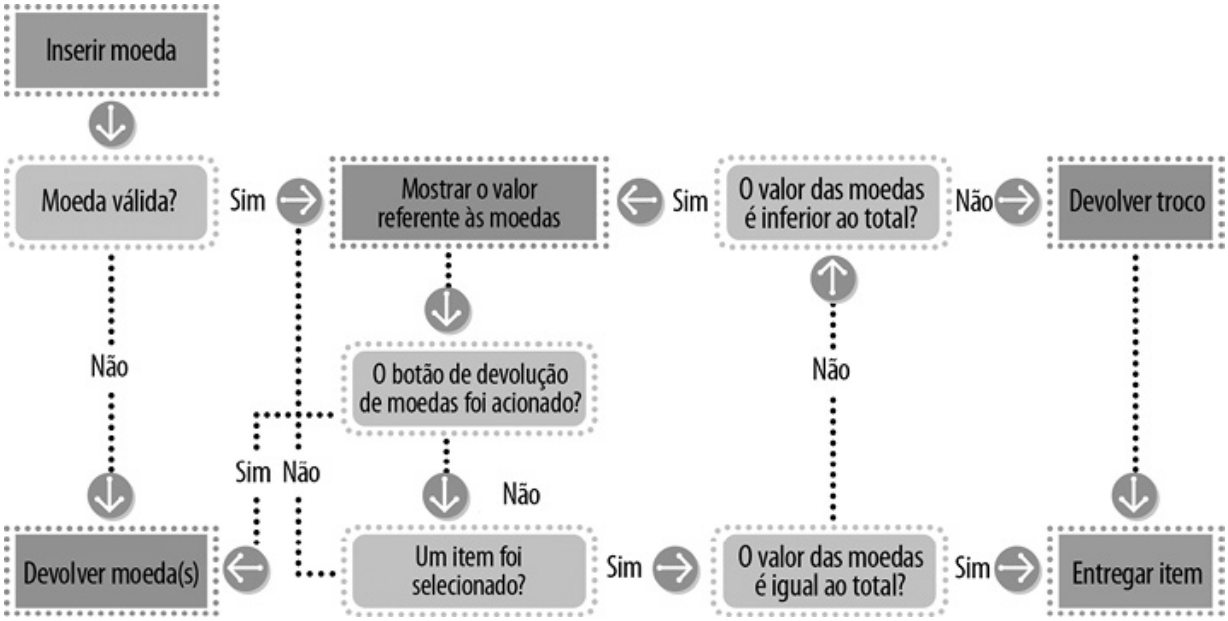


Figura 4.2 – Diagrama de fluxo de trabalho para uma máquina de venda automática.

Os diagramas de banco de dados e de fluxo de trabalho podem ser tão simples ou tão complexos quanto se queira. Obviamente, quanto mais detalhes, melhor, mas prefiro um diagrama limitado a não ter nada.

Documentando protótipos

Os protótipos podem ser ferramentas úteis para dar suporte ao processo de design centrado no usuário. Os protótipos permitem que você converta os requisitos de usuário e os requisitos funcionais em algo mais tangível. Como os usuários poderiam saber o que você pretende entregar se eles não podem ver?

Acho que o maior desafio que os desenvolvedores enfrentam com protótipos é que eles acabam com a grande revelação — aquela sensação de euforia inconfundível que você tem imediatamente antes de mostrar ao usuário o produto no qual estava trabalhando. Em certos aspectos, essa euforia é o que nos mantém motivados a continuar a desenvolver nosso projeto em segredo. Se nossa motivação for fazer uma grande revelação, isso fará com que escondamos nossas ideias do usuário até que tenhamos terminado.

No livro *Equipes de software* (Novatec), os autores Brian Fitzpatrick e Ben Collins-Sussman descrevem perfeitamente esse problema:

> Bem no fundo, desejamos secretamente ser gênios. A fantasia definitiva de um geek é descobrir um incrível conceito novo. Você permanece na Batcaverna durante semanas ou meses, trabalhando como escravo em uma implementação perfeita de sua ideia. Então, você "solta" seu software no mundo, chocando a todos com sua genialidade. Seus colegas ficarão atônitos com sua inteligência. As pessoas farão fila para usar seu software. Fama e fortuna serão consequências naturais.

O problema com essa abordagem é que, se os usuários não forem incluídos no processo de desenvolvimento, haverá uma grande chance de eles ficarem desapontados quando você terminar. A revelação dramática não deveria ser nosso motivador. Em vez disso, devemos ser guiados pelo desejo de fazer o que é certo, e a melhor maneira de fazer isso é envolver os usuários ao longo de todo o processo.

Discutiremos mais a respeito de protótipos no capítulo 8. Por enquanto, entenda que sua documentação deve incluir imagens de tela e mockups preliminares de seus protótipos.

Revisando sua documentação

Outro aspecto importante quanto a documentar o processo de design centrado no usuário desde o princípio é que você poderá fazer revisões muito tempo depois que o projeto tiver terminado. Ver como seu aplicativo mudou ao longo do tempo pode ser um exercício interessante e reflexivo. Revisar templates e documentação antiga permite ver todo o processo, da concepção à realização. Descobrir e acumular conhecimentos sobre como seu projeto mudou ao longo do tempo consiste em uma ferramenta poderosa.

Pode ser que alguns de vocês já tenham contratos e especificações longos para que seus clientes assinem. Se você não trabalha nesse tipo de ambiente, eu o aconselho a convencer as pessoas-chave a assinar um plano de projeto, mesmo que isso o faça se sentir um pouco desconfortável.

Descobri que nossos usuários não se importam com esse tipo de acordo. A maioria deles, na verdade, gosta de ver um documento que defina claramente um plano que vá atender às suas necessidades — e eles estão dispostos a assinar. Isso faz com que eles se sintam mais como um cliente, e não apenas como um colega de trabalho.

Em um ambiente corporativo de desenvolvimento, acho que é assim que os usuários devem se sentir. É claro que nossa equipe não considera que esse seja um documento de valor legal; é somente uma maneira simples de dar mais importância ao plano documentado.

No Apêndice A, você encontrará o template para planejamento de projetos que usamos no hospital. Sinta-se à vontade para usá-lo como base para desenvolver seu próprio template. Você poderá decidir que é necessário reescrever nosso template para focar em outras áreas. Tudo bem.

No final das contas, seu plano deve atender às necessidades de

sua equipe e de seus usuários. Desde que você o conceba como uma ferramenta para documentar e seguir uma metodologia padronizada, ele deverá provar que tem valor inestimável.

A versão resumida

- Sua equipe deve ter uma definição de missão que seja um reflexo de seu propósito e de seu comprometimento com os usuários e de uns com os outros.
- Considere a criação de um template de projeto que comece com o planejamento, prossiga para implementação e testes, e termine com implantação e manutenção.
- Você deve incluir uma seção de detalhes do projeto em seu template. Os detalhes do projeto devem incluir o título, uma descrição do que é o projeto e uma avaliação de impacto, se for apropriado.
- Seu plano documentado deve incluir uma lista de requisitos de usuário. Essencialmente, essa lista representa as necessidades dos usuários e não deve incluir soluções tecnológicas.
- Os requisitos funcionais são uma lista das necessidades de seu aplicativo. Essa lista foca na tecnologia e em como ela atenderá a cada um dos requisitos de usuário.
- Se o seu aplicativo baseia-se em um conjunto de dados ou em fluxos de trabalho complexos do usuário, certifique-se de incluir diagramas de banco de dados e/ou de fluxos de trabalho em sua documentação.
- Protótipos podem ser uma ferramenta poderosa no processo de design centrado no usuário. Não se esqueça de documentar qualquer imagem de tela ou mockups preliminares de seu protótipo.
- Revisar documentação antiga pode ser um exercício valioso para aprender com erros passados.

---

1 N.T.: As *mockups*, também referenciadas pela forma reduzida “mocks”, são figuras

que mostram como será a aparência de certas partes da interface do usuário; as imagens podem ser feitas em papel ou criadas no computador por meio de softwares gráficos ou específicos para essa finalidade. Em geral, as mockups terão a aparência da interface final, mas sem a sua funcionalidade.

CAPÍTULO 5
Criando um manifesto pessoal

“Uma meta nem sempre foi feita para ser alcançada; com frequência, ela existe somente para servir de alvo.”

— Bruce Lee

Na seção anterior, discutimos sobre a importância de ter uma definição de missão para a equipe. Ter uma visão clara, independentemente de você trabalhar em uma equipe ou desenvolver software por conta própria, é vital para obter sucesso. Afinal de contas, como você poderia seguir na direção correta se ainda não tivesse decidido aonde quer chegar?

Robby Ingebretsen, desenvolvedor e cocriador da Pixel Lab, um estúdio para design de experiência de usuário e desenvolvimento de software, acredita que todos devem ter um manifesto pessoal:

> Não tenho uma definição de dicionário para manifesto à mão. Evitei procurar porque estou disposto a fazer um sequestro (quero dizer, dos significados da palavra!). Penso nela da seguinte maneira: um manifesto consiste em uma declaração sobre o motivo pelo qual o que você faz tornará o mundo um lugar melhor.

Ingebretsen acredita que os desenvolvedores devem ter um foco único e abrangente. Esse foco deverá conter um motivo resumido pelo qual desenvolvemos produtos de software.

No capítulo anterior, discuti sobre como nossas equipes desenvolvem aplicativos para ajudar nossa organização a atingir “os mais elevados padrões no atendimento aos pacientes e nos serviços”. Esse é o foco único de nossa equipe. Se não estivermos realizando isso, então não estaremos cumprindo nossa missão.

Porém aqui está outra constatação: você não só deve ter uma

definição geral de missão, mas deve ter também um manifesto para cada projeto em que estiver trabalhando.

Por exemplo, em vez de dizer “Estou desenvolvendo um aplicativo para viagens”, um manifesto afirmaria, “Estou desenvolvendo um aplicativo que reproduz a experiência e agrega o valor de ter um assistente pessoal de viagens”.

Ingebretsen afirma que a empresa de desenvolvimento de software FiftyThree é um excelente exemplo de como ótimos produtos resultam de um manifesto claro.

A FiftyThree, com base em Nova York e em Seattle, é responsável pelo Paper, o aplicativo para iPad que fez um tremendo sucesso. Somente duas semanas após a disponibilização na App Store da Apple, houve mais de 1,5 milhão de downloads do Paper, colocando-o no topo dos gráficos com a velocidade de um foguete. O aplicativo, que permite pintar, desenhar e fazer esboços, recebeu ótimos elogios da crítica e, em 2012, ganhou o Apple Design Award (Prêmio de Design da Apple).

Em uma entrevista de março de 2012 ao *The Verge*, Georg Petschnigg, cofundador da FiftyThree, falou sobre a missão do Paper:

> O Paper é... onde as ideias começam. Certo? É aonde você vai para – por exemplo – esboçar e escrever, desenhar, colorir, contornar. Se você quiser gastar tempo com suas ideias, o Paper será o lar para suas ideias.

“O Paper é o lar para suas ideias” soa como um ótimo manifesto para mim. Essa é uma visão singular. A FiftyThree não se propôs a criar outra ferramenta de design ou um programa para pintar. Eles desenvolveram um aplicativo, mostrado na figura 5.1, que dá a impressão de ser o lar perfeito para suas ideias. É um manifesto específico, com uma visão clara em mente.

Se você abordar seus aplicativos com um manifesto, do modo como a FiftyThree fez com o Paper, ele começará a moldar seu caminho e sua direção. Ele o ajudará a saber quais recursos devem ser incluídos e, mais importante ainda, quais recursos você deixará

de lado. Ao comparar constantemente seu progresso em relação à sua definição de visão, você garantirá seu rumo na direção correta. Isso conduz ao meu próximo ponto.

Figura 5.1— Paper para iPad, da FiftyThree.

Exercitando restrições

Um erro que vejo os desenvolvedores cometerem o tempo todo, incluindo a mim mesmo, é que muitos de nós permitimos que os aplicativos fiquem sobrecarregados de recursos. Para mim, quando vejo um aplicativo com recursos demais, muitos dos quais eu não quero ou de que não preciso, está claro para mim que os desenvolvedores não possuíam um manifesto ou uma visão. Eles colocaram a tecnologia em primeiro lugar e ficaram focados demais naquilo que poderiam fazer, e não no que deveriam fazer.

Acho que o motivo pelo qual fazemos isso é o fato de, em virtude de nossa própria natureza, amarmos a tecnologia. Queremos exibi-la. Para nós, o valor de nossos aplicativos provém daquilo que eles

podem fazer.

Quando os desenvolvedores exibem seus trabalhos, em que, geralmente, eles focam? Eles falam dos recursos que exibem os elementos que mais fazem com que se sintam orgulhosos e dos obstáculos tecnológicos que foram capazes de superar.

Não acho que há nada de errado em sentir-se orgulhoso de suas habilidades de programação; no entanto é preciso lembrar-se de que isso tem pouco valor para os usuários se não estivermos dando a eles aquilo de que necessitam.

Por exemplo, se estivéssemos criando um aplicativo de mensagens instantâneas, poderíamos ficar empolgados com o fato de termos implementado uma tecnologia de localização por GPS, permitindo que os usuários vejam de onde suas mensagens estão sendo enviadas. Contudo, se enviar mensagens a um colega usuário for uma tarefa árdua, qual será o valor geral de nosso aplicativo? Os usuários podem ficar impressionados com o recurso de localização por GPS, mas, como o aplicativo falha em atender suas necessidades principais, eles decidirão não utilizá-lo.

A expressão “menos é mais” pode vir à mente.

Em minha experiência, muitos desenvolvedores lutam com esse conceito. Não é que eles não queiram que seus aplicativos sejam fáceis de usar — é simplesmente difícil decidir quais recursos são essenciais para a experiência. Com frequência, os desenvolvedores têm problemas com o layout, a organização e com o estabelecimento de prioridade dos recursos. Eles cometem o erro de esconder recursos valiosos que seus usuários desejam para promover funcionalidades das quais sentem orgulho.

Bill Buxton, pesquisador principal na Microsoft Research, tem um axioma fundamental:

> Tudo é melhor para uma tarefa e pior para outras. O truque é saber o que é o que, para que, quando, para quem, onde e, acima de tudo, por quê.

É por isso que devemos ter um manifesto ou algum tipo de

definição de visão para nossos aplicativos. Como desenvolvedores, devemos ter uma visão singular à qual podemos recorrer quando nosso desenvolvimento estagnar por causa de novos recursos ou novas barreiras técnicas. Devemos nos lembrar do que estamos, em última instância, tentando realizar.

Uma maneira de criar um manifesto é ter uma história para contar e conhecer bem essa narrativa.

Construindo uma narrativa

A FiftyThree queria que o Paper fosse um aplicativo que permitisse aos usuários capturar suas ideias. Eles queriam que ele tivesse um apelo amplo e que não fosse limitado somente a artistas profissionais. Eles acreditavam que, quando as pessoas usassem o Paper, suas ideias floresceriam, em vez de ficarem presas pelas limitações ou pela complexidade das ferramentas.

Para isso, a FiftyThree necessitava de uma história que estivesse alinhada com seu manifesto. Eles tinham de entender claramente como o Paper poderia realizar aquilo para o qual havia sido concebido. Os desenvolvedores e os designers tiveram de avaliar recursos e funcionalidades para garantir que faziam sentido no contexto da história do Paper.

Uma narrativa do aplicativo é uma história contínua de como o aplicativo molda as vidas de seus usuários. Os manifestos são afirmações específicas e declarativas, enquanto as narrativas podem ser enriquecidas com cenários detalhados sobre como o aplicativo será utilizado.

Julian Walker, o engenheiro líder responsável pelo Paper, afirma que a FiftyThree não implementou uma técnica específica de desenvolvimento de produto. Em vez disso, eles se basearam em sua história para o Paper:

> Na FiftyThree, não temos nenhuma ideia pomposa sobre desenvolvimento de produtos — não saímos por aí professando nenhuma técnica, mas começamos a chamar o nosso processo de “design baseado em narrativa”.

> Os desenvolvedores sempre têm a tendência de querer dar mais poder ao usuário, especialmente se puderem fazê-lo... facilmente. Fazem afirmações do tipo “O recurso está quase pronto. Tudo o que temos de fazer é acrescentar o botão!”, e isso nem sempre pode se encaixar na história.

Quando você usar o Paper, poderá perceber que a história da FiftyThree não incluía ter uma porção de recursos. Julian é um engenheiro brilhante, e não tenho dúvidas de que ele poderia ter criado uma ferramenta complexa de design. Porém recursos complexos não estavam alinhados com a visão da FiftyThree. Eles parecem estar fora de lugar na narrativa e no manifesto do Paper.

O Paper possui somente seis ferramentas e um conjunto limitado de opções padrão para cores, o que é totalmente intencional. Ao explorar o ato de gerar ideias com desenhos e pinturas, a FiftyThree percebeu que outros aplicativos reduziam a criatividade ao sobrecarregar os usuários com recursos. O Paper, desde então, foi atualizado para incluir um misturador de cores, mas continua evidente que a FiftyThree aborda cada recurso adicional com muita restrição.

A narrativa do Paper estimula a simplicidade dando ênfase à redução. Ao usar o Paper, você acaba focando naquilo que quer desenhar, e não na forma como você quer desenhar. Essa redução na complexidade permite que os usuários se concentrem em sua criatividade, e essa é a chave para o sucesso do Paper.

Por exemplo, digamos que eu esteja trabalhando em uma linha de camisetas para uma oferta promocional. À medida que começo a fazer esboços, o design — ou a narrativa — do Paper me mantém focado no conceito geral por trás de minha ideia. Por exemplo, se estou criando uma camiseta vermelha, o Paper oferece somente um tom de vermelho como opção. Eu escolho essa cor e prossigo.

Outras ferramentas de design iriam me apresentar um selecionador de cores e opções virtualmente ilimitadas de cores. Quando tenho um número muito grande de opções à disposição, começo a focar na escolha da cor vermelha perfeita. Isso não quer

dizer que ter opções seja algo negativo (posteriormente, a FiftyThree acrescentou a capacidade de misturar cores); porém vai contra o manifesto da FiftyThree. Eles querem que eu permaneça focado em minha ideia do design da camiseta, e não na seleção da cor perfeita.

No final das contas, o design da FiftyThree, baseado em narrativa, funciona muito bem e, como resultado, sou muito mais criativo quando uso o Paper. Ao reduzir as opções de cores e oferecer ferramentas limitadas, porém, altamente estilizadas, o Paper é diferente de qualquer outro aplicativo que uso para minhas necessidades criativas. Desse modo, ele se tornou minha ferramenta padrão para explorar novas ideias e conceitos preliminares.

Conforme Petschnigg descreveu, o Paper tornou-se o lar para minhas ideias.

Reflita sobre a história de seu aplicativo: Que elementos devem ser incluídos? Que elementos devem ser deixados de lado? Os recursos oferecidos estão de acordo com a história que você estava tentando contar?

Criando personas

Uma maneira de poder ampliar sua narrativa é por intermédio da criação de personas.

Uma persona é um elemento determinado segundo a personalidade que ajuda você a se lembrar para quem o aplicativo está sendo criado. É uma personagem de ficção que consiste na personificação de seus usuários reais. Para criar uma persona, você deve obter informações de seus usuários do tipo:

- Cite o nome de um de seus produtos preferidos. Por que ele é melhor do que outros produtos similares?
- Qual é o produto que mais deixa você frustrado? Por quê? O que

você faria para melhorá-lo?

- Se você pudesse criar o aplicativo perfeito para ajudá-lo nessa tarefa, como ele seria? O que ele faria?

Ao fazer perguntas como essas, você terá uma compreensão melhor do que motiva seus usuários e de quais são as experiências de que eles gostam. Por exemplo, ouvir um usuário descrever facilidades de que gostam em seu tocador de música pode proporcionar ideias para um aplicativo esportivo que você esteja desenvolvendo. No princípio, pode parecer que esses dois produtos não estejam relacionados. Porém, se você estiver disposto a ouvir, é possível aprender novas maneiras de melhorar a experiência de usuário em seus próprios aplicativos.

Com as informações coletadas dos usuários, você construirá uma persona. Ela deverá incluir detalhes como:

- nome
- idade
- estado civil e quantidade de filhos
- localização
- profissão
- *hobbies* e itens prediletos
- necessidades e frustrações

Também podem ser incluídas imagens de como você imagina que será a aparência da pessoa e qualquer outra informação que possa dar vida à persona. No Apêndice A, você encontrará uma persona de exemplo para um usuário do Paper para iPad.

A persona pode provar ter importância vital em situações nas quais você tem uma distância entre você e o usuário. Um exemplo pode ser visto quando você estiver criando algo para o mercado de massa, como, por exemplo, um aplicativo para smartphone. Caso você não esteja trabalhando diretamente com os usuários, a

persona descreverá o tipo de pessoas que usará seu aplicativo. Se for necessário, converse com usuários em potencial e crie uma persona a partir de suas descobertas. Você pode até mesmo usar essas informações para criar diretamente sua persona (eliminando suas informações pessoais, é claro).

Uma persona é um instrumento que permite profundas reflexões e que ajuda a considerar todos os aspectos de um usuário. Com ela, você poderá mergulhar mais fundo em sua psique e imaginar o que os motiva. Você gastará tempo compreendendo suas frustrações e percebendo o que os deixam felizes.

Se a persona estiver calcada em informações coletadas de usuários, ela poderá produzir um efeito bastante tangível.

Criando cenários

Após obter personas que representem os usuários de seu aplicativo, você poderá dar início ao processo de criar cenários. Os cenários são mini-histórias que refletem situações em que suas personas podem se encontrar. Em um cenário, você deve prestar uma atenção particular na forma como seu aplicativo melhorará a experiência do usuário.

Quanto mais detalhes você atribuir a suas personas, mais fácil será imaginar como elas reagirão a um dado cenário. Como diz o nome, os cenários são como cenas em um filme. A persona é o personagem, e seu aplicativo pode ser visto como o dispositivo para a trama, ou uma maneira de fazer a história avançar.

O ideal é que você crie diversos cenários e variadas situações em que sua persona possa se encontrar com seu aplicativo. Se você for honesto em relação às limitações de seu aplicativo, explorar diferentes cenários poderá ajudar a entender o quanto uma limitação em particular representaria um problema.

Por exemplo, suponha que estamos desenvolvendo um aplicativo para smartphone que ajudará Susan, nossa persona, a encontrar

receitas de baixas calorias. Aqui estão alguns dos cenários apropriados que poderíamos encontrar:

- Susan está em casa. Ela está procurando em seus armários ingredientes para uma receita selecionada. Como nosso aplicativo poderia auxiliá-la nessa tarefa?
- Susan está no parque com seus filhos. Ela quer encontrar uma receita de lasanha para preparar para o jantar. Quais recursos do aplicativo ela usaria para realizar essa tarefa?
- Susan está com uma amiga no smartphone conversando sobre uma receita para um cozido de baixas calorias. Como ela poderia enviar essa receita à sua amiga? Ela poderia realizar essa tarefa sem desligar o telefone?
- Susan está fazendo compras e deseja localizar um determinado tempero usado em uma receita que ela encontrou. Como o aplicativo poderia ajudá-la a encontrar o tempero?

Cenários podem ser tão detalhados quanto necessários para vislumbrar como seu aplicativo responderá. Além do mais, quando você combinar esses cenários com as personalidades ricas de suas personas, será possível avaliar suas decisões de design e decidir se elas atendem às necessidades de seus usuários.

Como parte do processo de design centrado no usuário, você deve reservar tempo para considerar sua narrativa, as personas e os cenários: cada um deles ajudará você a criar um caminho mais claro até seu objetivo. Também poderá ajudá-lo a perceber caso esteja se afastando de sua visão e adicionando recursos desnecessários. Se possível, deixe que os usuários revisem suas personas e seus cenários, e permita que eles forneçam *feedbacks*.

A combinação de um manifesto com uma narrativa detalhada com personas e cenários pode moldar o caminho e a direção de seu desenvolvimento de software.

A versão resumida

- Considere ter um manifesto ou uma definição de visão para seu aplicativo. Um manifesto corresponde ao propósito geral e à visão para seu aplicativo.
- Ao construir uma narrativa, você toma a declaração de seu manifesto e tece uma história rica sobre como seu aplicativo poderá ser utilizado.
- Personas são elementos ficcionais, determinados segundo a personalidade, que personificam seu usuário ideal. Use personas para melhorar sua narrativa e ajudar a contar a história de seu aplicativo.
- Cenários, assim como as cenas em um filme, correspondem a situações específicas em que suas personas podem se encontrar. Ao usar cenários, você poderá explorar como o aplicativo responderá (ou não) às necessidades dos usuários.

CAPÍTULO 6
Criatividade e experiência de usuário

“Bons artistas emprestam, ótimos artistas roubam.”
— Pablo Picasso

A essa altura, vale a pena observar que as sensibilidades artísticas da FiftyThree podem ser um pouco intimidadoras. Não tenho nenhum problema em admitir que parte do sucesso do Paper deve ser atribuído às intuições naturais de design de sua equipe.

O exemplo da FiftyThree não é “Veja! Se você estiver usando design centrado no usuário, poderá criar aplicativos incrivelmente lindos também!”. A questão é que devemos admirar a implementação de sua visão por parte da FiftyThree. No mercado do iPad, há muitos aplicativos que permitem aos usuários desenhar e pintar. A FiftyThree poderia ter criado um aplicativo sem inspiração, usando exatamente os mesmos recursos e tendo a mesma complexidade.

Porém, em vez disso, eles decidiram reexaminar a forma como os aplicativos de seus concorrentes poderiam estar sufocando a criatividade e criaram uma visão singular para o Paper. Mais importante ainda, eles usaram essa visão para criar uma narrativa que os ajudasse a permanecer focados em sua missão. Essa é a lição que aprendemos com eles.

Como seu aplicativo (e, francamente, qualquer outro produto em que você estivesse trabalhando) poderia ser diferente, caso você garantisse esse mesmo nível de dedicação?

O capítulo anterior é um exemplo de foco, e não de criatividade. A

criatividade não poderá ajudá-lo se você não tiver uma visão ou uma narrativa para seu aplicativo. Vemos isso em aplicativos que estão repletos de intenções criativas, mas que deixam a desejar quanto à funcionalidade essencial e ao seu propósito. Esses aplicativos podem ser lindos no que diz respeito à aparência, mas são virtualmente inúteis para nós.

Há pessoas que são naturalmente mais criativas? Certamente. No entanto, já ouvi muitos desenvolvedores dizerem algo como “Não sou criativo” ou “Não sou artista, portanto, minha interface de usuário (UI, ou User Interface) será bem básica”. Fico de coração partido quando ouço frases como essas. Não é preciso aceitar imagens em tons de cinza ou uma fonte Comic Sans como estilo de vida. Não devíamos criar aplicativos que não sejam inspiradores ou que não tenham sido bem pensados. Se estiverem dispostos a trabalhar com um pouco mais de afinco, todos poderão fazer algo melhor.

Nossos usuários estão se tornando um grupo mais experiente e mais exigente. Já se foram os dias em que ficávamos satisfeitos com tempos de carga reduzidos ou com recursos adicionais. Os usuários esperam ter uma experiência rica. Desejam obter aplicativos que tenham sido projetados de forma inteligente e que sejam cativantes.

Atualmente, há serviços como locação de filmes on-demand, gravadores digitais de vídeos (DVRs), quiosques eletrônicos para checkout e Internet banking. As pessoas estão usando esses aplicativos e se tornando muito mais criteriosas quanto ao que querem e o que não querem.

Temos de nos adaptar. Temos de aceitar que nem tudo tem a ver com funcionalidades.

Você poderá encontrar centenas de aplicativos que o ajudarão a executar uma determinada tarefa, mas somente alguns deles farão com que isso seja agradável.

Porém não me entenda mal. Os aplicativos sempre serão um sucesso ou um fracasso de acordo com suas funcionalidades. Se um aplicativo não agregar nenhum valor ou não funcionar para o que foi projetado, você estará em apuros. Não há nenhum pó mágico ou uma UI sofisticada que possa dar um jeito em funcionalidade ruim. Contudo devemos admitir que estética e design correspondem a uma parte bastante significativa da experiência de usuário. Muitos de nós não temos acesso a uma equipe de designers, portanto é preciso investir tempo explorando nossa própria criatividade para alcançar as metas da experiência de usuário de nosso aplicativo.

Ter metas de experiência de usuário

O design centrado no usuário representa mais do que coletar requisitos de usuário e convertê-los em requisitos funcionais. Você também deve levar em conta o tipo de experiência que o usuário terá enquanto executar as tarefas. Ao investir tempo para definir metas para a experiência de usuário, podemos garantir que nosso aplicativo crie uma experiência que esteja à altura das expectativas dos usuários.

A diferença entre requisitos de usuário e metas da experiência de usuário pode causar confusão. Afinal de contas, uma excelente experiência do usuário com nosso aplicativo não deveria ser um requisito? É claro que sim; porém é melhor ter metas específicas em mente. Por exemplo, podemos considerar algumas das seguintes perguntas:

- Qual a importância do tempo de carga? O usuário estará disposto a esperar pela carga dos itens se isso significar que podemos proporcionar uma experiência mais rica?
- Como os usuários irão interagir com o aplicativo? Eles usarão uma interface sensível ao toque, voz, gestos, teclado e mouse ou uma combinação desses itens? De que modo o tipo de entrada

afetará a maneira de apresentar as informações a eles?

- O aplicativo deve ser divertido de usar? Quero que os usuários fiquem maravilhados e surpresos ou eles esperam algo mais consistente?
- Que tom de linguagem devo usar para me comunicar com o usuário? Será cômico, profissional, incentivador ou cheio de vitalidade? A linguagem escolhida está consistente em todo o aplicativo?

Quando abordam um aplicativo, os usuários esperam um tipo específico de experiência. Um site de aprendizado infantil passaria a impressão de ser estéril e distante, caso usasse uma linguagem profissional. No entanto, se um site de banco estivesse cheio de cores primárias fortes e extravagantes, os usuários pensariam duas vezes antes de deixar seu dinheiro lá.

Isso não quer dizer que não possamos ser criativos e que não devamos testar a sabedoria convencional. Um banco, por exemplo, poderia dedicar uma seção de seu site para mostrar às crianças a importância de economizar dinheiro. Vamos pensar em algumas metas de experiência de usuário para um projeto como esse:

- O design deve incluir cores fortes e apresentar a nova mascote do banco, “Scotty, a Coruja Sábia”.
- A linguagem deve corresponder a um texto para alunos da segunda série do ensino fundamental. Evite jargões do mercado e terminologia bancária.
- Os jogos devem estar disponíveis para corroborar a mensagem de que economizar dinheiro é bom e empolgante.
- Animações e vídeos devem ser disponibilizados para manter as crianças entretidas e envolvidas.
- Inclua atividades que possam ser impressas e compartilhadas offline com a família.

Se não estiver pensando na experiência que seus usuários

buscam obter, você poderá estar fora de sintonia e poderá confundi-los. Reservar tempo para listar suas metas de experiência de usuário o ajudará a criar uma experiência consistente em todo o aplicativo. Isso pode exigir criatividade, e a maioria dos desenvolvedores, sendo do grupo em que o lado esquerdo do cérebro é dominante, tem problemas com esses tipos de tarefas. Vamos explorar maneiras pelas quais podemos nos estimular a pensar de modo mais criativo.

Criatividade exige coragem e trabalho árduo

Acho que alguns desenvolvedores acreditam que design e criatividade sejam futilidades. Eles acreditam que tudo isso são meros detalhes e que o verdadeiro cerne do aplicativo está em seu código. Eles estão errados. A verdade é que criatividade exige uma dose significativa de motivação pessoal e de determinação.

Também exige coragem, especialmente se você não estiver acostumado a se expressar de maneira criativa. É preciso ser vulnerável, experimentar tarefas diferentes e não se incomodar com o fato de se sentir desconfortável.

Admiro profundamente os desenvolvedores que são corajosos no que se refere à criatividade; aqueles que assumem riscos e desafiam a sabedoria convencional, independentemente de terem sucesso ou não. Ser criativo exige muita coragem. Incertezas, coragem e criatividade estão onde a inovação prospera; e, embora o terreno possa ser fértil, é necessário ter determinação para que o que quer que seja possa criar raízes.

A maioria dos artistas concorda que a criatividade resulta de prática, dedicação e do bom e velho trabalho árduo.

No livro *How to Think Like a Great Graphic Designer* (Allworth Press), de Debbie Millman, o famoso designer gráfico Milton Glasser diz à autora o que o levou aos seus quase 50 anos de sucesso criativo:

> Não sei. Apenas fico em minha mesa trabalhando e tentando fazer isso da melhor maneira possível. Também sou um homem bastante persistente: um homem persistente e teimoso. E a recompensa continua sendo a mesma: fazer algo de qualidade, que continue sendo potente e atinja as pessoas. E, é claro, há o puro prazer de fazer. Adoro ir ao meu escritório e trabalhar.

A criatividade exige uma busca incansável pelo design perfeito. Você deve estar preparado para falhar e tentar novamente. É mais do que simplesmente ter uma ótima ideia — é a concretização dessa ideia. Você deve estar disposto a trazer sua criatividade para a luz do dia, para que ela possa ser rigorosamente examinada. Só então você poderá descobrir grandes ideias e descartar as restantes. No livro *Untitled: Thoughts on the Creative Process* (Creative Collective), o autor Blaine Hogan define o que é exigido de nós para superarmos os limites de nossa criatividade:

> Sua visão deverá estar na proporção direta com o trabalho que você está disposto a fazer para dar vida a ela.
>
> Conheço várias pessoas com muitas ideias realmente excelentes, mas somente algumas delas acabam fazendo algo. E conheço menos pessoas ainda que acabam fazendo um ótimo trabalho.
>
> Talento raramente é a questão, caso você esteja se perguntando.
>
> Não, a verdadeira questão é descobrir se estamos dispostos ou não a arriscar nossas reputações para realizar o trabalho árduo, necessário para criar algo maravilhoso, ou se preferimos tomar o caminho mais fácil desvalorizando nossas ideias, regurgitando antigas visões e recriando aquilo que já conhecemos.

Não há motivos para desejar que você seja mais criativo. Esteja disposto a fazer o trabalho árduo, necessário para ter as ideias criativas que você deseja desenvolver.

Pegue um lápis

Uma das maneiras de exercitar sua criatividade é investir tempo para esboçar suas ideias. Sei que muitos se sentem intimidados com a ideia de desenhar, mas garanto que isso traz vários benefícios.

Em primeiro lugar, quando criamos esboços, estamos usando o lado direito de nossos cérebros, a parte responsável pelo raciocínio abstrato e pela intuição. Programar, por outro lado, é uma atividade

fundamentalmente pertinente ao lado esquerdo do cérebro. E, embora o lado esquerdo de nosso cérebro seja ótimo para raciocínios analíticos e lógicos (por exemplo, codificar), ele tem dificuldade para a descoberta de padrões abstratos e novas ideias. Os conceitos que são difíceis de expressar por meio de código ou de raciocínio lógico podem simplesmente aparecer quando você tentar desenhá-los. Apesar disso, muitos desenvolvedores evitam expressar suas ideias dessa maneira, acreditando que é necessário ser artista para utilizar as ferramentas de um artista. No entanto, só porque você não consegue desenhar a *Mona Lisa*, não significa que desenhar não poderá ajudá-lo.

Na conferência de 2002 do Agile UX em Nova York, Jeff Gothelf fez uma apresentação: “Demystifying Design: Fewer secrets, bigger impacts” (Desmistificando o design: menos segredos, maiores impactos). Ele explicou que, se você puder desenhar um triângulo, um quadrado ou um círculo, poderá desenhar praticamente qualquer interface de usuário. Eu concordo. Diferentemente de codificar, desenhar exige um investimento menor de tempo e proporciona a agilidade para transmitir rapidamente as ideias. Não há praticamente nenhum comprometimento. Se uma ideia não estiver funcionando, você poderá jogá-la fora e começar a fazer outro desenho. Diga-me o que é mais fácil jogar fora: um esboço de cinco minutos contendo algumas formas básicas ou um exemplo de código que levou cinco horas para ser feito? Não tenho mais nada a declarar.

Uma das ferramentas mais valiosas que tenho em meu escritório é meu quadro branco. Durante as reuniões, não é incomum que eu use o quadro para comunicar visualmente todo tipo de ideia. Às vezes, eu o uso simplesmente para escrever algumas palavras que ouço quando um usuário está falando. Isso não só lhes prova que estou ouvindo com atenção, mas eles também podem ver o que estou absorvendo como resultado de nossa discussão. O aplicativo começa a tomar forma só pelo fato de isolar algumas palavras.

Também posso usar o quadro branco para desenhar um layout inicial da interface. Faço isso, como você pode ver na figura 6.1, desenhando quadrados, traçando linhas e outras formas básicas, para que o usuário possa ver como estou interpretando o que eles estão me dizendo. Muitas vezes, ao fazer esses esboços preliminares, consigo descobrir áreas em que eu não havia entendido nada do que o usuário estava descrevendo; isso já me fez economizar horas incontáveis de programação de recursos que o usuário não queria.

Certa ocasião, uma usuária ficou tão exasperada com minha incapacidade de compreender que ela pegou a caneta de minhas mãos e começou a desenhar. Só foram necessários alguns minutos até eu finalmente poder entender o que ela estava tentando dizer. No mínimo, você deve pensar em ter ferramentas como canetas, papel ou quadros brancos à disposição. Novamente, as pessoas têm várias maneiras diferentes de analisar e descrever um problema. Você pode preferir não desenhar, mas essa pode ser a ferramenta perfeita para que um de seus usuários possa se comunicar. Explore essas possibilidades junto aos seus usuários e você poderá economizar bastante tempo e evitar frustrações.

Um componente fundamental do processo de design centrado no usuário consiste em explorar maneiras eficientes de se comunicar com os usuários. Esteja disposto a tentar novas opções, caso suas reuniões não estejam sendo malsucedidas, deixando-o confuso. Se os seus usuários não estão lhe dando aquilo de que você necessita, tente outros métodos. Desenhar pode ser divertido e poderá estimular usuários menos entusiasmados a participar.

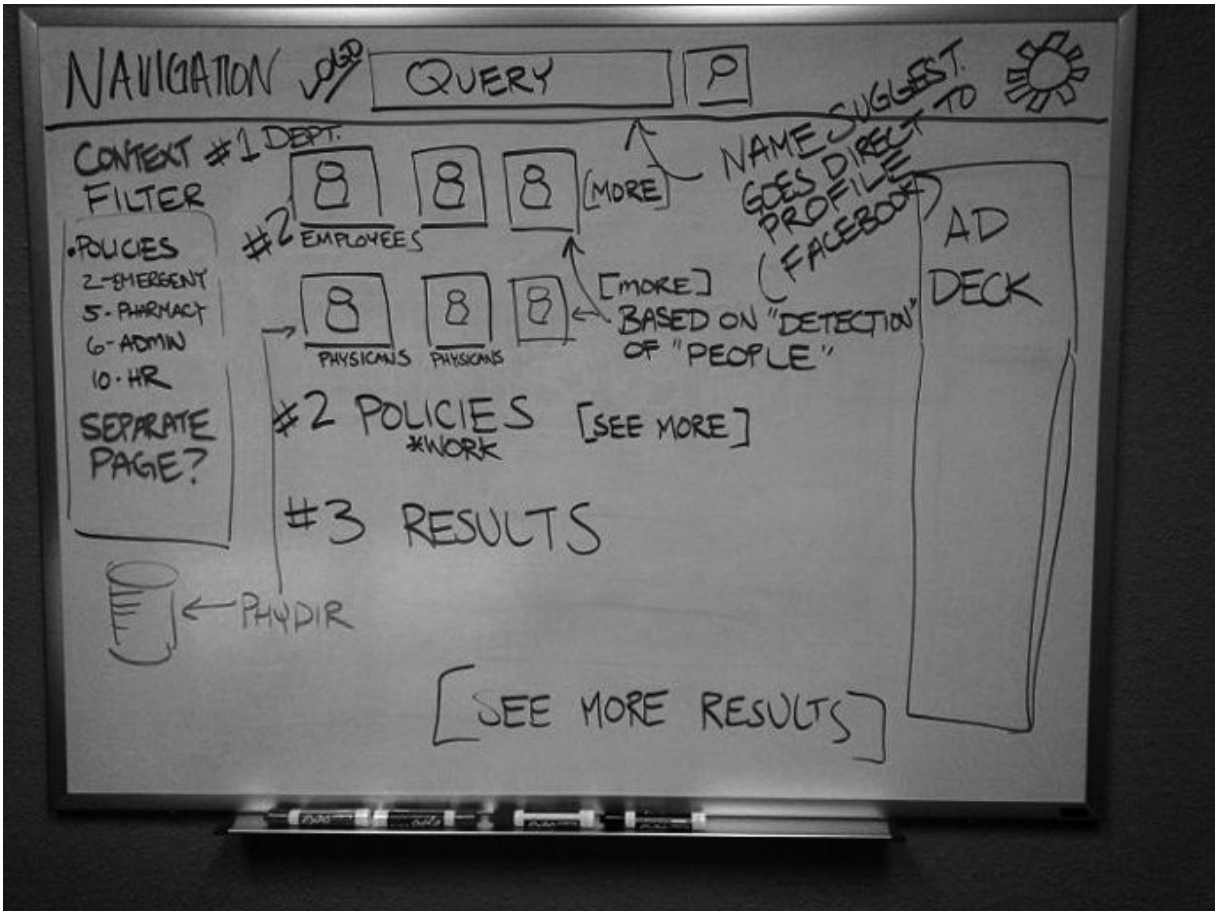


Figura 6.1 — Um dos muitos desenhos em meu quadro branco.

Liberdade de criação

Vamos encarar os fatos: às vezes, trabalhar no mesmo projeto, dia após dia, pode ser exaustivo. Sentar-se à sua mesa e confrontar os mesmos problemas que existiam ontem e no dia anterior podem acabar com sua motivação. Não se esqueça de passar um tempo longe de seus projetos atuais para exercitar seus músculos criativos e continuar a desenvolver novas ideias.

Considere isto: talvez você possa trabalhar em um projeto que não tenha nada a ver com o que você esteja trabalhando no momento. Pode até ser algo diferente de um programa de software. Pode ser um trabalho de carpintaria, de pintura ou de criação musical. Se essas tarefas forem criativas demais, procure explorar novas tecnologias ou frameworks de programação. Talvez você possa

desenvolver um aplicativo para um serviço de web que você nunca tenha usado antes, como Twitter, Facebook, Flickr, Yahoo!, Weather etc.

Daniel Pink, autor de *Drive: The Surprising Truth About What Motivates Us* (Riverhead Books), compartilha uma história que mostra como uma empresa dedica um dia inteiro por trimestre à liberdade de criação. A empresa chama-se Atlassian, e seu cofundador, Mike Cannon-Brookes, teve a ideia de criar o que ele chama de "ShipIt Days" — algo como Dias de Entrega (esses dias costumavam ser chamados de "FedEx Days", mas foram renomeados por causa da violação de direitos autorais). O ShipIt Days da Atlassian permite aos engenheiros de software ter um dia livre para trabalhar naquilo que quiserem e com quem eles quiserem. A única exigência é que os engenheiros entreguem seus trabalhos a seus colegas no dia seguinte, daí o nome.

Pink explica como o ShipIt Days funciona:

> Às 14 horas de uma quinta-feira, o dia começa. Engenheiros, incluindo o próprio Cannon-Brookes, mergulham de cabeça em um código novo ou em um hack[1] elegante — como quiserem, com quem quiserem. Muitos trabalham durante toda a noite. Então, às 16 horas da sexta-feira, os resultados são mostrados ao restante da empresa em uma reunião animada e barulhenta, com grande quantidade de cerveja gelada e bolo de chocolate. A Atlassian chama essas 24 horas de explosões de liberdade e de criatividade de "ShipIt Days" — porque as pessoas devem entregar algo no prazo de uma noite. E os funcionários da Atlassian o fazem. Ao longo dos anos, esse pequeno exercício singular produziu um conjunto de correções de software que, de outra maneira, nunca teriam sido feitas.

Imagine que você e sua equipe pudessem reservar um dia de cada trimestre — ou de cada mês — e focar no que quisessem. A Atlassian percebe o valor que há nessa liberdade de criação e está disposta a investir dinheiro naquilo em que acredita. Tenho certeza de que muitos engenheiros retornam no dia seguinte com ideias ou demonstrações que têm muito pouco valor de imediato. Porém, essa não é a questão. A Atlassian quer estimular uma cultura de empresa que abrace a liberdade de explorar novas ideias e valorize o raciocínio que vai além do lugar comum.

Se você estiver trabalhando em um ambiente corporativo ou em uma empresa de desenvolvimento de software, considere mostrar ao seu chefe artigos que defendam o poder da liberdade de criação. Sugira que sua empresa deve implementar os ShipIt Days. Esteja disposto a organizá-los e oriente os demais a respeito de como isso funcionará. Pode ser que seja exatamente isso que sua empresa necessite para desencadear a inovação e reconquistar seus funcionários.

Se estiver trabalhando por conta própria, reserve tempo para explorar soluções criativas. Vá a um parque e leve sua câmera. Faça um filme com seus filhos e mostre-o à sua família fazendo uma grande estreia. Crie uma conta no Instagram e tire fotos de itens interessantes que você encontrar.

A maneira escolhida para você se expressar criativamente realmente não importa, desde que você tenha um propósito e arrume tempo para fazê-lo.

Entendendo sua meta

Como já afirmei no capítulo anterior, uma visão clara e uma meta para seu projeto são essenciais para o sucesso. Entender sua meta também tem implicações para a exploração criativa. Observe que, quando você realmente compreende o que está determinado a alcançar, pode ser mais fácil afastar aqueles que dizem não. Isso pode ser importante nos estágios vulneráveis da criatividade inicial.

Jeff Weir é um designer de UX na Microsoft, e seu projeto Viscosity é um ótimo exemplo de compreensão de meta.

O Viscosity é um aplicativo vencedor de prêmios baseado em web que permite aos usuários espalhar tintas multicoloridas para criar obras de arte. Em última instância, ele consiste em uma representação digital da natureza viscosa da pintura, conforme mostrado na figura 6.2. Os usuários criaram e compartilharam milhares de projetos de arte usando o Viscosity. Eles também

podem ver um *slideshow* que se expande continuamente, contendo os trabalhos criativos uns dos outros.

Ao avaliar o trabalho de Weir com o Viscosity, pode-se considerar que ele seja supérfluo. Um aplicativo de web que permite que os usuários espalhem tinta por aí... Qual é o propósito disso?

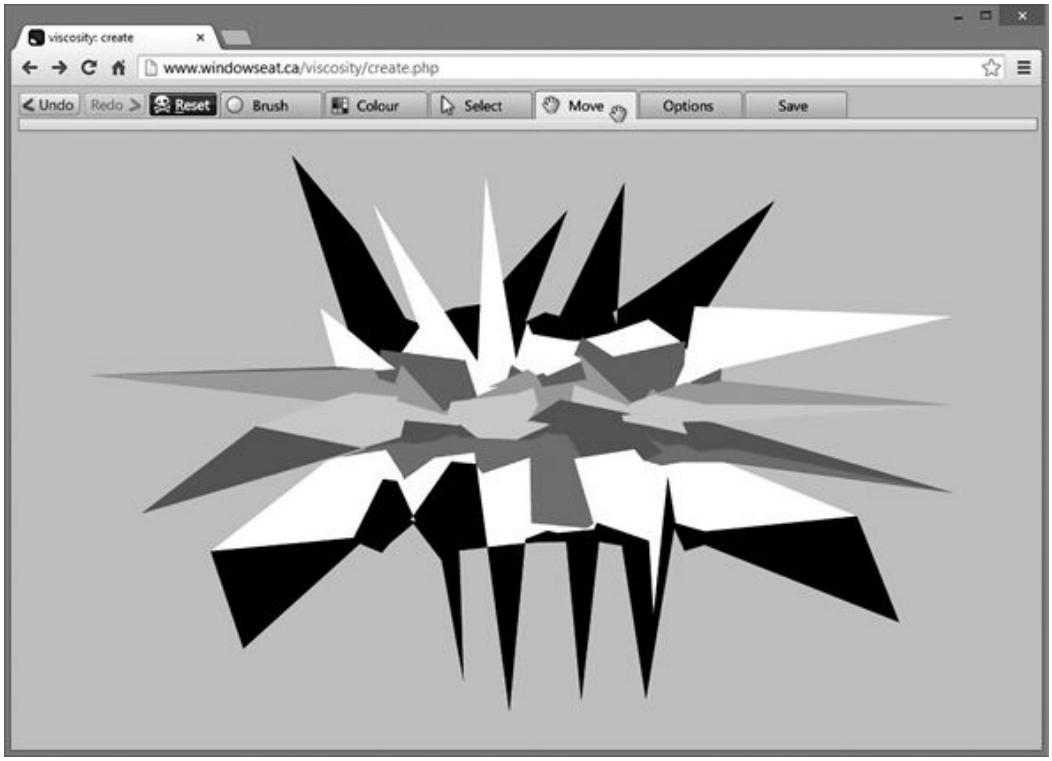


Figura 6.2 — Interface do Viscosity.

Weir realmente não vê necessidade de ter um propósito:

> Muitas pessoas vieram me dizer que eu deveria parar. Qual é o propósito? Como você ganhará dinheiro com esse aplicativo? E nada disso representava uma meta para mim. A meta não era ganhar dinheiro — era criar algo maravilhoso.

Essa história não serve para sugerir que devemos ser altruístas em nossos empreendimentos envolvendo desenvolvimento de software. Afinal de contas, não podemos viver somente de macarrão instantâneo e morar com nossos pais para sempre!

No entanto, para esse projeto, a meta de Weir não era criar algo que fosse rentável. Se essa fosse sua meta, conforme ele mesmo diz, ele teria adotado uma abordagem diferente. A meta era explorar como reproduzir as propriedades naturais da tinta usando um

navegador de web e, como diz ele, “criar algo maravilhoso”.

Quando os críticos disseram que não fazia sentido gastar tempo com o Viscosity porque ele não lhe traria lucros, isso não teve impacto sobre Weir. Ele sabia que esse não era o objetivo do projeto, em primeiro lugar. Ele entendeu o propósito do Viscosity e partiu para o trabalho com uma compreensão clara do que queria realizar. Isso permitiu que ele ignorasse os comentários negativos e visse sua ideia florescer. Esse é o poder da visão!

Ver o Viscosity ganhar vida era a meta de Weir. Ele afirma que a criação pode ser tão importante quanto o produto final. Ele criou o Viscosity porque a ideia lhe interessava e gostou do desafio. Nesse caso, ele não precisou de um plano de negócios ou de uma estratégia elaborada. Para Weir, o Viscosity era um exercício de criatividade:

> A criatividade é como um músculo. É preciso exercitá-la. E, muitas vezes, não tem a ver com fazer certo. Tem a ver com fazer.

Portanto, considere investir algum tempo exercitando ideias diferentes. Afaste-se um pouco de seus projetos correntes e permita-se ser criativo com o mínimo possível de restrições.

Roube (quero dizer, empreste) de outros

E se você estiver cansado e sem ideias? A tela em branco pode ser uma proposição assustadora. Se você estiver sob muita pressão ou não estiver se sentindo inspirado, então não vamos reinventar a roda. Devemos fazer com que olhar o trabalho de outras pessoas em busca de novas ideias se torne uma prática.

No livro *Steal Like an Artist: 10 Things Nobody Told You About Being Creative* (Workman), o autor Austin Kleon sugere que temos de nos libertar das pressões de ser original:

> Se nos libertássemos do peso de tentar ser totalmente originais, poderíamos parar de tentar criar algo do nada e poderíamos abraçar a influência, em vez de fugir dela.

Frequentemente, pressionamos a nós mesmos, sem necessidade, para sermos originais ou inovadores, e acabamos bloqueando

nossa criatividade. Atualmente, temos acesso a milhões de sites, aplicativos móveis, jogos, menus na televisão, leitores de livros digitais, quiosques eletrônicos em shoppings e muitas outras experiências de usuário. Deveríamos estudar incansavelmente essas interfaces e procurar novas possibilidades dentro de nossos próprios aplicativos. No entanto vejo muitos desenvolvedores descartarem suas ideias iniciais rapidamente, com um tom consternado, dizendo “isso já foi feito”. Eles se esforçam e vagam nas terras áridas de suas ideias em busca do oásis de uma ideia original. Muitas vezes, eles nunca o encontram e, por causa disso, nunca chegam a uma descoberta inovadora.

No mundo literário, grandes escritores sabem que, se você quer escrever bem, é preciso escrever. Escreva constantemente e, quando não estiver escrevendo, leia. Você deve ler obras dos grandes mestres e analisar como eles tecem suas histórias. Aprenda com seus estilos e suas prosas e analise continuamente seu próprio trabalho. Grandes escritores não estão preocupados com a influência de outros em seu trabalho; eles permanecem confiantes de que sua própria criatividade irá se sobressair.

No livro *The Writer’s Idea Workshop* (Writers Digest Books), Jack Heffron discute a luta dos novos escritores contra a ansiedade causada pela influência:

> [Os escritores] temem não encontrar sua própria voz. Temem que suas ideias não tenham originalidade. Acho que essa preocupação é exagerada. Muitos aprendizes podem se beneficiar muito mais com esse tipo de influência do que se tentassem evitá-la. Sua própria voz emergirá. Suas próprias ideias surgirão a partir das ideias dos mestres. Nesse meio-tempo, você estará aprendendo a arte e como os bons textos funcionam.

Há muitos paralelos entre escrever um livro e escrever um aplicativo. Ambos os processos exigem uma boa dose de criatividade pessoal e de resolução. Ambos envolvem receber críticas e aprender a aceitar *feedbacks* de maneira construtiva e fazer um trabalho melhor. Muitos desenvolvedores, assim como os escritores, temem que não haja mais ideias originais e começam a

bloquear seu raciocínio criativo.

É isto que digo aos desenvolvedores quando os vejo pensando dessa maneira: “Pode ser que haja muitos apps para Twitter, mas não há nenhum app *seu* para Twitter.” Cada um de nós tem a capacidade de levar nossa experiência pessoal, nossos conhecimentos de programação, as preferências e a base de usuários para um determinado aplicativo. O próprio fato de você estar trabalhando nele torna-o diferente.

Não estou dizendo que você não deva se desafiar para inovar e melhorar os aplicativos que já estão disponíveis; estou dizendo que o fato de algo já existir não deveria ser motivo para que você não explorasse suas ideias.

Se Michael Crichton tivesse decidido que já havia livros demais sobre dinossauros, pode ser que ele jamais tivesse escrito *O Parque dos Dinossauros*. Se a Anne Rice tivesse acreditado que os vampiros estavam ultrapassados, ela poderia não ter escrito *Entrevista com o vampiro*.

Não muito tempo atrás, eu estava trabalhando com um colega, analisando alguns conceitos novos de design para o portal de nossa empresa. Estávamos discutindo várias ideias sobre como os resultados de pesquisas poderiam incluir informações de contato dos funcionários. Estávamos tendo dificuldades em imaginar maneiras de mostrar resultados de pesquisas que proporcionassem esse tipo de informação. Então meu colega sugeriu que víssemos como o Facebook lidava com seus resultados de pesquisa. Também olhamos o Google+ e o Twitter (a figura 6.1 é um resultado dessa discussão).

Observar como essas empresas lidavam com pesquisas e com relevância em redes sociais estimulou o surgimento de novas ideias para nosso projeto. Não estávamos roubando diretamente o que essas empresas haviam feito. Pelo contrário, estávamos aprendendo com o que elas já haviam implementado. Tomamos as

ideias que faziam sentido para nosso projeto e descartamos aquelas que não faziam. Afinal de contas, nosso projeto estava lidando com as informações de contatos profissionais dos funcionários. No entanto pudemos observar paralelos na forma como, por exemplo, o Facebook lidava com informações para contatos pessoais nos resultados de pesquisas.

Também devíamos catalogar o que acharmos interessante. Se estiver usando um aplicativo que possua uma maneira singular de fazer login, faça uma captura de tela e armazene-a para uso futuro. Pode ser que você não esteja trabalhando em uma tela de login no momento, mas poderá estar dentro de alguns meses.

Com o tempo, você deverá ter uma gaveta física ou virtual à qual recorrer quando a criatividade estiver em baixa. Cerque-se de pessoas criativas e observe o que elas estão fazendo. Você pode não ter Vincent van Gogh como seu vizinho, mas há uma infinidade de contas no Twitter, nos blogs, há os livros e vídeos na Internet. Localize esses sites, adicione-os aos seus favoritos e retorne a eles periodicamente: veja o capítulo 11 para obter uma boa lista que o ajudará a começar.

Mantenha sua coleção organizada e estimulante. Não reúna simplesmente um conjunto de imagens de tela, recortes e artigos aleatórios. Kleon afirma que sua criatividade será somente tão boa quanto os elementos que o cercam:

> Minha mãe costumava dizer: “Lixo entra, lixo sai.” Isso me deixava louco. Porém agora sei o que ela queria dizer.
>
> Sua tarefa consiste em reunir boas ideias. Quanto mais ideias boas você reunir, mais opções terá para ser influenciado.

Uma ferramenta que me ajuda a coletar excelentes designs é o site Dribbble (*http://dribbble.com/*). Essencialmente, o Dribbble é um espaço de demonstração e compartilhamento para designers. Artistas e ilustradores postam dizendo sobre o que estão trabalhando e os usuários podem fazer comentários, curtir ou segui-los.

Criei uma conta e, de vez em quando, dou uma olhada no site à procura de itens de que gosto ou que acho interessantes. Tudo o que tenho de fazer é curtir algo, e o item será colocado em uma lista para que eu possa revê-lo posteriormente. É uma gaveta perfeita para ideias. Além do mais, o Dribbble permite procurar qualquer item em que eu possa estar interessado. Outro dia, um amigo e eu estávamos fazendo pesquisas no Dribbble à procura de ideias sobre como deveria ser a aparência de um tocador de áudio em seu aplicativo.

Outro site, o Pinterest (*http://pinterest.com/*), tem uma funcionalidade semelhante, mas contém uma coleção de arte muito mais ampla. Com o Pinterest, você e seus amigos podem "fixar" ("pin") qualquer item da Web que lhes agrade. Pode ser uma camiseta engraçada ou um uso interessante de algum tipo de letra. Pode ser uma frase inspiradora ou uma peça exclusiva de mobília. O Pinterest é uma ótima maneira de colecionar ideias e, por ter um escopo mais amplo que o Dribbble, ele expõe você a outros tipos de influência. Acredite ou não, já usei o estilo de uma cadeira como inspiração para um site no qual estava trabalhando.

Criar experiências de usuário que sejam agradáveis exigirá esse tipo de dedicação, mas não é necessário que seja uma jornada solitária. Ao fazer perguntas a seus usuários, você poderá trazê-los também para a jornada.

Criatividade exige questionamento

Você poderá começar a perceber um tema recorrente em nossa discussão. O design centrado no usuário consiste em uma busca incansável por respostas, e um bom designer ou desenvolvedor nunca para de perguntar *por quê*. Navegar por sites como Dribbble e Pinterest terá pouco impacto se você não estiver reservando tempo para refletir sobre suas descobertas. Quando estiver observando o trabalho de outras pessoas, você deverá fazer

perguntas a si mesmo, tais como:

- Por que esse produto ou serviço foi um sucesso ou um fracasso?
- O que este site está fazendo que seus concorrentes não estão? Essa é a melhor abordagem?
- O que há no design ou no layout desse aplicativo de que eu gosto ou de que não gosto?
- Por que esse produto me deixa frustrado? O que eu faria para melhorá-lo?

Devemos ser capazes de articular os motivos pelos quais gostamos ou não de algo. Quando um design não estiver funcionando, devemos ser capazes de expressar o que está errado; e quando não tivermos a linguagem correta, devemos ler blogs, artigos e livros escritos por aqueles que têm. Muitos desenvolvedores que conheci podem gastar horas reclamando das falhas de design de um produto, mas, quando solicitados a apresentar ideias sobre como melhorá-lo, estranhamente, eles se calam.

Assim como qualquer grande desenvolvedor centrado no usuário, você poderá também fazer perguntas para as pessoas que o cercam. Todos nós já estivemos próximos de alguém que estava frustrado ao usar um telefone celular, um caixa eletrônico, um site ou qualquer outro produto. A frustração de um usuário é uma mina de ouro para ideias criativas. Atrás de cada “Isso é uma porcaria!” ou “Quem criou isso?” há uma riqueza de informações que podemos usar para evitar falhas no design em nossos próprios projetos. No entanto tudo estará perdido se não estivermos dispostos a ouvir. Grandes desenvolvedores e designers estão constantemente fazendo perguntas e observando o comportamento humano.

Quando vou ao centro comercial local ou ao shopping center, gosto de andar por aí e observar as pessoas. Gosto de ver que tipos de smartphones as pessoas estão usando ou posso observar os

consumidores lidando com um terminal de autoatendimento. Às vezes, se eles não parecerem estar ocupados demais, posso parar e perguntar-lhes o que acham do produto que estão usando: “De que você mais gosta nesse produto?”; “Em que você gostaria que ele fosse melhor?”; “Você o recomendaria a algum amigo?”. Certa ocasião, ouvi uma senhora atrás de mim em um restaurante, conversando com uma amiga, explicando por que ela havia devolvido sua televisão à loja (era muito difícil de instalar). Não tenha medo de perguntar a outras pessoas por que elas gostam ou não de algo. Pergunte a seus colegas de trabalho o que eles acham de um novo serviço que descobriram ou de um game que estão jogando. A maioria das pessoas que conheci não se importava em compartilhar suas opiniões. Com frequência, as pessoas se sentem honradas pelo fato de você se importar com o que elas pensam; acho incríveis os conhecimentos que adquiro e que são provenientes desses tipos de discussão.

Outra fonte valiosa de informações sobre usabilidade é seu salão de beleza. Já fiz várias perguntas a todas as mulheres que trabalham no salão que frequento sobre seus smartphones ou seus sites favoritos. Afinal de contas, fico lá sentado esperando, enquanto elas cortam meu cabelo — sobre o que mais conversaríamos? Elas me conhecem atualmente como “o cara dos computadores”. Sempre que vou lá, elas têm novas histórias para compartilhar sobre tecnologias que descobriram.

Mas nem sempre é tão fácil assim.

Devemos nos lembrar de que, como desenvolvedores de software, temos a tendência de sermos bastante técnicos. Nossas percepções são distorcidas. Assuntos técnicos simplesmente fazem sentido para nós, e são quase naturais. Embora possamos compreender a tecnologia com certa facilidade, não acho que isso ocorra com a população em geral.

Para muitas pessoas, a tecnologia é somente um meio para um

fim. Elas não se importam com o fato de ela estar lá ou não. Se a tecnologia servir a um propósito e agregar valor, então ótimo; caso contrário, elas poderiam viver sem ela. Você poderá descobrir que as pessoas comuns têm poucas opiniões sobre produtos e serviços que utilizam. Não é algo sobre o qual elas pensem. Apesar disso, se estiver fazendo as perguntas corretas, você poderá levar essas pessoas a fornecer informações úteis.

Por exemplo, uma amiga minha recentemente comprou um smartphone novo. Nossa discussão transcorreu mais ou menos da seguinte maneira:

> "Ah, você comprou o [modelo mais recente de smartphone]!"
>
> "Comprei o quê?"
>
> "Seu celular novo... o que executa [o sistema operacional mais recente para smartphones]. O que você achou?"
>
> "Ah, sim. Hum, é bom. Eu só queria um celular que tirasse fotos, então esse é o modelo que disseram que eu deveria comprar."

É claro que fiquei desapontado. Eu queria explorar todas as possibilidades que, agora, ela tinha disponíveis. Queria mostrar como ela poderia baixar músicas e vídeos, fazer download de apps ou participar do mais recente jogo online para múltiplos jogadores.

No entanto percebi que ela só queria tirar fotos com seu celular, de modo que mudei minha linha de perguntas:

> "Você gosta mais de tirar fotos com seu celular do que com sua câmera?"
>
> "Sim, gosto. É melhor."
>
> "É mesmo? Por quê?"
>
> "Porque posso compartilhá-las facilmente com meus amigos e familiares no Facebook. Eu adoro o Facebook. Faz com que seja realmente fácil fazer compartilhamentos. Às vezes, porém, isso é um pouco assustador."

Isso nos conduziu a uma longa discussão sobre os méritos das redes sociais e os riscos de compartilhar informações pessoais. Aprendi quais eram os aspectos do Facebook que ela mais valorizava. Percebi que víamos o serviço de maneira diferente e que os recursos que eram importantes para mim eram raramente usados por ela. Era um lembrete importante de que, embora eu não esteja

necessariamente preocupado com privacidade online, outras pessoas acham isso uma proposta desafiadora.

Essa conversa tinha uma ligação direta com os projetos em que eu estava trabalhando naquele momento? Não. Contudo discussões como essas me tornam um programador mais informado e me expõem a perspectivas e valores de outras pessoas. Ao aprender como outras pessoas usam a tecnologia, em algum momento, converto esse conhecimento em novas ideias e melhores experiências de usuário em meus aplicativos. É claro que tudo isso exige que eu saia de minha concha.

Para sermos criativos, devemos estar dispostos a correr riscos. Devemos explorar assuntos que não estejam, normalmente, em nosso círculo de interesses. Devemos experimentar novos pratos e ler livros diferentes — e com diferente não quero dizer livros sobre uma linguagem de programação diferente. Devemos viajar para novos lugares e conversar com pessoas diferentes; devemos fazer novas perguntas e assistir a diferentes tipos de filme. É claro que nossos colegas poderão pensar que é um pouco estranho que tenhamos feito alguns desenhos em nosso mais recente projeto. Poderão nos lançar olhares estranhos quando formos almoçar e eles observarem que estamos com um prato daquele restaurante estrangeiro que acabou de abrir.

Tudo bem, porque criatividade exige coragem. Exige que nos sintamos desconfortáveis, e exige que cometamos erros. Acho que o cineasta americano Woody Allen resume bem isso:

> Se não estiver falhando de vez em quando, é sinal de que você não está fazendo nada inovador.

Se você se vir falhando, e estiver se sentindo até mesmo um pouco envergonhado, isso é um bom sinal de que está no caminho certo para desenvolver suas ideias mais inovadoras.

A versão resumida

- Certifique-se de incluir metas de experiência de usuário em seu processo de design centrado no usuário. Seus usuários esperam mais do que uma funcionalidade básica: eles esperam aplicativos que sejam agradáveis e que tenham sido criteriosamente considerados.
- A criatividade exige coragem e vulnerabilidade de nossa parte. Embora experimentar novidades possa ser arriscado e embaraçoso, é a chave para a descoberta de novas ideias.
- Desenhar e fazer esboços pode exercer um impacto positivo em sua criatividade. Além disso, considere a possibilidade de incentivar seus usuários a desenhar. Isso poderá ajudá-los a transmitir suas ideias.
- Liberdade de restrições é essencial para estimular a criatividade. Afaste-se momentaneamente do trabalho que estiver fazendo atualmente e considere gastar um tempo explorando novos projetos, tecnologias ou interesses.
- Entender completamente sua meta ou sua visão ajudará você a permanecer focado e criativo.
- Se estiver tendo problemas para ter suas próprias ideias, não tenha medo de observar o trabalho de outras pessoas e incorporá-lo.
- A criatividade exige que você faça perguntas constantemente.

---

1 N.T.: Um hack corresponde a uma solução considerada eficiente para um problema, embora nem sempre seja elegante.

CAPÍTULO 7
Princípios de design

“As pessoas ignoram designs que ignoram pessoas.”
— Frank Chimero

Estudar o trabalho de outras pessoas pode inspirar nossa criatividade, porém estudar e entender os princípios de design nos protegerá de cometer erros. Os princípios de design correspondem às leis científicas do mundo da usabilidade, muito semelhantes às leis da gravidade e da relatividade no mundo da física. Os princípios de design são relativamente constantes e foram concebidos ao longo de vários anos a partir do estudo da cognição e do comportamento humano. Eles nos ajudam a oferecer diretrizes baseadas na compreensão do ser humano e na interpretação do mundo que o cerca.

Ter um bom domínio dos princípios de design e de modelos previsíveis pode ajudar você a criticar eficientemente o seu trabalho. É a linguagem perfeita para expressar o que está correto ou o que está errado em um design. Além do mais, você pode usar esses princípios para educar seus usuários, que, com frequência, têm dificuldades em expressar suas necessidades. Ajude-os a encontrar a terminologia correta explicando o significado dos vários princípios de design. Isso proporcionará a ambos uma linguagem comum e correta com a qual trabalhar.

É impossível descrever todos os princípios de usabilidade neste livro, pois muitos deles estão além do escopo de nossa discussão. Considere que essa será uma visão geral de alguns dos princípios

mais comuns.

Princípio da proximidade (Princípio da Gestalt)

O princípio da proximidade é um dos muitos princípios definidos nos *Princípios de percepção da Gestalt*. Embora você deva estudar todos os princípios da Gestalt, acho que o *princípio da proximidade* tem o impacto potencial mais significativo sobre seus aplicativos e exige o menor volume de esforços.

O princípio estabelece que nós percebemos relacionamentos entre objetos que estão mais próximos. Inversamente, objetos que estão mais distantes, aparentemente, teriam menos relação.

Em virtude desse fato, você poderá ver o princípio de proximidade sendo referenciado como princípio de agrupamento. Basicamente, é mais fácil ver padrões de operação quando os itens estão agrupados de acordo com suas funções, conforme mostrado na figura 7.1.

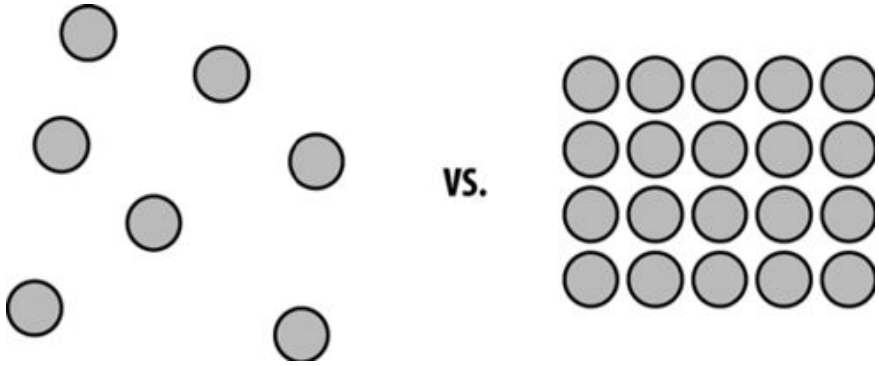


Figura 7.1 — Um exemplo do princípio de proximidade, com o grupo à direita parecendo estar relacionado.

É por isso que o princípio da proximidade pode exercer o impacto mais significativo. Ao simplesmente organizar e agrupar itens de uma maneira que sua função seja descrita, você poderá melhorar significativamente a experiência de usuário com seu aplicativo. Um layout organizado faz com que seja mais fácil aprender a usar seu aplicativo e exige menos do usuário para que ele possa encontrar os

itens. Muitos desenvolvedores ignoram esse princípio simples porque não reservam tempo para refletir sobre como seus aplicativos deveriam estar organizados. Eles acham que o layout de seus aplicativos faz sentido; enquanto isso, seus usuários ficam confusos e frustrados. O princípio da proximidade pode ser usado como um poderoso indicador de que determinados recursos devem estar juntos.

Considere o pacote de aplicativos Office da Microsoft. Em 2007, a Microsoft introduziu a interface Ribbon, que é um agrupamento de funções do Office na parte superior da janela de um aplicativo. Essa interface resultou da confusão cada vez mais crescente que muitos usuários sentiam em relação à localização de determinados recursos do Office. A Microsoft introduziu o Ribbon como uma maneira de colocar funções semelhantes em uma relação de proximidade umas com as outras.

Por exemplo, no Ribbon do Word, apresentado na figura 7.2, as funções que alteram o estilo do texto são colocadas bem próximas, assim como as funções que manipulam imagens, as funções que alteram o layout do documento, e assim por diante. Além do mais, a Microsoft fez o Ribbon ser contextual, de modo que ele se altera conforme o item selecionado no documento. Isso ajuda os usuários ao destacar os recursos mais importantes, de acordo com o conteúdo que estão manipulando.

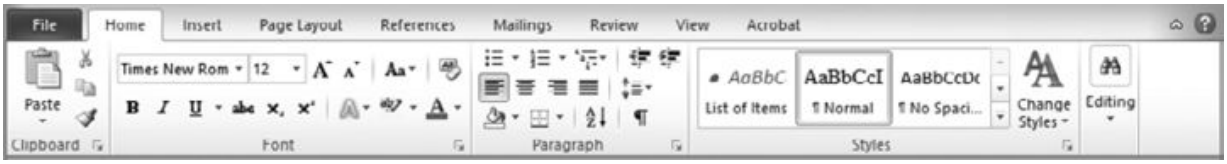


Figura 7.2 — Interface Ribbon do Microsoft Word.

Nada é mais frustrante do que um aplicativo desorganizado. Isso exige que o usuário percorra menus complexos e opções, procurando uma agulha virtual em um palheiro. Esse processo reduz nossa eficiência e acaba com nossa paciência. Organizar um aplicativo por proximidade ajuda os usuários a entender como seu

aplicativo funciona e permite que eles avaliem rapidamente as opções disponíveis.

Visibilidade, feedback visual e proeminência visual

A visibilidade corresponde a tudo o que você utilizar para concentrar o foco visual em um elemento ou uma ação na interface de usuário de seu aplicativo. Há diversas maneiras de fazer isso:

Tipo de letra

Diferentes estilos e tamanhos de texto podem atrair a atenção do usuário.

Opacidade

Ajustar a opacidade de um item ajuda a reduzir ou aumentar sua visibilidade.

Proeminência

Elementos que são maiores que outros terão mais visibilidade, conforme mostrado na figura 7.3.

Status

Indica que o aplicativo está processando uma solicitação ou que recebeu dados de entrada do usuário.

Cor/Contraste

Tradicionalmente, itens com cores mais fortes ou contrastantes atrairão mais atenção.

Figura 7.3 — Exemplo de proeminência, um dos princípios de visibilidade.

O princípio da visibilidade pode ser usado para indicar o status de um aplicativo. Por exemplo, o cliente de mensagens que uso no trabalho possui um ícone que se torna cinza quando não estou mais logado, como mostrado na figura 7.4. Quando faço login, ele se torna verde. Essa pequena mudança ajuda a me manter informado sobre meu *status* durante a utilização do serviço.

Figura 7.4 — Meu cliente de mensagens disponibiliza um indicador verde para informar que estou disponível para um bate-papo no momento.

Outro aspecto do princípio de visibilidade é proporcionar um *feedback* visual. O princípio do *feedback* visual estabelece que os aplicativos devem responder à entrada de dados dos usuários. Em outras palavras, seu aplicativo deve apresentar algum tipo de indicação de que recebeu informações do usuário. Um exemplo simples desse caso seria disponibilizar um ícone giratório ou uma mensagem de “procurando...” quando um usuário submeter uma solicitação de busca. A questão geral do princípio de *feedback* visual é notificar o usuário de que houve uma interação. Sem essa confirmação, o usuário fica confuso a respeito de a sua ação ter sido recebida ou não pelo aplicativo.

A maioria dos aplicativos fornece *feedback* ao usuário. Contudo, já vi algumas implementações precárias desse recurso. Por exemplo, a figura 7.5 mostra como o sistema de catálogo de cursos de minha universidade responde quando procuro uma determinada disciplina.

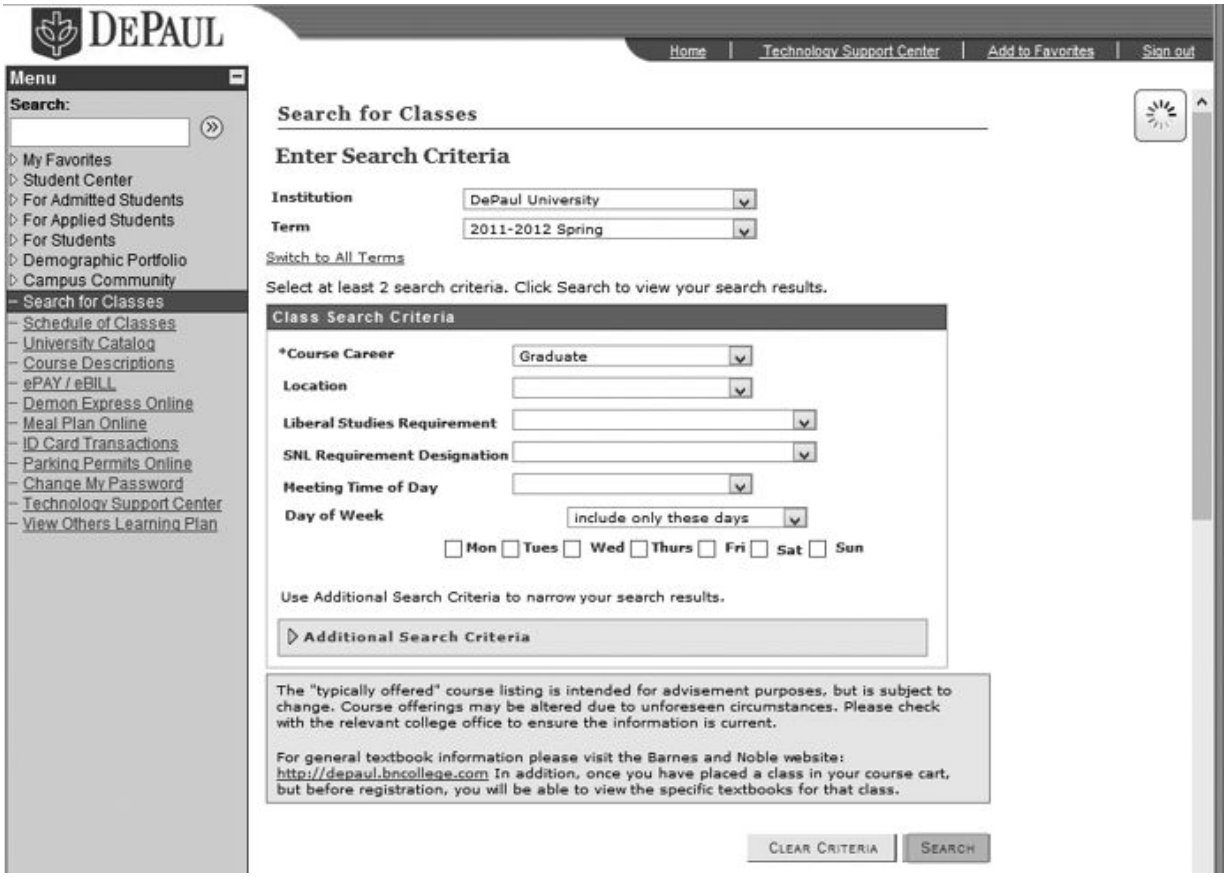


Figura 7.5 — Interface para pesquisas no catálogo de cursos da DePaul University.

No primeiro dia de matrícula, estava convencido de que a ferramenta de buscas não estava funcionando porque eu submetia uma solicitação de busca e nada acontecia. Alguns minutos se passaram até eu perceber o ícone giratório no canto superior direito. Nesse caso, o sistema estava fornecendo *feedback* visual, mas, como esse estava posicionado de maneira precária, eu não o havia notado.

Além do mais, em virtude do alto tráfego causado por todos se matriculando ao mesmo tempo, o sistema estava lento para responder. Isso significa que o *feedback* visual seria mais importante ainda porque as consultas estavam consumindo mais tempo que o normal. Um posicionamento mais adequado do ícone giratório seria mais próximo ao botão de pesquisas, local em que meus olhos estavam focados quando fiz minha solicitação. Com esse posicionamento mais apropriado, eu teria clicado no botão de

pesquisas e visto imediatamente minha solicitação sendo processada.

Problemas com visibilidade e *feedback* visual adequado são os mais comuns de usabilidade que vejo nos aplicativos. Sempre que ouço um usuário reclamar dizendo que uma interface é confusa ou difícil de ser compreendida, começo a examinar maneiras pelas quais eu poderia estar violando os princípios de visibilidade.

Seu aplicativo deve fornecer mensagens apropriadas de *status* continuamente. Nunca permita que seu usuário questione se o seu aplicativo continua funcionando. Se o seu aplicativo exigir algum tempo de processamento para atender a uma solicitação, informe isso ao usuário.

Hierarquia

Ao desenvolver sistemas mais complexos, pode ficar mais difícil organizar todos os recursos de seu aplicativo. O princípio da hierarquia, ou da hierarquia visual, estabelece que os aplicativos devem fornecer indicadores visuais para ajudar o usuário a perceber como o aplicativo está organizado. Com muita frequência, isso assume a forma de submenus e de outros elementos para navegação. Também pode ser aplicado por intermédio do uso do princípio de proximidade, discutido anteriormente neste capítulo.

Já trabalhei em projetos que possuíam hierarquias incrivelmente complicadas. A parte mais difícil para o desenvolvedor é tentar organizar seu aplicativo de uma maneira que faça sentido. Você pode gastar horas tentando determinar o local ao qual um recurso particular pertence ou como ele deveria se chamar.

Uma ferramenta que tem sido de valor inestimável para esses tipos de desafios é o *diagrama de afinidades*, que consiste no processo de posicionar os recursos de seu aplicativo (tipicamente usando notas adesivas, do tipo “post-it”) e organizá-los em grupos que façam sentido.

Gosto de usar notas adesivas de cores fortes, como mostrado na figura 7.6, porque tornam meu agrupamento mais visual. Também uso marcadores para desenhar pontos nas notas e indicar outros aspectos que desejo ver. As cores dos marcadores e das notas adesivas possibilitam perceber padrões rapidamente, e o adesivo das notas permite que eu tente obter diferentes organizações.

Figura 7.6 — Um exemplo de um diagrama de afinidades usando notas adesivas coloridas.

Ao organizar o portal de nossa empresa, esse tipo de diagramação foi útil para observar todos os recursos que queríamos que estivessem disponíveis aos usuários. Havia centenas de seções, políticas, aplicativos e sites. Com o diagrama de afinidades, o desafio torna-se mais plausível e ele proporciona uma maneira de tentarmos obter diferentes padrões de organização rapidamente.

Modelos mentais e metáforas

Independentemente de percebermos ou não, quando nos

deparamos com um novo aplicativo ou produto, aplicamos nossos conhecimentos provenientes de outros produtos para entender como ele poderia funcionar. Com efeito, nossas experiências anteriores moldam nossa compreensão acerca de como o mundo funciona.

Por exemplo, as funções de computador Cortar e Colar baseiam-se em nossa familiaridade com cortar papéis em pedaços e colá-los. Com efeito, a maioria dos aplicativos indica o recurso de Cortar com a imagem de uma tesoura. Esse ícone ajuda a reforçar a metáfora da função de Cortar porque a maioria de nós sabe como uma tesoura funciona. Se não tivéssemos usado o recurso de Cortar em um aplicativo antes, poderíamos ver o ícone com a tesoura e fazer uma suposição segura acerca de seu propósito.

Porém o que acontece quando nosso modelo mental nos engana?

No livro *O design do dia-a-dia* (Rocco), Donald Norman explica o desafio representado pelos modelos mentais causadores de equívocos, ao descrever eletrodomésticos:

> Aquecedores domésticos, aparelhos de ar-condicionado e até a maioria dos fornos domésticos possuem somente dois níveis de operação: capacidade máxima ou desligado. Sendo assim, eles estão sempre aquecendo ou resfriando para atingir a temperatura desejada o mais rapidamente possível. Nesses casos, ajustar o termostato a uma temperatura muito elevada não faz nada além de causar desperdício de energia quando a temperatura exceder o alvo.

Aqui está um exemplo: uma mulher se registra em um hotel e encontra seu quarto insuportavelmente quente. Ela caminha até o controle do ar-condicionado, e ele está marcando uma temperatura sufocante de 30°C! Em sua tentativa desesperada de se refrescar, ela pressiona a seta para baixo no controle até atingir a temperatura mínima de 10°C. Seu modelo mental de ar-condicionado está incorreto. Ela acredita que, ao configurar o controle para o valor mínimo de temperatura, ela fará o quarto atingir a temperatura desejada mais rapidamente. Na verdade, o ar-condicionado pode refrigerar somente a uma taxa fixa — geralmente alta ou baixa. Ao configurar o ar-condicionado com o valor mínimo de temperatura,

ela somente garantiu que ficará muito mais frio do que ela desejava.

Como desenvolvedores, devemos ter ciência dos modelos mentais que os usuários estão utilizando em nossos aplicativos. Os ícones e a linguagem devem representar, com precisão, o funcionamento de nossos aplicativos. Já vi muitos desenvolvedores escolherem ícones inadequados para os aplicativos. Sem perceber, eles deduziram um determinado propósito, e quando os usuários clicam no ícone, ficam confusos com o resultado.

Por exemplo, se você estivesse desenvolvendo um site para viagens, poderia pensar que ter um coco como o botão usado para pesquisar seria bonito e divertido. Infelizmente, os usuários não possuem um modelo conceitual de como um coco se aplica. Uma lente de aumento tem uma relação mais próxima com pesquisar algo. Isso ocorre porque muitos de nós sabemos que, no mundo real, lentes de aumento são usadas para procurar textos minuciosamente em livros e em periódicos. Embora eu o estimule a desafiar a sabedoria convencional e buscar inovações, alguns modelos devem ser deixados intactos.

Outro modelo mental interessante está na noção de Salvar nos aplicativos. Alguns de nós estamos familiarizados com o disquete tradicional como sendo o ícone para salvar arquivos em um computador. Esse modelo teve origem nos computadores mais antigos que utilizavam drives de disquetes de 3.5 polegadas para salvar documentos.

As gerações mais novas não estão familiarizadas com esse modelo porque muitas dessas pessoas nunca usaram um disquete. Certa vez, ouvi um garoto referir-se ao ícone de Salvar como uma “coisinha quadrada”. Não foi apenas divertido; foi também um lembrete eficaz a respeito da importância dos modelos mentais. Acho que, em algum momento, será necessário criar uma representação atualizada para a atividade de salvar documentos em um computador. À medida que passamos para o conceito de nuvem

para fazer armazenamentos no dia a dia, até mesmo modelos mentais como documentos e pastas se tornarão obsoletos também.

Como você acha que poderíamos melhorar essas metáforas? Realmente, isto é interessante!

Revelação progressiva

A revelação progressiva é uma ótima maneira de ajudar os usuários a entender quais recursos estão disponíveis a eles em seu aplicativo. Ao simplesmente ocultar opções que não são possíveis, você poderá reduzir a carga cognitiva dos usuários e orientá-los de maneira mais eficiente durante suas tarefas. O princípio da revelação progressiva é muito fácil de ser empregado e é especialmente útil em aplicativos mais complexos com menus carregados de recursos.

Por exemplo, o Adobe Photoshop, um software profissional para edição de fotografias, está cheio de recursos e de ferramentas para designers. Se a Adobe não incorporasse a revelação progressiva, todos os seus recursos apareceriam como disponíveis, independentemente do que eu estivesse fazendo no aplicativo. Isso representaria um fardo significativo para mim, à medida que tentasse descobrir o que é e o que não é possível fazer. Em vez disso, a Adobe deixa em cinza e desabilita os itens que não são aplicáveis à minha situação corrente, conforme pode ser visto na figura 7.7. Esse indicador sutil proporciona uma ajuda para navegar por entre as várias possibilidades do Photoshop.

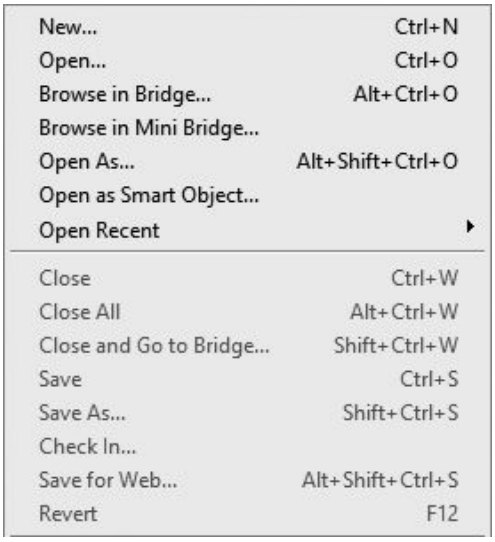


Figura 7.7 — Esse menu deixa recursos que são inacessíveis em cinza.

Consistência

O princípio da consistência pode parecer óbvio, mas vejo-o ser ignorado por muitos desenvolvedores. Esse princípio estabelece que os usuários aprendem e compreendem aplicativos com mais facilidade quando eles são consistentes com aquilo que já é conhecido. Já vi desenvolvedores introduzirem novos métodos para a execução de tarefas que já estão bem consolidadas.

Por exemplo, lembro-me de um aplicativo que exigia a criação de uma visualização prévia de meu documento antes de imprimi-lo. Em todo aplicativo que eu havia usado até então, nunca fui obrigado a criar uma visualização prévia de um documento antes de imprimi-lo. Os desenvolvedores podem ter tido boas intenções para criar esse fluxo de tarefas. Talvez achassem que seria melhor que eu fizesse uma visualização prévia de meu documento e, desse modo, seria menos provável que imprimisse algo que não quisesse. No entanto esse design não estava consistente em relação àquilo que eu já conhecia. Posso apreciar o fato de os desenvolvedores tentarem melhorar o processo de impressão, mas, nesse caso, o ato de criar uma visualização prévia antes de imprimir não era óbvio para mim e gerou uma confusão desnecessária.

Mais uma vez, sempre tento procurar maneiras novas de executar

tarefas em um aplicativo. No entanto você deve ser criterioso na introdução de novos fluxos de tarefas que possam ser inconsistentes com o entendimento geral. E se você introduzir algo novo, tenha certeza de que será melhor do que aquilo que já conhecemos!

Um exemplo disso é a maneira pela qual a FiftyThree implementou a ação Undo (Desfazer) no Paper para iPad.

O usuário posiciona dois dedos na tela e começa a movê-los no sentido anti-horário (Undo, ou Desfazer) ou horário (Redo, ou Refazer). Outros aplicativos disponibilizam um botão para Undo. No entanto a FiftyThree decidiu que um botão não se enquadrava em sua visão para o Paper. A empresa acreditava que procurar e pressionar um botão seria desnecessário e interromperia o fluxo criativo dos usuários.

Desse modo, a FiftyThree decidiu que iria melhorar o processo de Undo. Para isso, ela teve de pensar em novas maneiras de implementá-lo. A FiftyThree estudou outros mercados para obter novas ideias sobre como seria possível desfazer um trabalho. Foi assim que ela descobriu como os cineastas desfaziam seu trabalho, como explica Petschnigg:

> Nosso designer de interação Andrew Allen — que é cineasta — trabalhou bastante com dials de jog[1] em videocassetes. E, para ele, a ideia era: “Precisamos de retrocesso. Não precisamos de Undo, precisamos de Rewind (Retrocesso)!”.
>
> E foi meio daí que o gesto surgiu. E isso realmente se enquadra em nossa maneira de pensar — de certo modo — como seria a criação em dispositivos móveis. Como um app deveria funcionar quando você está em trânsito? Da forma como mantemos as pessoas em suas tarefas, em vez de fazê-las acionar um menu e encontrar um pequeno botão... parece fluir mais naturalmente.

Os desenvolvedores da FiftyThree poderiam simplesmente ter dado uma olhada no que seus concorrentes estavam fazendo e assumir que um botão tradicional de Undo seria o padrão que todos os usuários iriam entender. Poderiam ter sido conservadores e ter economizado tempo ao deixar de avaliar suas noções preconcebidas sobre como o Undo deveria funcionar.

Em vez disso, a equipe focou em seu objetivo de criar um aplicativo que estimulasse a criatividade. Da maneira como viam, o paradigma do Undo representava uma ameaça à sua missão. Portanto, eles conduziram intensas discussões sobre o que algumas pessoas teriam considerado uma função trivial. Eles procuraram entender como a maneira atual de desfazer uma tarefa não estava articulada, ao exigir a descoberta e o uso de menus e de botões. Exploraram outros mercados e tentaram descobrir como eles lidavam com a manipulação de trabalhos criativos. Ao longo de todo esse processo, eles chegaram a uma percepção inacreditável: não precisávamos de Undo; precisávamos de Rewind.

Desse modo, assim como acontece em várias situações, não há uma resposta fácil. Consistência em seus aplicativos é crucial porque reduz o fardo cognitivo de seus usuários e deixa-os mais à vontade para aprender como seu aplicativo funciona. Não há nada pior do que ter de reaprender funções básicas porque o desenvolvedor achou que seria melhor fazer as tarefas de maneira diferente.

Às vezes é bom quando um aplicativo se comporta conforme esperado, quando um item de menu está exatamente onde você espera que ele esteja ou uma ação produz o resultado adequado. Outras vezes, como no caso de desfazer tarefas no Paper, é agradável ser surpreendido e experimentar algo diferente.

Balancear a consistência em seu design pode ser desafiador, mas, se isso for feito corretamente, será possível criar um aplicativo fácil de ser aprendido e agradável de ser usado.

Disponibilidade e restrições

Muitos objetos, como ferramentas e eletrodomésticos, foram projetados para proporcionar um uso adequado e nos impedir de usá-los inadequadamente. Isso corresponde aos princípios de disponibilidade e de restrição. Um exemplo está no plugue de três pinos e na tomada. Esses objetos foram projetados não só para se

complementarem, mas também para funcionarem de uma determinada maneira. É virtualmente impossível conectar um plugue de três pinos, mostrado na figura 7.8, de maneira incorreta. Com os pinos achatados e um arredondado, o plugue deixa imediatamente claro para as pessoas como elas devem usá-lo. E se não estiver claro, ele evita que elas o conectem incorretamente e acabem se ferindo!

Figura 7.8 — Plugue de três pinos.

No hospital, temos um ditado: “Faça com que seja fácil fazer *o que é certo* e difícil fazer *o que é errado*.” A disponibilidade faz com que seja fácil fazer o que é certo, enquanto as restrições dificultam fazer o que é errado.

Se você observar usuários cometendo erros em seu aplicativo, considere limitar as opções ou antecipar seus fluxos de trabalho. Desenvolva ações que funcionem de modo que seja impossível fazer o que é errado. Os usuários apreciarão o fato de você estar tomando conta deles e terão mais confiança em seu aplicativo.

Confirmação

Uma maneira de evitar que os usuários cometam erros é pedir

confirmação. O princípio da confirmação estabelece que um aplicativo deve evitar ações indesejadas ao solicitar uma verificação, conforme mostrado na figura 7.9.

Figura 7.9 — Esse carinha já me salvou em mais de uma ocasião.

Na maioria dos aplicativos, se estou trabalhando com um documento e tento fechá-lo sem salvar, uma pergunta será mostrada. Geralmente, ela indaga se quero salvar antes de sair do programa. Se eu selecionar Cancel (Cancelar) e tentar fechar o aplicativo novamente, a pergunta será apresentada de novo. Essencialmente, não há nenhuma maneira de fechar o aplicativo sem antes lidar com a questão de querer ou não salvar o documento. Isso me protege de cometer erros e perder meu trabalho.

Certifique-se de que seu aplicativo antecipará uma ação indesejada. Nada fará seus usuários o odiarem mais do que permitir que eles percam seus trabalhos de maneira não intencional.

A Lei de Hick

A Lei de Hick consiste em um modelo prescritivo que ajudará você a calcular o tempo necessário para os usuários tomarem uma decisão, como resultado da quantidade de opções que possuem. Também é conhecida como tempo de reação, ou TR, e é matematicamente representada da seguinte maneira:

$$RT = a + b\text{Log}_2 N$$

O modelo pode se provar útil na avaliação de seus menus para garantir que não estejam sobrecarregados. Uma pergunta comum

no design de um aplicativo é: “Quantos itens devem estar presentes em um menu, e como devem ser organizados?”.

Por exemplo, o esquema de navegação em um portal de uma empresa pode ser extremamente difícil de administrar. Considere o portal de empresa que discutimos anteriormente no princípio de hierarquia. É mais do que provável que os usuários queiram que aquilo que estejam procurando seja o primeiro item no menu. Afinal de contas, é o item mais importante, porque eles estão procurando por ele! Veja a figura 7.10 para analisar um exemplo.

Obviamente, pode haver somente um primeiro item, portanto, poder identificar e dar prioridade aos itens em um menu pode ser um verdadeiro cabo de guerra.

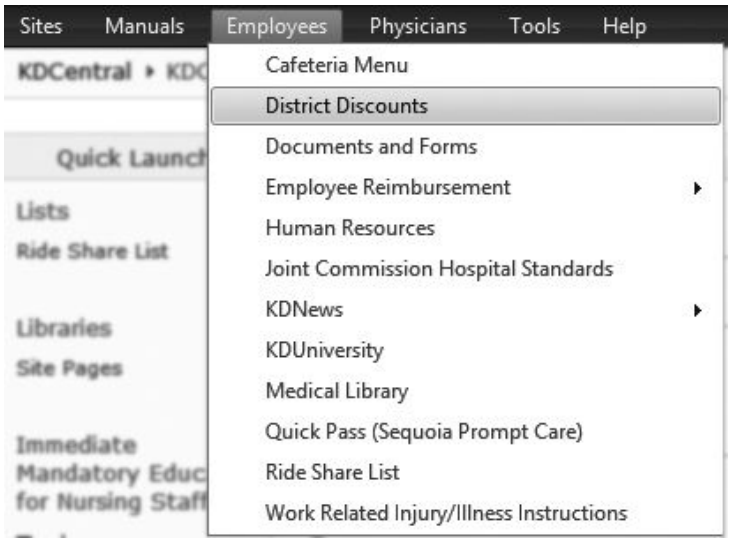


Figura 7.10 — Um exemplo de menu suspenso no portal de nossa empresa.

Desse modo, em virtude de sua natureza linear, a Lei de Hick sugere que processamos as informações a uma taxa constante. Resumindo, quanto mais itens você colocar diante dos usuários, mais tempo será necessário para que eles encontrem o que estão procurando.

Parece óbvio, mas ainda vejo desenvolvedores criarem aplicativos ou sites com menus de navegação incrivelmente complexos. Acho que é fácil perder o controle sobre nossos aplicativos. Ficamos

acrescentando mais e mais itens e, antes que tenhamos percebido, não é mais possível administrá-los. Então movemos e reorganizamos os itens para evitar a difícil tarefa de decidir do que devemos nos livrar.

Ao aplicar o princípio da hierarquia e a Lei de Hick como modelo prescritivo, podemos tomar melhores decisões sobre o valor de cada item em nossos menus.

Lei de Fitt

A Lei de Fitt pode ajudar você a determinar o tamanho dos elementos-alvo, tais como botões, menus etc., em sua interface, com base na distância que o dispositivo apontador do usuário deve percorrer. Esse modelo prescritivo é expresso em termos de tempo de movimento, ou TM, e prova que, quanto maior a distância que o usuário tiver de percorrer entre dois elementos, menor será a precisão com que o usuário alcançará o alvo. Se sua intenção for fazer o usuário clicar em um botão, o tamanho desse botão será determinado pela distância entre o botão e o cursor do usuário. A equação é apresentada a seguir:

$$MT = a + b \log_2(2A/W)$$

Suponha que o Google tivesse feito os botões Google Search (Pesquisa Google) e I'm Feeling Lucky (Estou com sorte) menores e mais para a lateral, conforme mostrado na figura 7.11. Isso aumentaria a distância entre a caixa de pesquisa, em que o cursor se encontra, e os botões, ou alvos. Desse modo, nossa precisão seria menor e nosso tempo de movimento seria maior.

Figura 7.11 — Os usuários não ficariam satisfeitos com essa alteração no design!

A distância que os usuários devem percorrer a partir de um objeto deve determinar o tamanho do objeto para o qual estão se dirigindo. A Lei de Fitt é correlativa. Em outras palavras, quanto maior a distância que um usuário deve percorrer, maiores deverão ser os objetos-alvo. O tamanho exato será determinado pelo tempo de movimento aceitável.

A Lei de Fitt é útil no caso de interfaces orientadas a mouse; porém, há alguns estudos novos que ajustaram o modelo para acomodar interfaces sensíveis ao toque também.

A versão resumida

- É importante aprender e estudar os princípios de design. Há vários princípios e, ao simplesmente seguir suas diretrizes, você poderá melhorar incrivelmente a experiência em seus aplicativos.
- O princípio de proximidade estabelece que os seres humanos percebem um relacionamento entre objetos que estejam mais próximos. Utilize esse princípio agrupando funções. Isso fará com que seja mais fácil aprender a usar e entender seu aplicativo.
- O princípio da visibilidade estabelece que indicadores visuais devem estar presentes para ajudar os usuários a compreender o *status* de seu aplicativo.
- O princípio da proeminência visual estabelece que a atenção de um usuário será atraída para objetos que sejam maiores, que tenham cores mais fortes ou que sejam mais proeminentes.
- O princípio do *feedback* visual estabelece que um aplicativo deve responder ao usuário ao indicar que uma entrada de dados foi recebida. Os aplicativos também devem indicar ao usuário quando estiverem processando uma solicitação.
- Modelos mentais e metáforas são maneiras pelas quais os seres humanos transferem o conhecimento do mundo real

para o mundo da computação. Cerifique-se de que sua iconografia, a linguagem e outras metáforas estejam calcadas em conhecimentos que os usuários compreendam.

- A revelação progressiva remove e desabilita recursos que não são aplicáveis ao estado corrente dos usuários.
- O princípio da consistência estabelece que as tarefas em um aplicativo devem funcionar conforme esperado. Você não deve inventar novos fluxos de trabalho para executar tarefas que já são compreendidas pelo usuário.
- A disponibilidade ajuda os usuários a fazer o que é certo, e as restrições evitam que façam o que é errado.
- O princípio da confirmação estabelece que os aplicativos devem exigir verificação para evitar que os usuários executem ações indesejadas. O diálogo de confirmação para salvar um documento é um bom exemplo.
- A Lei de Hick consiste em um modelo prescritivo usado para determinar o tempo necessário para selecionar um item, com base na quantidade de itens disponíveis para seleção. Quanto mais itens você incluir em um menu, maior será o tempo de resposta do usuário.
- A Lei de Fitt consiste em outro modelo prescritivo, usado para determinar o tempo necessário para um usuário mover o cursor de um local para o local-alvo. Quanto maior for a distância que um usuário percorrer com seu cursor, menor será a precisão com que o usuário alcançará o objeto-alvo.

---

1 N.T.: Um dial de jog é uma espécie de controle com formato circular, usado em geral para controle de áudio e de vídeo, comum em equipamentos profissionais.

CAPÍTULO 8
Reunindo feedback

“Milhões viram a maçã cair, mas foi Newton quem perguntou por quê.”

— Bernard Mannes Baruch

É hora de sair pelo mundo e descobrir o que os usuários realmente pensam dos aplicativos que criamos para eles. Devemos fazer perguntas detalhadas e estar abertos a críticas que podem ser difíceis de escutar. Devemos observar os usuários e documentar nossas descobertas para adquirir uma compreensão geral do que funciona e do que não funciona.

Receber críticas não é divertido. Se alguém disser a você que gosta de receber *feedback* em seu trabalho, essa pessoa estará mentindo. Ser informado de que você deixou de fazer algo, cometeu um erro ou seguiu na direção errada essencialmente significa que você não terminou. Significa que tem mais trabalho a fazer. Significa que você não é perfeito.

Não vou dizer a você que passe a gostar de críticas. Não vou dizer que você deva sair gritando de alegria diante da ideia de refazer o design de seu aplicativo ou que deva sair dançando alegremente quando perceber que terá de recodificar uma função complexa.

A verdade é que todos querem marcar um gol de placa. Queremos mostrar nosso novo aplicativo aos usuários, clientes, amigos e familiares e ouvir que acertamos em cheio: entendemos tudo corretamente, terminamos e fizemos tudo na versão 1.0.

Aqui está minha sugestão: devemos aprender a ser tolerantes com

os *feedbacks*. Os melhores desenvolvedores que conheço desejam fazer o que é certo, não importam as consequências. Acima de tudo (sono, dinheiro ou ego), eles querem desenvolver o melhor aplicativo possível. Percebem que receber *feedback* honesto, de qualidade, é a única maneira de chegar lá, e aqui está o segredo — na verdade, eles pedem para ter *feedback*. Não disse que eles gostam disso, mas eles o valorizam. Esses desenvolvedores colocam seus aplicativos nas mãos de potenciais usuários o mais rápido possível. Fazem perguntas detalhadas aos usuários e os estimulam a ser brutalmente honestos. Eles pesquisam o mercado, observam seus concorrentes e revisam seus aplicativos para garantir que proporcionarão vantagem competitiva.

Gasto todo o meu tempo e ganho minha vida escrevendo código; eu realmente amo fazer isso. Em minha opinião, é a melhor parte de criar qualquer aplicativo. O ato de criar um produto funcional a partir do nada, para dizer o mínimo, é viciante. Se você não tiver o desejo de codificar, será impossível ser um desenvolvedor de sucesso. De que outra maneira você poderia reunir energia para passar uma semana tentando descobrir exatamente qual é o acionador de um evento?

Quando estou trabalhando no código, sinto-me confortável e, por mais que possa soar estranho, ele proporciona as menores dificuldades. Posso ficar golpeando o teclado e permanecer superfocado para fazer os recursos funcionarem. O problema com essa abordagem é que nunca paro para refletir: “Os usuários, antes de tudo, *querem* esse recurso?”. Sei que devo passar um tempo com meus usuários previamente, e sei que deveria estar fazendo perguntas e criando um plano de ação. Em vez disso, acabo dizendo algo do tipo “Só quero ver se [um desafio de programação] é possível, em primeiro lugar.” Digo a mim mesmo, inúmeras vezes, que retornarei para um processo mais estruturado de desenvolvimento de software assim que eu tiver [um desafio de programação] funcionando. Quando faço isso, estou fadado ao

fracasso.

Com frequência, lembro a mim mesmo que chegarei à fase de codificação no tempo devido, mas, até lá, devo obter respostas para algumas perguntas. Devo adquirir uma compreensão básica sobre o que estou tentando criar. É claro que posso ter uma ideia geral, mas tenho de garantir que entendi as especificidades. É impossível descobrir de maneira eficiente como será meu aplicativo ao mesmo tempo em que estou escrevendo seu código. Se eu quiser ser bem-sucedido e criar um aplicativo que os usuários irão querer usar, deverei ser disciplinado e paciente.

Se você estiver trabalhando em um aplicativo no momento, eu o desafio a responder as seguintes perguntas básicas:

- Que problema seu aplicativo resolve? Qual é seu foco principal?
- Quem são os usuários ideais para seu aplicativo? Como são eles? São usuários experientes ou são novatos?
- Quais são os três recursos que devem ter presença obrigatória? Como você sabe que eles são os mais importantes?
- Como seu aplicativo agregará valor? Como ele trará melhorias em relação ao que as pessoas estão usando atualmente?

Se você não puder responder facilmente a algumas dessas perguntas, não se desespere. A maioria dos desenvolvedores que conheço tem dificuldades com elas. A verdade é que, se estivermos focados somente em fazer os recursos funcionarem, será impossível manter uma perspectiva sobre a visão geral e o propósito de nosso aplicativo. É por isso que sugiro que os desenvolvedores abracem o design centrado no usuário, motivo pelo qual escrevi este livro! Somente se permitirmos que os usuários ofereçam *feedback* continuamente, poderemos testar nossos pressupostos e garantir que estamos seguindo na direção correta.

No entanto isso vai muito além de simplesmente pedir aos usuários que compartilhem suas opiniões. Devemos ser metódicos,

quase antropológicos, em nossos estudos. Devemos pedir informações a nossos usuários e observar seus comportamentos. Dessa maneira, podemos aprender com o que eles nos dizem e podemos ler nas entrelinhas.

Vamos dar uma olhada em algumas ferramentas que podemos usar para coletar *feedback* e observar nossos usuários de maneira eficiente.

Quantos usuários serão necessários?

Em primeiro lugar, você poderá estar se perguntando quantos participantes você deverá ter para obter um *feedback* que seja significativo. Assim como ocorre com qualquer grande pergunta, a resposta é que isso depende. Depende realmente do que você está tentando realizar e do valor que pretende atribuir aos resultados.

Em alguns casos, eu começaria com algumas perguntas bastante informais. Posso fazer algumas perguntas somente para um grupo pequeno de pessoas a respeito de suas necessidades no domínio de um determinado problema. Sei que o *feedback* dessas pessoas pode não ser representativo da base mais ampla de usuários, portanto, contextualizo isso quando estiver analisando suas respostas.

Eu argumentaria que a variedade pesa muito mais do que a quantidade. Em outras palavras, obtenha informações de um grupo diversificado de pessoas. Os desenvolvedores, com frequência, buscam *feedback* junto a seus colegas. Não há nada de errado nisso, mas é necessário perceber que nossos amigos desenvolvedores têm a tendência de avaliar nosso trabalho a partir de um ponto de vista técnico. Obviamente, se o público-alvo de seu aplicativo for os desenvolvedores, então fará sentido coletar *feedback* de desenvolvedores que tenham todos os níveis de experiência.

Se eu estiver desenvolvendo um aplicativo para os enfermeiros

supervisores organizarem os funcionários em cada turno, vou querer reunir *feedback* de uma variedade de pessoas que estejam envolvidas no processo de organização dos enfermeiros. Poderei pesquisar junto aos enfermeiros supervisores, os enfermeiros, os auxiliares de enfermagem, os diretores dos enfermeiros e os coordenadores das unidades. Pode ser um grupo pequeno de cinco pessoas ou um grupo grande de trinta. Só depende da quantidade de recursos que tenho disponível. Não faria sentido testar o aplicativo com meus colegas desenvolvedores. Eles podem oferecer algum *feedback* útil, porém o conhecimento de mundo de um profissional de enfermagem não estaria presente.

A maneira pela qual você recrutar seus participantes também poderá afetar a legitimidade de sua amostra. Vamos supor que você esteja desenvolvendo um site para idosos, para ajudá-los a conhecer alguns benefícios para sua saúde. Você vai querer saber o quanto os idosos se sentem confortáveis em usar a Internet, de modo que você lhes enviará uma pesquisa por email. Ao enviar uma pesquisa por email, você poderá visar ao grupo errado. É claro que eles podem estar na faixa etária desejada, mas o fato de eles estarem usando email pode colocá-los em um grupo em particular, ou seja, o das pessoas que se sentem confortáveis com o uso de computadores. Para garantir que estamos obtendo *feedback* de um grupo diversificado de idosos, devemos considerar o envio da pesquisa por email e por escrito. Dessa maneira, nossas decisões de design serão baseadas em idosos que se sentem confortáveis, ou não tão confortáveis assim, com o uso de computadores.

Muitos estudos foram feitos para descobrir o número mágico de pessoas que tornam uma amostra estatisticamente viável. Embora haja algumas discussões sobre essa questão no mercado, o famoso *expert* em usabilidade Jakob Nielsen acredita que você pode atingir os melhores resultados com um mínimo de cinco usuários. Ele afirma que a maioria dos erros de usabilidade será descoberta pelos cinco primeiros usuários e que será possível aprender muito pouco

a partir desse ponto, como mostrado na figura 8.1. Desse modo, estudar mais do que cinco usuários não agrega valores adicionais. Pelo contrário, torna o estudo mais complexo e difícil de ser administrado.

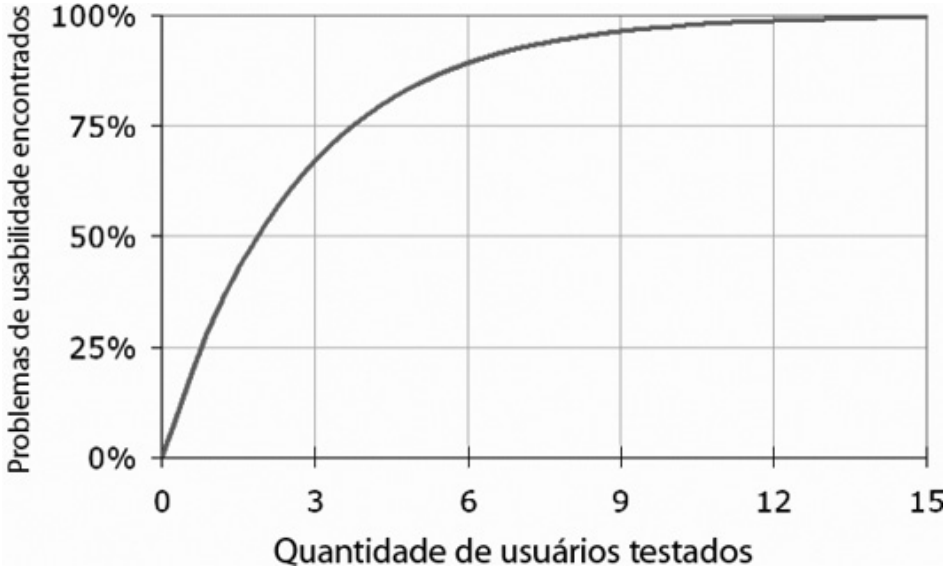


Figura 8.1— Problemas de usabilidade encontrados por número de usuários estudados (Fonte: Nielsen Norman Group).

Pense em quando você pede uma pizza. A primeira fatia é sempre a melhor porque proporciona o mais alto nível de satisfação. Se estiver com fome, você poderá estar disposto a pagar 5 dólares por ela. A sexta fatia (se você conseguir comer tanta pizza quanto eu) simplesmente não será tão satisfatória, e é provável que você estará menos disposto a pagar 5 dólares por ela. Sendo assim, podemos concluir que a sexta fatia de pizza simplesmente não é tão valiosa quanto a primeira.

Nielsen sugere que acrescentar usuários em seu estudo produz o mesmo efeito. Adicionar mais usuários simplesmente aumentará a complexidade do estudo e, na realidade, não agregará tanto valor quanto os cinco primeiros.

Ciente disso, eu acho que, se puder reunir de cinco a dez participantes para observar enquanto eles usam seu aplicativo, você terá ideias significativas sobre o que poderia ser melhorado.

Observe o gráfico da figura 8.1. Se você não estudar nenhum participante, poderá esperar descobrir uma quantidade de problemas de usabilidade *igual a zero*. Esse número parece ser o mais garantido.

Pesquisas

Uma maneira de obter *feedbacks* diretamente dos usuários é por intermédio da criação e coleta de resultados de pesquisas. As pesquisas podem ser uma ferramenta eficiente para atingir um amplo espectro de pessoas em um breve intervalo de tempo. Há vários sites e diversas ferramentas online para ajudá-lo a preparar e a distribuir pesquisas (SurveyMonkey [*http://www.surveymonkey.com*] é uma das ferramentas que usei em diversas ocasiões).

Criar uma pesquisa válida e confiável é uma forma de arte. É verdade! É preciso não só saber que perguntas deverão ser feitas, mas também fazê-las de modo a obter *feedbacks* úteis. Realizar perguntas claras, concisas e imparciais, bem como selecionar o grupo correto de pessoas, é a chave para uma pesquisa bem-sucedida.

Minha preferência pessoal pelo estilo das perguntas que farão parte da pesquisa é usar a escala de Likert. A escala recebeu esse nome por causa de seu criador, doutor Rensis Likert, e apresenta um conjunto de respostas para cada pergunta. Aqui está um exemplo:

Esse livro me ajudou a melhorar minhas habilidades no desenvolvimento de aplicativos.

a. Concordo plenamente
b. Concordo
c. Nem concordo nem discordo
d. Discordo
e. Discordo plenamente

Ao fornecer um conjunto de respostas, as escalas de Likert

ajudam você a ter maior percepção sobre o quanto um entrevistado concorda com uma determinada afirmação. Isso pode ser valioso quando estiver lidando com pessoas que ficam “em cima do muro”: usuários que dizem que tudo está bom quando solicitamos suas opiniões.

Esses tipos de resposta ajudam os entrevistados a estruturar suas respostas e os orienta no sentido de qualificar suas opiniões. Com as escalas de Likert, você poderá abranger uma variedade de tipos de resposta:

Frequência

Sempre; frequentemente; às vezes; nunca.

Importância

Muito importante; importante; um pouco importante; nem um pouco importante.

Valor

Muito alto; alto; médio; baixo; muito baixo.

Satisfação

Totalmente satisfeito; satisfeito; neutro; insatisfeito; totalmente insatisfeito.

Classificação

5, 4, 3, 2 e 1.

No entanto uma das desvantagens das escalas de Likert está no potencial para *distorções de concordância*, que consiste na tendência dos entrevistados em concordar com as afirmações porque parece ser menos confrontador ou exige menos esforço. Com muita frequência, você conduzirá uma pesquisa somente para descobrir que tem um conjunto de usuários que está totalmente de acordo com o que você está afirmando. Essas informações não são muito úteis, se estiver tentando decidir em quais recursos você deverá investir seu tempo.

Se estiver preocupado com distorções de concordância, você sempre poderá empregar o *método das pontuações mais altas* em seus cálculos. Com esse método, somente os valores que sejam os melhores ou os mais desejados deverão ser considerados.

Por exemplo, se estiver perguntando aos usuários sobre a importância de ter um corretor ortográfico em seu aplicativo, você somente contabilizará aqueles que marcarem “Muito Importante” e dividirá esse número pelo total de entrevistados. Isso pode parecer radical, mas ajudará você a entender o quanto um determinado recurso é desejado.

Se optar pelo uso da escala de Likert, dê o máximo de si para fazer perguntas usando uma linguagem clara e neutra. Quanto mais difícil de processar for uma pergunta — ou quanto mais radical for a afirmação —, mais provável será que os entrevistados respondam de uma maneira que não reflita suas intenções.

Por exemplo, uma pergunta como esta é tendenciosa demais para o lado positivo:

O recurso de mensagens do IT Works é útil.

a. Concordo plenamente
b. Concordo
c. Nem concordo nem discordo
d. Discordo
e. Discordo plenamente

À primeira vista, parece que estamos tentando entender a utilidade do recurso de mensagens do sistema IT Works. Essa pergunta é compreensível, mas a formatação poderia levar a respostas que sejam menos viáveis. Por exemplo, e se os usuários acreditassem que o recurso de mensagens é valioso porque, durante o treinamento, foram informados de que era? Podemos nos equivocar acerca da percepção de valor dos usuários tomando a quantidade de vezes que eles usam o recurso de mensagens como indicador.

Uma maneira de corrigirmos isso é fazer a pergunta usando a escala de frequência:

Eu uso o recurso de mensagens do IT Works:

a. Todo dia
b. Algumas vezes por semana
c. Algumas vezes por mês
d. Raramente
e. Nunca

Ao formular a pergunta dessa maneira, estamos indo direto ao ponto ao qual queremos chegar. Se quisermos saber o quanto o recurso de mensagens é útil, devemos perguntar qual é sua frequência de uso.

Para melhorar a legibilidade em minhas pesquisas, gosto de selecionar uma ou duas escalas de Likert e ater-me a elas. Por exemplo, se optar por usar uma escala de frequência duas vezes, eu devo garantir que o conjunto de respostas seja o mesmo. Também prefiro limitar o conjunto de respostas a não mais do que cinco, e certifico-me de que elas sejam fáceis de ser compreendidas. Não quero que os usuários tenham a intenção de concordar com uma afirmação e discordem acidentalmente. Ao criar afirmações breves e oferecer consistência nas opções para respostas, permito que os entrevistados avaliem rapidamente e respondam às minhas pesquisas.

Além do mais, ao combinar suas afirmações para que reflitam uma linguagem tanto positiva quanto negativa, você fará com que os entrevistados parem e reflitam sobre suas respostas. Dê o melhor de si para alcançar um equilíbrio. Uma pesquisa consistente é fácil de ser lida e compreendida, mas, com uma pitada de inconsistência, ela deixará os entrevistados atentos e eles ficarão envolvidos com o questionário.

Você poderá coletar respostas abertas, como, por exemplo, por meio de respostas dissertativas, mas acho que muitos participantes não querem gastar tempo para respondê-las. Além disso, erros de ortografia, sentenças fragmentadas e afirmações obscuras acabam por agregar muito pouco valor. Como tudo mais, o uso de respostas

dissertativas dependerá, provavelmente, do público-alvo que você estiver pesquisando.

Conduzindo entrevistas

Embora as pesquisas possam atingir um público-alvo numeroso com poucos esforços, elas proporcionam somente um entendimento geral das necessidades de seus usuários. Não importa o quanto você seja cuidadoso com seu processo de pesquisa, ainda continuará parecendo que você está fazendo suposições com base nos resultados. Embora as entrevistas exijam mais tempo e alcancem um público-alvo menos numeroso, com frequência, elas fornecem informações mais detalhadas. A condução de entrevistas garante que você irá capturar a intenção das observações dos usuários. Você poderá esclarecer perguntas que não foram corretamente compreendidas e prestar atenção em sinais não verbais, como linguagem corporal e tom.

Em geral, há três tipos de entrevistas: estruturada, não estruturada e contextual. Elas também são conhecidas, respectivamente, como entrevistas formais, entrevistas informais e investigações contextuais. Todas as três abordagens possuem seus prós e seus contras, mas compartilham o valor de ser uma interação direta. As diferenças entre as entrevistas estruturadas e não estruturadas são mínimas; porém as investigações contextuais são um pouco diferentes. Não é necessário que deva ser um ou outro. Você pode optar por usar cada método em diferentes etapas do desenvolvimento de seu aplicativo.

Quando começar a fazer pesquisas para seu aplicativo, você poderá optar por conduzir entrevistas não estruturadas para explorar o domínio do problema. Entrevistas não estruturadas permitem um diálogo mais aberto, adequado a esse tipo de exploração. A chave para as entrevistas não estruturadas está no nível de informalidade. Não estruturar sua entrevista permite que ideias livres venham à

tona porque você estará tendo uma discussão aberta sobre o domínio do problema.

Isso não quer dizer que sua discussão deverá ser aleatória. Ela simplesmente será menos focada em formalidades e em consistência. Entrevistas não estruturadas são úteis quando você realmente não sabe quais perguntas deve fazer. Talvez você tenha uma ideia para um aplicativo, mas não tenha tomado nenhuma decisão ainda sobre qualquer recurso específico. O objetivo das entrevistas não estruturadas é conversar com o número máximo de pessoas que puder sobre o problema que você está tentando solucionar.

As entrevistas não estruturadas permitem uma conversação normal. Ao perguntar aos usuários sobre o domínio do problema e permitir que eles falem livremente, você ficará surpreso com os conhecimentos que obterá. É bem possível que eles mencionem um problema do qual você não tinha conhecimento ou uma solução alternativa que eles usam e que acabará sendo um recurso diferenciado em seu aplicativo.

Em oposição à entrevista não estruturada está a entrevista estruturada. Enquanto as entrevistas não estruturadas estimulam a liberdade e a exploração, as entrevistas estruturadas valorizam a consistência. Geralmente, as entrevistas estruturadas são conduzidas utilizando-se um roteiro. As mesmas perguntas são feitas a todos os usuários, no mesmo tom de voz e na mesma ordem, e suas respostas são cuidadosamente documentadas.

Esse nível de formalidade é melhor quando você precisa de respostas para perguntas específicas. Por exemplo, suponha que você queira saber como os usuários se sentem em relação a um novo menu para navegação em seu aplicativo. Em uma entrevista estruturada, você poderá fazer perguntas específicas sobre o menu e tomar decisões de design com base no *feedback* recebido diretamente dos usuários. Normalmente, as entrevistas estruturadas

têm uma tendência a ser mais rápidas porque você já sabe quais perguntas gostaria que fossem respondidas.

Se você planeja conduzir entrevistas estruturadas, eu o aconselho a assumir um comprometimento em relação a elas. Prepare um roteiro com antecedência e faça o máximo para lê-lo, de forma consistente, para cada usuário. Foque somente no roteiro e evite distrações ou conversas paralelas. Isso permitirá comparar igualmente as respostas de cada usuário.

De maneira muito semelhante às entrevistas estruturadas e não estruturadas, as investigações contextuais exigem que você faça uma série de perguntas. Contudo o processo é muito mais íntimo porque você mergulhará no ambiente do usuário. Em vez de conduzir entrevistas em uma sala de reuniões ou por telefone, o foco de uma investigação contextual é garantir que você envolverá os usuários no ambiente em que eles usarão seu aplicativo. Esse tipo de estudo permite entrar em contato com fatores ambientais que poderão afetar a capacidade de seu usuário de usar seu aplicativo. Esses fatores, tais como iluminação, barulho e ergonomia, podem ter efeito amplo, embora significativo, sobre a experiência geral dos usuários. Alguns usuários aceitam esses fatores como naturais, e jamais pensariam em mencioná-los em uma entrevista por telefone ou em seu escritório. Às vezes, vale a pena ir até onde está seu aplicativo ou onde ele será usado e ver com seus próprios olhos.

Como exemplo, vamos voltar ao nosso aplicativo do diretório de visitantes, discutido no capítulo 3. O diretório de visitantes era um sistema sensível a toques, e, para fornecer uma resposta adequada, passei semanas ajustando o áudio perfeito que indicaria que um item havia sido pressionado. Tentei vários sons e trabalhei no sentido de encontrar o tom perfeito para o áudio. Um era agudo demais e outro, grave demais. Tentei até mesmo criar meu próprio som usando um sintetizador. Levei uma eternidade para encontrar

um som com o qual estivesse satisfeito.

Quando finalmente encontrei um som adequado e finalizei o projeto, instalei o quiosque eletrônico na recepção do hospital. Foi aí que percebi que havia cometido um erro fundamental.

A cerca de cinco metros do quiosque eletrônico, havia uma fonte. Com o som de água correndo tão perto, não era possível ouvir a resposta audível de meu aplicativo!

Também percebi que, como não era possível ouvir os sons, meu *feedback* visual não era significativo. Mal dava para ver o botão mudando de cor quando era pressionado. Desse modo, tive de deixar a cor mais viva para garantir que os usuários pudessem confirmar a entrada de dados, independentemente de terem ouvido ou não o som. A fonte, embora fosse maravilhosa de se ver, havia afetado dramaticamente a capacidade dos visitantes de usar meu aplicativo.

Também aumentei o volume do som para que pudesse ser ouvido, mesmo com a fonte. Foi um erro. Descobri, posteriormente, que os voluntários que trabalhavam na recepção frequentemente desligavam os alto-falantes. Quando perguntei a eles por que estavam fazendo isso, disseram que o volume era alto demais e era desagradável quando estavam conversando com visitantes.

A questão dessa história é que, se eu tivesse reservado um tempo para explorar a recepção do hospital por meio da condução de uma investigação contextual, teria tido uma noção desses fatores ambientais. Teria percebido que o barulho da fonte exerceria um impacto sobre a capacidade dos usuários de receber a resposta adequada de meu design. Teria percebido também que havia um limite para o volume do som usado pelo meu aplicativo, antes que começasse a incomodar os funcionários.

Além do mais, se eu tivesse trabalhado com um grupo de foco composto de visitantes em uma sala de reuniões, não esperaria que ninguém mencionasse um problema com a fonte. Em vez disso,

teríamos gastado tempo discutindo os recursos que eles gostariam de ter em um diretório. Ao não passar um tempo no ambiente em que o diretório de visitantes iria ser usado, deixei de perceber fatores importantes.

Como dizem por aí, o campo de visão é de 20/20[1]. Quando posicionei o quiosque eletrônico na recepção, ficou evidente para todos que não havia resposta visual ou audível suficiente. No entanto, quando testei o aplicativo em meu escritório silencioso, não havia nenhuma maneira de eu saber que as respostas visual e audível seriam um problema. Poderia ter optado por me recriminar por causa desse problema, mas é impossível prever todas as variáveis que afetarão a experiência de usuário em um aplicativo.

É por isso que é importante se envolver em atividades como investigações contextuais e entrevistas com usuários. A partir de cada aspecto do modelo de design centrado no usuário, eu amplio minha janela de compreensão.

Isso exige que saiamos de nossas mesas, façamos perguntas, caminhemos pelo ambiente e exploremos, nós mesmos, o domínio do problema.

Análise de tarefas

Existem mais dois tipos de análise que você pode usar para avaliar a eficiência de seu aplicativo: análise de tarefas e avaliação heurística.

A análise de tarefas consiste no estudo de cada passo de uma determinada tarefa. A questão, nesse tipo de análise, é compreender totalmente todos os passos exigidos para completar uma tarefa e melhorar o processo com seu aplicativo. No capítulo 4, falamos da utilidade dos diagramas de fluxo de trabalho. A análise de tarefas é uma ótima maneira de gerar esses tipos de modelo.

Diferentemente da investigação contextual, não é necessário observar os usuários enquanto estão envolvidos com suas tarefas,

embora, definitivamente, eu recomende que isso seja feito. Com a análise de tarefas, é possível adquirir uma compreensão acerca de uma tarefa simplesmente fazendo perguntas ou lendo um manual de procedimentos.

O desafio de percorrer uma análise de tarefas com os usuários ocorre, em geral, porque eles querem descrever todos os aspectos da tarefa, tudo de uma só vez. Se não for cuidadoso, sua análise estará repleta de passos confusos, fora de sequência. Os usuários nem sempre têm a habilidade de explicar as tarefas que executam. Com mais frequência do que se espera, eles fornecem detalhes insignificantes ou deixam de lado passos cruciais. Não é culpa deles. Eles não sabem que os desenvolvedores pensam em termos de loops e de instruções condicionais. Eles não entendem que estamos pensando na tarefa e tentando aplicar 0s e 1s a ela.

Por isso, gosto de usar questionários baseados em cenários para obter a história da tarefa. No hospital, meu questionamento, em geral, seria algo do tipo:

> Tudo bem, vamos recapitular. Suponha que meu braço tenha sido decepado e que acabei de chegar à emergência. O que acontece a seguir?

É uma imagem terrível — e também traz um pouco de humor —, mas ajuda os usuários a percorrer uma tarefa e dar um passo de cada vez. Também procuro interrompê-los, caso eles se adiantem demais, e peço que expliquem os acrônimos ou os termos técnicos.

Além do mais, é crucial que você repita o que compreendeu aos usuários. Com frequência, será algo assim:

> Então, deixe-me ter certeza de que entendi o que você está dizendo. Você disse que seu primeiro passo consiste em ir até o scanner; em seguida, você chama o aplicativo de scanning no computador, e depois você...

É incrível como é fácil interpretar incorretamente o que os usuários estão nos explicando. Somente ao repetir minhas percepções, posso ter certeza de que entendi corretamente suas explicações.

Avaliação heurística

Em vez de focar em uma tarefa em particular, a avaliação heurística consiste no processo de examinar os aplicativos em relação a um conjunto de regras ou diretrizes. Usada de forma pioneira por Jakob Nielsen, as avaliações heurísticas podem ocorrer com o usuário presente ou não. É quase como o processo de editar um trabalho escrito, mas, em vez de revisar a gramática e analisar as regras de ortografia, você estará avaliando a eficiência de seu aplicativo com base em padrões ou princípios do mercado.

Há vários recursos disponíveis para orientá-lo. Por exemplo, a Microsoft, o Google e a Apple possuem diretrizes razoavelmente estritas para o design de seus aplicativos em suas plataformas. Essas diretrizes podem ajudá-lo a entender o que seu aplicativo deve ou não deve fazer. Já vi diretrizes de design que são muito específicas até o último pixel. Elas fornecem orientações sobre a posição dos elementos da interface de usuário (UI), o uso de animações, os tempos de carga, a terminologia e várias outras instruções.

Por exemplo, se os usuários sabem que o botão Share (Compartilhar) nos aplicativos iOS são usados para compartilhar conteúdo via email, redes sociais e outros serviços, não seria sensato criar seu próprio mecanismo de compartilhamento que os usuários teriam de aprender. Em alguns casos, as empresas irão rejeitar a submissão de seu aplicativo, caso essas diretrizes não sejam seguidas. As empresas de software reforçam esses padrões porque querem garantir que seus usuários tenham uma experiência consistente entre os aplicativos. Sendo assim, não compreender as diretrizes da plataforma de desenvolvimento que você estiver utilizando pode sair caro.

Quanto mais familiarizado você estiver com as regras de design do framework, mais fácil será identificar inconsistências.

Criação de storyboards

Outra maneira de obter *feedbacks* iniciais no processo de design é por intermédio da criação de *storyboards* – a prática de esboçar uma experiência, ponto por ponto. A criação de *storyboards* é usada com frequência em Hollywood para planejar os passos de uma cena de filme.

Assim como a criação de esboços, a criação de *storyboards* pode ser intimidadora, mas não posso deixar de reforçar: não é necessário ser um artista para se envolver com a criação de *storyboards*. Se conseguir desenhar formas básicas ou traçar linhas, é tudo de que você precisará. Veja a figura 8.2 para um exemplo. A questão do *storyboarding* não é criar experiências artísticas maravilhosas. É começar a formular a progressão de seu aplicativo de modo visual. Se você souber desenhar somente imagens que parecem palitinhos, então desenhe todo o processo com imagens que parecem palitinhos!

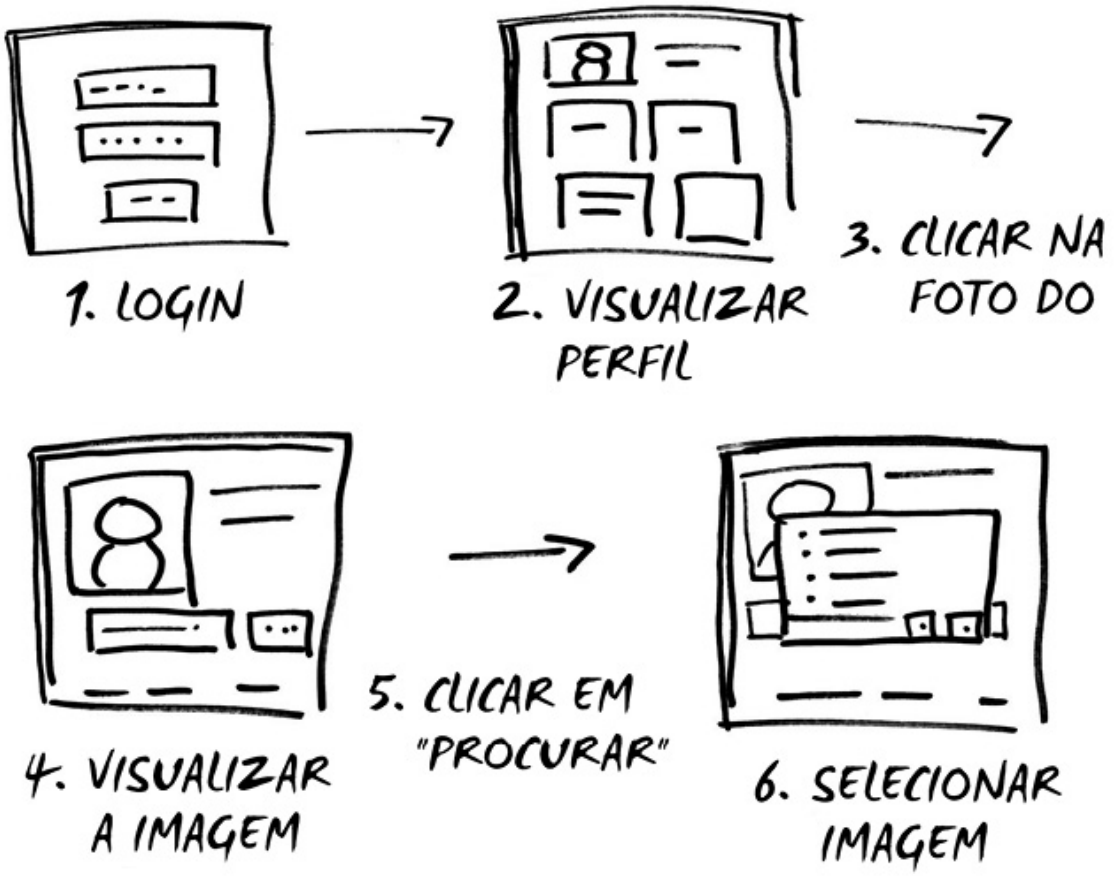


Figura 8.2 — Storyboard sobre a mudança de foto no perfil do usuário.

Invista tempo pensando: “Em primeiro lugar, os usuários farão isso, depois, farão aquilo, depois...”. Diferentemente de um diagrama de fluxo de dados ou de fluxo de trabalho, a criação de *storyboards* é muito mais visual. Seus *storyboards* devem incluir esboços iniciais de elementos da UI. Pense também no layout e no design de seu aplicativo e na forma como ele responderá aos usuários à medida que eles navegarem pelo aplicativo: Com o que isso se parecerá? Quais telas eles poderão ver? Haverá animações para indicar a progressão de uma tela para outra?

Já ouvi algumas pessoas se referirem a esse processo como pseudocodificação, e, de certo modo, isso faz sentido. É uma maneira de começarmos a codificar sem escrever realmente em uma linguagem de codificação. Em vez disso, estamos usando uma linguagem visual. Ela nos prepara para todas as possibilidades e funcionalidades que deverão ser codificadas. Com a criação de *storyboards*, teremos uma visão geral do aplicativo e um mapa muito mais claro para indicar o caminho a ser seguido.

A criação de *storyboards* pode levar você a tomar decisões prévias sobre o layout e o processo de seu aplicativo. Você começará a perceber quais conceitos serão ou não serão necessários para proporcionar uma boa experiência de usuário. Mostre seus *storyboards* a seus usuários e certifique-se de que o fluxo de trabalho atenda às suas necessidades. Essa é uma ótima maneira de trazer à tona qualquer erro de comunicação e evitar erros custosos de design.

Usar um software para fazer seus *storyboards* pode ser tentador. Vou argumentar contra isso. Para a criação de *storyboards*, papel e caneta serão seus melhores companheiros. Com aplicativos como Adobe Photoshop e QuarkXPress, você correrá o risco de focar na qualidade do *storyboard*, e não na qualidade da experiência de usuário.

Os *storyboards* podem ajudar você a transformar seu desenvolvimento em uma história, assim como aprendemos com o processo de design da FiftyThree para o Paper, baseado em narrativa. A criação de *storyboards* ajuda você a explorar cada passo de seu aplicativo e avaliar seu design.

Tente usar um dos fluxos de trabalho de seu aplicativo e criar um *storyboard* em torno dele. Quantos passos serão necessários? Quais telas serão apresentadas? Como o usuário reagirá a elas? Há alguma maneira de eliminar ou de combinar telas para facilitar o processo?

Usando protótipos

A prototipagem consiste no processo de criar mockups de alta ou de baixa qualidade para o design de seu aplicativo para ter algo tangível a ser testado junto aos usuários. A prototipagem é uma maneira eficiente de ajudar seus usuários a visualizar o que você pretende disponibilizar por meio de seu aplicativo.

Há um benefício inacreditável resultante de ver suas ideias iniciais tomarem forma. Embora possam exigir tempo em um primeiro momento, os protótipos podem resultar na economia de horas para desenvolver algo que, em última instância, não funcionaria. Os protótipos são uma maneira de fazer a avaliação do design visual de seu aplicativo sem fazer um investimento significativo em programação.

Se cometer o erro de abordar seu design a partir do código, você estará mais inclinado a encontrar soluções que serão melhores para seu código, mas não para a experiência de usuário. A prototipagem liberta você de pensar nos desafios da codificação e o deixa mais focado na interação do usuário com seu aplicativo.

Contudo existe um perigo na prototipagem para nós, desenvolvedores famintos por código. A prototipagem pode imitar bem de perto o desenvolvimento real do aplicativo, e pode,

facilmente, ficar fora de controle.

O desenvolvedor e designer Billy Hollis tem uma ótima maneira de saber se o seu protótipo atingiu esse ponto:

> Se chegar a fazer testes automatizados em seu protótipo, então você estará fazendo errado.

Não estou sugerindo que ter um protótipo de alta fidelidade como objetivo não tenha seus méritos. Em alguns casos, os desenvolvedores de software criam um protótipo semifuncional que eventualmente se transforma no produto final.

O problema é que os desenvolvedores cometem o erro de focar no protótipo. Eles o usam como uma maneira de fingir que estão testando o design; mas, na verdade, estão simplesmente programando. Além do mais, programadores podem optar por desenvolver seus protótipos em sua ferramenta nativa de desenvolvimento. Acho, realmente, que se você estiver desenvolvendo um protótipo usando suas ferramentas de desenvolvimento (por exemplo, Microsoft Visual Studio, XCode ou algum outro ambiente de desenvolvimento integrado popular), a tentação de mergulhar no código será maior ainda.

Há outras soluções (veja o capítulo 11) mais adequadas para o desenvolvimento de protótipos funcionais. Esses softwares oferecem as ferramentas adequadas para garantir seu foco nos elementos corretos de seu protótipo. Eles facilitam ajustar rapidamente seu layout e limitam sua capacidade de mergulhar em uma programação elaborada. Isso manterá seu foco longe do código, permitindo focar no design visual e interativo do aplicativo.

Quando considerar a qualidade de seu protótipo, pense na percepção que o usuário terá sobre o estágio de desenvolvimento em que o aplicativo se encontra. Maior fidelidade e riqueza do protótipo poderão ter implicações no tipo de *feedback* que você receberá dos usuários.

Por exemplo, se você colocar um protótipo bem elaborado diante

de um usuário e pedir que ele o avalie, não fique surpreso se ele lhe fornecer poucas ideias substanciais. Isso ocorre porque, quanto mais funcionalidades e design seu protótipo oferecer, mais provável será ele acreditar que o produto esteja perto de ser finalizado. Não faz sentido fornecer *feedback* a essa altura — parece que está quase tudo pronto!

Ele poderá deixar de questionar algumas das asserções fundamentais feitas por você. Ele poderá se equivocar, achando que elas já foram decididas. Desse modo, ele só dará *feedbacks* acerca de detalhes menores.

No entanto, se você colocar um protótipo de baixa fidelidade – como aquele mostrado na figura 8.3 – diante dele, a impressão será de que houve poucos comprometimentos em relação ao design em geral. Ele se sentirá mais encorajado a questionar as funcionalidades e os conceitos principais.

De maneira semelhante, você estará mais disposto a aceitar *feedback* se tiver investido menos tempo desenvolvendo seu protótipo. Se um usuário disser que você se esqueceu de uma tela crucial em seu protótipo, e você só tiver feito um desenho em papel, não há problemas! Basta pegar uma nova folha de papel e, em minutos, você terá organizado seu protótipo para explorar essa necessidade. Você gastará menos tempo defendendo seu trabalho e mais tempo trabalhando construtivamente com o usuário — e é exatamente isso que você vai querer fazer.

Figura 8.3 — Teste de usabilidade utilizando um protótipo em papel (Fonte: Workshops públicos do Luma Institute).

Quando os usuários não sabem do que necessitam, é comum você se ver com o que chamo de “a síndrome do artista por encomenda”. Isso acontece quando você se encontra em um ciclo no qual estará constantemente apresentando opções.

Você mostra a eles o fluxo de trabalho do aplicativo. Eles não gostam. Eles não conseguem dizer por que não gostam, portanto dizem “Tente usar menos telas”.

Você volta ao seu escritório, trabalha durante alguns dias e retorna com o aplicativo redesenhado com um novo fluxo de trabalho. Eles também não gostam desse. Eles sugerem: “Algumas telas a mais poderiam fazer com que melhorasse”.

Todo desenvolvedor teme esse processo, e eu poderia argumentar que esse é o motivo principal pelo qual evitamos o envolvimento dos usuários. Ninguém quer criar algo por encomenda, portanto, permitimos que os usuários se envolvam, mas limitamos severamente o escopo em que eles podem nos ajudar.

Criamos um protótipo quase finalizado e perguntamos sobre recursos pouco significativos aos usuários. Ao limitar suas opções, efetivamente limitamos seu envolvimento. É um design fantasma centrado no usuário. É mais um discurso vazio do que a realidade. Hollis explica como cometemos esse erro:

Você pediu ajuda aos usuários — eles o ajudarão. Mas você não lhes deu nenhum

> espaço para que pudessem ajudá-lo. Não ofereceu nenhuma opção. Não lhes ofereceu nenhum espaço para que extraíssem um julgamento significativo de valor sobre o que você está fazendo. Você lhes ofereceu um túnel bem estreito e disse: “Agora, me ajudem nesse espaço pequeno e confinado!”.

Nossa tarefa é apresentar opções e mostrar as possibilidades aos usuários. Fazemos isso para ajudá-los a explorar e comunicar o que estão procurando. A conclusão é que os usuários não conseguem fazer isso de modo eficiente sem observar e visualizar as várias opções ou possibilidades. Temos de nos desafiar para colocar diferentes conceitos e ideias diante de nossos usuários, para que possamos atingir o centro de suas necessidades.

Uma estratégia que emprego é colocar um mockup desafiador diante de meus usuários. Talvez seja algo radicalmente diferente daquilo com que eles se sentiriam confortáveis. Não espero que eles amem o design. Na verdade, espero que eles o detestem! Acho que os usuários tendem a ser mais articulados quando explicam o que não gostam.

Ao ouvi-los listar o que não gostam, posso chegar até o fundo, no coração das necessidades de meus usuários. É uma ótima estratégia quando você estiver diante de pessoas que ficam “em cima do muro”, e que não conseguem decidir o que querem. Tente mostrar-lhes um protótipo que não estejam esperando e, se eles não gostarem, pergunte por quê. Você poderá se surpreender com a quantidade de decisões de design que podem ser tomadas dessa maneira.

Se você quiser evitar que seus usuários se atenham a minúcias relativas a detalhes insignificantes de seu aplicativo, forneça-lhes detalhes mais importantes para criticar. Faça com que eles se envolvam mais cedo e permita-lhes oferecer sugestões sobre itens importantes. Isso não só evitará que você siga na direção errada, mas seus usuários também se sentirão mais comprometidos com o produto final.

Testes A/B

Mesmo com dados viáveis de *feedbacks* diretos do usuário, ainda poderá ser difícil tomar uma decisão definitiva de design. Talvez você esteja lutando com o velho dilema “Devo optar por isso? Ou por aquilo?”. Em situações em que ambas as decisões de design possuem mérito ou respostas positivas dos usuários, você poderá considerar a condução de um *teste A/B*.

O teste A/B consiste em uma prática para testar usuários em relação a dois designs diferentes e deixar que os dados decidam.

Suponha que você esteja tentando decidir em que local deverá apresentar o botão “Compre agora nas suas páginas de produtos”. Algumas pessoas de sua equipe acham que ele deveria estar imediatamente abaixo da imagem do produto, enquanto outros acham que deveria ser posicionado abaixo da descrição do produto. Ambas as ideias parecem ser boas.

Nesse caso, seu teste A/B disponibilizaria duas páginas diferentes de produtos para dois grupos diferentes de usuários. Seria um estudo às escuras porque os usuários não estariam cientes de que, na verdade, há duas experiências diferentes quando estiverem comprando um produto.

Em um teste A/B, o design vencedor poderá ser a configuração que fizer com que os usuários cliquem com mais frequência no botão “Comprar agora”. Essencialmente, relegamos a tomada de decisão para que seja baseada em dados. Fizemos os usuários decidirem, com seus cliques de mouse, qual método é o mais adequado.

Garanto que se você passou qualquer quantidade de tempo na Internet, já terá participado desse tipo de testes A/B. O serviço de pesquisas da Microsoft, o Bing, por exemplo, usa testes A/B com bastante frequência. É minha ferramenta padrão para pesquisas e, diversas vezes, percebi pequenas alterações aqui e ali. Certa vez, notei que o Bing havia alterado o botão de pesquisa, de uma lente

de aumento para um botão em que se lia “Search” (Pesquisar), semelhante ao da ilustração presente na figura 8.4. Mais tarde, naquele dia, o botão havia voltado a ser uma lente de aumento. Admito que a mudança foi sutil, mas, agora que conheço as práticas de testes A/B, costumo perceber essas alterações. Seja observador e você poderá notar seus sites prediletos fazendo isso também!

Figura 8.4 — Sites importantes, como o Bing, empregam testes A/B para observar os efeitos de mudanças sutis em usabilidade.

Embora os testes A/B possam ser um excelente método para tomar decisões de design, eles devem ser usados com moderação. Pode ser fácil adotar a prática de criar testes A/B para todas as facetas de seu aplicativo. Ao basear-se unicamente em dados para tomar decisões, você focará menos no entendimento do comportamento de seus usuários.

Jeff Atwood, autor do blog Coding Horror, compara os desafios dos testes A/B com o filme *Feitiço do Tempo*. O filme, caso você não o conheça, é estrelado por Bill Murray como um meteorologista chamado Phil que, inexplicavelmente, acaba vivendo o mesmo dia repetido todos os dias. Phil usa essa circunstância estranha para ter vários encontros com Rita, a mulher por quem havia se apaixonado. De forma muito semelhante aos testes A/B, em cada encontro, Phil captura pequenos detalhes sobre o que Rita gosta ou não e escolhe diferentes opções em cada encontro subsequente. Com o tempo, Phil usa todos os dados que havia coletado para criar um encontro que parecia ser perfeito. Quando vemos que Rita, apesar disso, não se apaixona por Phil, percebemos que sua ideia simplesmente não funciona.

Atwood explica:

> O Phil não estava escolhendo essas opções porque acreditava sinceramente nelas. Estava escolhendo essas opções porque queria obter um resultado específico — conquistar Rita —, e os dados experimentais haviam lhe informado o caminho que ele deveria tomar. Embora o encontro fosse tecnicamente perfeito, não pareceu ser sincero para Rita, e isso fez toda a diferença.

Se você planeja experimentar um teste A/B, tome cuidado para não focar em detalhes irrelevantes. Não há problemas em testar minúcias, mas considere o quadro geral também.

Observe também que um teste A/B adequado pode exigir uma amostra grande para a reunião de dados significativos.

Além do mais, o sucesso dos testes A/B baseia-se em sua habilidade de selecionar os indicadores corretos. Por exemplo, se você estiver julgando o sucesso do design de seu site pela cor do ícone do carrinho de compras, poderá estar ignorando critérios mais significativos, como a capacidade dos compradores de encontrar os itens para comprar.

A versão resumida

- Não existe um número mágico para a quantidade de participantes que você deva ter em um estudo. Embora algumas pessoas discordem, Jakob Nielsen, um líder e *expert* em usabilidade, sugere não mais do que cinco.
- As pesquisas podem ajudá-lo a coletar *feedback* rapidamente de vários usuários. As pesquisas são relativamente fáceis de serem distribuídas, mas geralmente fornecem resultados genéricos.
- A escala de Likert consiste em um formato de perguntas e respostas usadas em pesquisas. As respostas ajudam os participantes a informar o valor de aspectos como frequência, importância e satisfação.
- Um desafio da escala de Likert está na distorção de concordância, que corresponde à tendência dos entrevistados de concordarem ou responderem às perguntas de uma

maneira que eles acreditam que os pesquisadores iriam preferir.

- Conduzir entrevistas exige mais tempo, mas é uma ótima maneira de obter *feedbacks* diretamente dos usuários. Há três tipos de entrevista: não estruturadas (informais), estruturadas (formais) e contextuais (investigação contextual).
- As entrevistas não estruturadas incentivam a conversação. Elas tendem a ser abertas, sem uma agenda específica em mente. Esses tipos de entrevista devem ser usados quando você estiver começando a explorar o domínio do problema.

- As entrevistas estruturadas focam em consistência. Recomenda-se que você utilize um roteiro e faça perguntas aos entrevistados usando o mesmo tom, a mesma ordem e a mesma linguagem.
- As investigações contextuais são muito semelhantes às entrevistas, exceto pelo fato de focarem na visita e na conversa com os usuários em seus ambientes. Ao explorar o ambiente, você poderá trazer à tona fatores que afetarão seu aplicativo, tais como ruídos, iluminação, distrações etc.
- A análise de tarefas consiste no exame completo de uma tarefa ou de um processo. O ideal é que você realmente observe o usuário executar a tarefa; no entanto, não é uma exigência.
- A avaliação heurística consiste no processo de analisar seu aplicativo em relação aos padrões e princípios do mercado. Muitas empresas de software disponibilizam diretrizes de usabilidade e, com frequência, irão rejeitar aplicativos em seus mercados, caso esses padrões não sejam atendidos.
- Os *storyboards* são uma maneira de esboçar a experiência de usuário, passo a passo. Os *storyboards* deverão passar a impressão geral de como seu aplicativo irá fluir.
- Os protótipos podem ser uma ferramenta eficiente para permitir que os usuários visualizem o que você pretende desenvolver para eles. Dê opções a seus usuários e permita

que eles tomem decisões sobre aspectos importantes de seu aplicativo.

- O teste A/B consiste no processo de apresentar duas interfaces aos usuários e escolher o design vencedor com base na interface que proporcionar a ação desejada. As decisões são tomadas totalmente com base nos dados.

---

1 N.T.: Uma pessoa com visão 20/20 corresponde a alguém com uma visão considerada normal pelos oftalmologistas. Essa pessoa é capaz de enxergar normalmente a uma distância de 20 pés (6 metros).

CAPÍTULO 9
Estudos de usabilidade

“As mentes das pessoas mudam por meio de observação, e não por causa de argumentos.”

— Will Rogers

Ao longo deste livro, argumentei a favor do valor de envolver os usuários em seu design. No capítulo anterior, discutimos sobre diferentes maneiras pelas quais você pode coletar feedback e adquirir uma ampla compreensão do domínio do problema. Mas e se você já desenvolveu um aplicativo? Como saber se ele está atendendo efetivamente às necessidades dos usuários?

É claro que você poderia simplesmente perguntar a eles. Falamos bastante sobre discutir com os usuários e fazer perguntas a eles; esse processo continua sendo importante. Os usuários podem dizer muito sobre o que está funcionando e o que não está. No entanto a maneira mais eficiente de perceber as necessidades dos usuários é observando-os diretamente.

Há ocasiões em que as perspectivas dos usuários são tendenciosas e subjetivas. Experimente perguntar aos usuários quanto tempo seu aplicativo gasta para carregar e você poderá obter respostas variadas. Para alguns usuários, o tempo de carga de seu aplicativo será simplesmente adequado; para outros, o aplicativo será terrivelmente lento e inviável de utilizar. Os estudos de usabilidade podem ajudá-lo a determinar quanto tempo seu aplicativo gasta para carregar e, mais importante ainda, qual o impacto que isso representa para seus usuários. Os testes de

usabilidade quantificam suas observações.

Com efeito, os estudos de usabilidade são um dos recursos-chave do design centrado no usuário. Se você estiver iniciando um redesign de um site, por exemplo, como saber se seu novo design é melhor do que o original? Como saber se você realmente atingiu sua visão e melhorou a experiência de seus usuários?

Os estudos de usabilidade podem ajudar a fornecer essas respostas. Eles determinam linhas de base para acompanhar as melhorias no design de seu aplicativo. Ao observar ativamente os usuários e documentar seus comentários, suas ações, seus erros e sucessos, pode-se obter uma perspectiva valiosa sobre como, exatamente, seu aplicativo está sendo utilizado.

O que são estudos de usabilidade?

Um estudo de usabilidade, ou teste de usuário, consiste na observação mensurada do comportamento dos usuários, à medida que eles se envolvem no uso de seu aplicativo de software. Ele é científico na prática e favorece as métricas, as medições e os dados para provar as pressuposições. Há duas maneiras de abordar os estudos de usabilidade.

Você poderá optar por executar o estudo no ambiente do usuário. Por exemplo, se estiver desenvolvendo um aplicativo para gerenciamento de chamadas telefônicas, você poderá optar por observar os usuários em um call center de verdade. Esse estudo é chamado de estudo em ambiente natural ou estudo de campo. De maneira muito semelhante às investigações contextuais que discutimos no capítulo anterior, os estudos de usabilidade são ainda mais focados, mais sistemáticos e mais consistentes.

O objetivo geral de um estudo de usabilidade é medir a eficiência de um recurso ou de um conjunto de recursos presentes em seu aplicativo. Para isso você deve definir métricas, como o tempo para execução ou a quantidade de erros ou uma combinação dessas

medidas. O estudo também pode ser combinado com uma pesquisa para medir aspectos que são difíceis de ser observados, como satisfação ou percepção de valor.

Para executar um estudo de usabilidade com sucesso, você deverá ter um plano.

Criando um plano de testes

Pode parecer óbvio, mas, antes de poder conduzir testes, será necessário saber o que você está procurando. Se você estiver desenvolvendo um aplicativo para celulares, não entregue simplesmente o celular para as pessoas para ver se elas vão gostar. Melhores resultados poderão ser obtidos com uma abordagem organizada e que contenha medições. Provavelmente você vai querer ter os participantes corretos, um roteiro preparado e um conjunto de diretrizes. Isso garante que os dados que você coletar sejam consistentes e viáveis.

Assim como ocorre na criação de pesquisas, é importante ter o conjunto adequado de usuários. Se estiver desenvolvendo um aplicativo de linha de negócios para um grupo específico de usuários, obviamente você vai querer incluí-los em sua pesquisa. Se estiver desenvolvendo um aplicativo para celulares, para usuários experientes ou não, então certifique-se de que estará incluindo pessoas de todos os níveis de experiência e familiaridade. Conforme discutimos em relação aos questionários, certifique-se de evitar uma *seleção tendenciosa*, selecionando usuários que não sejam representativos do grupo.

Um roteiro pode ser uma maneira produtiva de prover consistência em seus testes. É fácil desviar-se do curso ou distrair-se quando estiver conduzindo um estudo. Preparar um roteiro permitirá executar o teste exatamente da mesma maneira com todos os participantes.

Pode parecer estranho ler um roteiro, mas realmente é a melhor

maneira de garantir a consistência. Acho que um roteiro define o tom para um bom estudo qualitativo. Os usuários saberão que você está agindo seriamente para obter resultados válidos e que você veio preparado para o estudo. Se necessitar de um exemplo de como um roteiro deve ser, não se esqueça de conferir o apêndice A no final deste livro.

Aqui estão alguns itens que deverão ser incluídos em seu roteiro:

Introdução

Não se esqueça de apresentar o conceito do estudo e seu propósito. Dependendo da situação, você poderá ter de pedir permissão legal ao participante. Certifique-se de saber com antecedência como você planeja usar os resultados, especialmente se tiver planos para publicar suas descobertas.

Garantias

Alguns participantes tendem a ficar um pouco nervosos. Tranquilize-os explicando que você está testando o aplicativo, e não a eles. Se estiver testando usuários que fazem parte de sua organização, é crucial que você explique que os resultados serão anônimos.

Nenhum funcionário vai querer que seu chefe saiba que ele está tendo dificuldades para usar o software da empresa. Acho que o uso de palavras como *testar* ou *classificação* pode deixar as pessoas nervosas; em vez disso, considere o uso de palavras como *observar* e *estudo*.

Diretrizes para testes

As diretrizes ajudam a definir como o estudo será conduzido. Os participantes serão treinados previamente? Eles poderão fazer perguntas? Quando tempo eles terão? Ter essas diretrizes ou regras estabelecidas ajudará os participantes a saber o que se espera deles e permitirá observar os usuários de maneira consistente.

Tarefas

As tarefas serão a base de seu estudo de usabilidade. Elas representam as métricas que você planeja usar. Por exemplo, você pode decidir medir o tempo que um usuário gasta para alterar a foto de seu perfil ou para pesquisar um item. Pode contabilizar também a quantidade de consultas necessária para localizar um determinado par de sapatos em um site de compras. Independentemente de quais sejam as tarefas ou a tarefa, certifique-se de que elas possam ser quantificadas.

Conclusão

Não se esqueça de reservar um tempo no final do estudo para endereçar qualquer pergunta ou preocupação. Descobri que pode ser divertido discutir minhas observações com o participante. Não importa quantas vezes você tenha dito a eles que não era um teste, eles ainda sentem aquela vontade de saber como se saíram.

Além disso, ao compartilhar suas observações, você poderá receber comentários adicionais ou explicações do participante. Conversas após um estudo podem esclarecer o que você acabou de observar.

Agradecimentos

Por fim, não se esqueça de agradecer aos seus participantes. É fácil ficar tão envolvido em observações e discussões a ponto de se esquecer dessa simples cortesia. Além do mais, pode ser uma boa ideia lembrá-los de como você planeja usar os resultados e responder a qualquer pergunta de última hora.

Se for apropriado, considere oferecer um presentinho pelo tempo dispensado por eles. Acho que um mimo da lanchonete faz maravilhas!

De que você precisará

Você deve estar se perguntando quantas pessoas devem participar para que o estudo seja eficiente. Uma opção é aplicar a regra de

“não mais do que cinco” de Jakob Nielsen, descrita no capítulo anterior. Lembre-se de que a única garantia é que, se você não conduzir nenhum estudo, poderá esperar que *nenhum* problema de usabilidade seja encontrado. Prefiro conduzir um estudo com três pessoas a não fazer nenhum estudo. Por outro lado, ter participantes *demais* pode ser um pesadelo logístico que traria complicações e atrasos aos seus esforços.

Não pense que testar somente algumas pessoas seria um desperdício de tempo. Se você tiver acesso somente a um grupo pequeno de usuários, mesmo assim, eu o aconselho a tentar realizar testes de usabilidade. Já conduzi estudos bem pequenos e, apesar disso, obtive conhecimentos valiosos.

Além da quantidade de participantes, aqui estão alguns itens que você deve ter quando estiver se preparando para um estudo:

Cronômetro

Se estiver planejando medir quanto tempo um participante gasta para completar uma tarefa para determinar o sucesso de seu design, use um cronômetro ou um dos vários aplicativos para smartphone com recursos de cronômetro. Obviamente, o tempo para completar uma tarefa não deve determinar todas as decisões de design. Só porque um usuário consegue completar rapidamente uma tarefa não significa que o design seja mais agradável. Mesmo que não seja um fator específico em minha decisão de design, gosto de registrar o tempo que cada participante gasta para completar as tarefas.

Bloco de anotações

Use um bloco de anotações e uma caneta, mesmo que possa ser tentador tomar notas em um laptop ou em um dispositivo móvel. Os usuários agem e falam rapidamente, e você vai querer estar preparado para capturar qualquer comportamento ou comentário quando surgirem. Acho que um símbolo ou um diagrama desenhado

rapidamente pode ser uma maneira mais eficiente de capturar minhas observações. Papel e caneta permitem que eu tenha essa flexibilidade.

Ambiente

Considere o ambiente que melhor complemente o que você está procurando estudar. Se for importante que haja um mínimo de distrações, você deve garantir que haja um local isolado ou que seja tranquilo. Certifique-se de ter preparado o local para seu estudo. Itens básicos como cadeiras confortáveis, iluminação, conectividade em rede e temperatura da sala afetam seu estudo. Não se esqueça de verificar o ambiente e as ferramentas que seus participantes estarão usando para garantir que eles não interferirão ou que não representarão uma distração. Os participantes apreciarão o fato de você ter preparado o espaço para seus estudos. Se você planeja gastar uma quantidade significativa de tempo com eles, certifique-se de ter água e lanches disponíveis.

Planilhas ou banco de dados

Insira seus dados em uma planilha ou em um banco de dados. Se o prazo permitir, você poderá desenvolver seu próprio miniaplicativo para coletá-los. Organizar eletronicamente os dados o ajudará a identificar padrões rapidamente. O Microsoft Access e o Excel são excelentes produtos para esse tipo de coleta de dados e relatórios.

Devo lembrar novamente que os usuários são rápidos. Haverá pouco tempo para inserir dados enquanto você estiver observando os comportamentos. Recomendo registrar suas descobertas em seu bloco de anotações antes e, então, inseri-las posteriormente em uma planilha ou em um banco de dados.

Câmeras e gravadores de áudio

Utilize câmeras de vídeo ou equipamentos para gravação de áudio para capturar todos os comentários e o comportamento dos

entrevistados. Geralmente, não sou fã desses equipamentos porque eles tendem a distrair o participante. A maioria das pessoas detesta a ideia de ser filmada e, não importa o quanto você lhes assegure do contrário, os participantes terão a sensação de estar sendo testados. Ter uma câmera apontada para seus rostos não ajuda a reduzir essa ansiedade.

Se achar que uma câmera ou um dispositivo para gravação seja necessário, certifique-se de que o equipamento seja discreto. Além do mais, inclua o fato de que você estará gravando o estudo em seu roteiro. Você também deverá considerar a opção de perguntar se o participante prefere não ser filmado.

Usar a câmera de seu smartphone para tirar algumas fotos poderá ser menos intrusivo. Às vezes, um conjunto de fotos é suficiente para documentar como os participantes responderam durante o estudo.

Sei que para alguns de vocês esse nível de preparação pode parecer assustador, e a ideia de escrever um roteiro pode parecer um pouco exagerada. Você poderá sentir-se tentado a simplesmente se sentar junto a seus usuários e observá-los usarem seu aplicativo. Não posso deixar de enfatizar o quanto é importante ter um plano para realizar seu estudo de usabilidade. De maneira muito semelhante a todo o processo de design centrado no usuário, ter uma visão e um plano documentados para seus estudos de usabilidade são a melhor maneira de garantir que você obterá resultados significativos.

Conduzindo o estudo

Após definir um plano, será o momento de conduzir seu estudo. Recomendo imprimir cada tarefa que você planeja estudar em uma folha separada. Se você colocar uma pilha de instruções diante dos participantes, haverá uma boa chance de que eles comecem a folhear as páginas e se adiantem demais. Imprimir as tarefas individualmente minimizará essas distrações e evitará que seus

participantes pulem etapas.

Além disso, tenha uma cópia impressa de seu roteiro para fazer a leitura. Prepare o roteiro para que coincida com cada tarefa. O ideal é que você tenha um roteiro geral para o estudo e minirroteiros para cada tarefa. Certifique-se de que a linguagem seja consistente de uma tarefa para outra, mesmo que você pareça estar sendo repetitivo.

Você deve recomendar continuamente aos participantes que eles pensem alto. Por meio desse processo a pessoa estará dizendo a você o que ela está pensando enquanto executa a tarefa. É muito fácil para os usuários focar silenciosamente na execução da tarefa. O problema com isso é que você estará perdendo seu processo de raciocínio. Você não poderá realmente saber o que eles estão pensando enquanto eles estiverem usando seu aplicativo a menos que eles digam a você.

Seu roteiro deve incluir uma instrução para que eles falem alto diversas vezes. Forneça essa instrução antes do início de cada tarefa. A maioria dos participantes apresentará resistência porque poderá parecer tolice — eles acharão que estão divagando —, mas incentive-os a fazer isso, de qualquer maneira. Seus fluxos de consciência estarão repletos de informações, mesmo que eles não percebam.

Suponha que uma usuária esteja procurando o botão Pesquisar em seu aplicativo. Se ela não disser “Muito bem, estou procurando o botão de pesquisar nesse momento”, não haverá nenhuma maneira de você associar o que ela está tentando fazer. Se os participantes pensarem em voz alta, você poderá documentar melhor suas ações e focar naquilo que poderá estar lhes atrapalhando. Suponha que você esteja testando um aplicativo para emails e calendário. Um ótimo estudo de usabilidade deverá ter participantes dizendo algo como:

> Muito bem, preciso enviar uma mensagem. Vou clicar neste botão porque aqui diz Novo.

Ah, espere. Parece que um novo compromisso foi criado. Não é isso que eu quero. Como fecho isso? Ah, aqui está.

Tudo bem, entendi agora. Devo selecionar primeiro a Caixa de entrada. Isso é estranho; gostaria que os ícones fossem diferentes para que eu pudesse diferenciá-los.

Nesse caso, teríamos documentado que o usuário estava tendo problemas para identificar a diferença entre criar uma nova mensagem e um novo compromisso. Isso nos lembrará de revisar a maneira como estamos lidando com a criação de novos itens em nosso aplicativo. Talvez nos levará a considerar uma nova decisão de design. Se o usuário não tivesse mencionado que estava tentando criar uma mensagem nova, poderíamos não ter percebido que ele estava tendo dificuldades.

Se um usuário tiver perguntas sobre como executar uma tarefa, sugiro enfaticamente que você esclareça somente o que é exigido pela tarefa. Limite suas instruções e faça o melhor que puder para eliminar qualquer orientação enquanto o usuário estiver tentando executar a tarefa. Você provavelmente vai querer que o participante tente descobrir sozinho e pense alto à medida que o fizer.

Com muita frequência, os estudos de usabilidade acabam se transformando em sessões de treinamento. Não há nada de errado em mostrar ao usuário como executar as tarefas, porém certifique-se de que esse treinamento não ocorrerá *durante* o estudo. Se o seu aplicativo exigir treinamento (por exemplo, um aplicativo complexo de linha de negócios, que os usuários nunca viram antes), então considere conduzir o treinamento antes do estudo. Se for possível, deixe um tempo suficiente entre o treinamento e o estudo para que os usuários possam se lembrar de como o aplicativo funciona.

Se os participante pedirem ajuda, em vez de dizer-lhes o que fazer, considere devolver a pergunta.

Participante: “Então, preciso descobrir uma maneira de imprimir isso. Essa é a maneira correta de imprimir?”.

Eu: “Como você acha que seria possível fazê-lo?”.

> Participante: “Bem, outros programas possuem o recurso Imprimir no menu Arquivo, mas esse aplicativo não tem esse recurso. ”

Ao responder ao participante com uma pergunta, recebi uma indicação de que eu deveria considerar colocar a função Imprimir no local em que outros aplicativos a colocam (no menu Arquivo). Evite a tentação de prover treinamento para aumentar suas chances de detectar itens que você possa ter deixado de lado.

Se possível, posicione-se atrás dos participantes. Acho que é muito mais fácil fazer anotações e observações sem distraí-los.

Quando cada tarefa for executada, reserve um momento para capturar suas medições (tempo decorrido, quantidade de erros etc.). Tente fazer com que os participantes respondam a um pequeno questionário enquanto você faz essas anotações de última hora. Descobri que isso representa uma transição suave, que não faz os participantes ficarem esperando, enquanto você escreve freneticamente em seu bloco de anotações. Além do mais, enquanto eles estiverem respondendo ao questionário, você poderá se preparar para a próxima tarefa (zerar o cronômetro, pegar as próximas instruções impressas, e assim por diante).

O questionário poderia ser baseado na satisfação, com perguntas sobre a experiência dos usuários enquanto executavam a tarefa. Veja o capítulo 7 para obter mais informações sobre a condução de pesquisas.

As ações acontecem rapidamente durante um teste de usabilidade. Você ficará surpreso com a quantidade de observações e comentários que vai querer anotar para rever depois. Ter um roteiro claro, materiais impressos e qualquer outra ferramenta pronta contribuirá para que seu estudo transcorra de maneira eficiente.

Não hesite em praticar

Se estiver ansioso para ir até lá e começar, ótimo! Contudo eu incentivo você a lutar contra a tentação de estudar prematuramente

seus usuários. Uma ótima maneira de testar o roteiro e a estrutura de seu estudo é praticar. Considere pedir a um amigo ou a um colega que analise todo o estudo junto a você. Isso dará uma oportunidade para identificar qualquer erro de digitação ou linguagem confusa em seu roteiro.

O ideal é que você encontre alguém que seja um bom candidato para seu aplicativo. Se praticar com um colega de projeto, ele poderá estar familiarizado demais com o aplicativo para apontar áreas que estejam confusas.

Independentemente de quem você encontrar, passe um tempo praticando para estar preparado para conduzir o estudo oficial.

Organizando seus resultados

Seus estudos de usabilidade devem proporcionar a você um conjunto de conhecimentos e de feedbacks. Pode ser tentador tomar um comentário e começar a fazer modificações no design, mas você terá melhores resultados se reservar um tempo para organizar seus resultados. Ao calcular suas medidas e organizar seus comentários, você converterá os dados em conclusões significativas. Os dados provenientes de seus estudos de usabilidade poderão ajudar a justificar suas decisões de design, especialmente quando houver conflito entre os membros da equipe quanto ao modo como o aplicativo deveria funcionar.

Aqui está um exemplo. Enquanto estava estudando um redesign de nossa intranet corporativa, descobri que era significativamente menos eficiente para os usuários navegar por uma lista de itens, em vez de procurar por um item específico. Durante esse estudo, examinamos o diretório de nossos funcionários, que oferecia aos usuários a capacidade de pesquisar um funcionário e ver uma lista deles selecionando uma letra do sobrenome, conforme mostrado na figura 9.1.

Figura 9.1 — Nosso diretório original fornecia uma lista de funcionários aos usuários e a possibilidade de pesquisar alguém.

Quando solicitados a localizar um funcionário específico, mais de 90% dos funcionários clicavam para ver uma lista de todos os funcionários que tinham o sobrenome que começava com a mesma letra. Isso exigia que analisassem uma lista bem extensa. Alguns usuários gastavam mais de quatro minutos até localizar o funcionário que estavam procurando.

Quando perguntei por que eles optavam por visualizar uma lista de todos os funcionários, em vez de usar a caixa de pesquisa disponível, recebi uma resposta comum: eles tinham medo de digitar incorretamente o nome do funcionário. Eles acreditavam que seria melhor procurar a lista de todos os funcionários, em vez de correr o risco de inserir o nome incorretamente.

O estudo me ajudou a perceber que, se tivessem a opção entre pesquisar e visualizar uma lista de resultados, nossos funcionários não se sentiriam seguros em usar a pesquisa, mesmo que essa fosse uma opção muito mais eficiente. Sua preocupação principal era digitar incorretamente o nome de um funcionário.

Errar a grafia de um nome era um problema legítimo. Os nomes dos funcionários possuem grafias específicas e os sobrenomes podem ser longos. Poderiam ser necessárias diversas consultas para obter o nome que estavam procurando.

Desse modo, decidi incluir sugestões em nossa pesquisa, conforme mostrado na figura 9.2. Percebi que já tínhamos a lista dos nomes dos funcionários. Por que não criar uma maneira de sugerir todos os funcionários possíveis, à medida que o usuário

digitasse sua consulta? Isso reduziria os erros e daria mais confiança aos usuários para que contassem com a ferramenta de pesquisas, em vez de eles analisarem listas extensas.

Figura 9.2 — Pesquisa sobre funcionários revisada, usando sugestões.

Assim como ocorre com qualquer alteração, senti resistência por parte de alguns funcionários e supervisores. Eles estavam convencidos de que analisar as listas seria mais fácil do que digitar o nome de um funcionário. No entanto sua discordância era simplesmente insignificante quando comparada com meus dados.

Fui capaz de provar que o usuário médio gastava 21 minutos por ano procurando funcionários. Com mais de 3.500 funcionários sendo pagos a uma média de US$ 20,50 por hora, isso representava uma despesa estimada em US$ 25.000 por ano! Devo admitir que não é uma matemática muito exata (na realidade, não poderia ser mais confusa), mas a questão é que ter medidas permitiu provar aos opositores que suas pressuposições estavam incorretas.

É por isso que os estudos de usabilidade são o centro da metodologia de design centrado no usuário — eles corrigem suposições por meio de observações sistemáticas de usuários e pela coleta de seus feedbacks. No mundo do design centrado no usuário, não seguimos somente a intuição.

Acredito que, quando você começar a fazer estudos de usabilidade, não só irá fortalecer os relacionamentos com os usuários, mas poderá também ser um melhor programador. Você aprenderá o que funciona, o que não funciona e, mais importante ainda, terá os dados em que se apoiar.

A versão resumida

- Um estudo de usabilidade consiste no processo de testar o design de seu aplicativo ao observar os usuários, medir seu desempenho e documentar seus comentários.
- Assim como em todo o processo de design centrado no usuário, certifique-se de que seu estudo de usabilidade seja cuidadosamente planejado. Saiba quais tarefas estará testando e como planeja medi-las.
- Prepare um roteiro por escrito. Isso garantirá que a maneira pela qual você interagir com cada sujeito seja consistente e controlada.
- Enquanto os usuários estiverem executando a tarefa, incentive-os a pensar em voz alta. Isso o ajudará a entender seus processos de raciocínio e permitirá que você perceba aspectos que possam ter sido deixados de lado em seu aplicativo.
- Você deverá estar preparado com um cronômetro, um bloco de anotações, um local aceitável e um conjunto de sujeitos que melhor represente seu público-alvo.
- Quando estiver conduzindo o estudo, sente-se atrás do usuário para que você não o distraia com suas anotações. Após cada tarefa, considere entregar um questionário para que você possa terminar as últimas documentações e se preparar para a próxima tarefa.

- Praticar é uma ótima maneira de garantir que você preparou um estudo que proporcionará os melhores resultados possíveis.
- Após ter recebido os resultados do estudo, calcule suas medidas e organize seus comentários. Os dados ajudarão você a justificar suas decisões de design.

CAPÍTULO 10
Você nunca termina

"Aqui está o teste para descobrir se sua missão na Terra chegou ao fim. Se você estiver vivo, não chegou."

— Richard Bach

Espero que o tempo que passamos juntos tenha o ajudado a entender o valor de colocar os usuários no centro de seu processo de desenvolvimento. Sei que o design centrado no usuário não é fácil e exige uma boa dose de investimentos de sua parte. Contudo, acredito seriamente que o investimento vale a pena.

Quando começar a trabalhar em sincronia com os usuários, você economizará um tempo valioso ao colocar-se na direção correta. Não sei quanto a você, mas odeio ter de reescrever código ou jogar fora uma hora de trabalho. Sinto-me tolo quando coloco meu aplicativo diante do usuário e percebo que deixei passar algo óbvio, o que torna o aplicativo inutilizável. São necessárias somente algumas situações como essa para fazer você perceber que não é possível criar aplicativos sem a ajuda dos usuários.

No entanto não vou sugerir que o processo de design centrado no usuário seja o método mais rápido de colocar seu aplicativo na rua. Você continuará cometendo erros, mesmo quando os usuários lhe disserem exatamente de que eles necessitam. Adivinhe só? Os usuários mudam de ideia o tempo todo! Eles pedem uma coisa, veem e decidem que querem outra coisa.

É por isso que é importante não só ouvir suas necessidades, mas também observá-los para ter uma visão holística do domínio do

problema.

Tendo dito tudo isso, um dos aspectos mais importantes do design centrado no usuário está na disposição de fazê-lo da maneira correta. Você deverá estar disposto a recomeçar, se necessário. Afinal de contas, de que serve testar seu design se não estiver disposto a alterá-lo com base nos resultados? Isso exige que você deixe seu ego de lado para dar aos seus usuários o melhor aplicativo que puder criar.

É impossível fazer certo da primeira vez

Por mais difícil que seja, você deve abrir mão do sonho de que fará tudo certo na primeira versão de seu aplicativo. Terá de lutar contra a paralisia resultante de não desejar disponibilizar seu aplicativo até ter feito tudo perfeitamente.

Atinja um equilíbrio adequado entre aceitar que está disponibilizando um produto imperfeito e estar determinado a executar as tarefas corretamente.

Jeff Weir explica isso muito bem:

> Somente com a iteração seu produto ou o que quer que você crie poderá se tornar melhor. Você nunca conseguirá fazê-lo corretamente na primeira tentativa. *Ninguém consegue fazer corretamente na primeira tentativa.* Caso contrário, ainda teríamos o primeiro iPod.

Tendo desenvolvido aplicativos e sites já há algum tempo, posso dizer a você que continua sendo difícil ouvir críticas. Por mais que eu me orgulhe de ser resistente, ainda quero ouvir o usuário dizer “Está perfeito. Não será necessário mudar *nada*!”. Até hoje, acho que nunca ouvi isso, e não espero ouvir tão cedo.

Uma habilidade que cultivei ao longo dos vários anos consiste em distanciar meus valores pessoais e o meu próprio valor dos resultados de meus aplicativos. Com o risco de soar como um psicólogo, vou afirmar isso novamente porque merece ser repetido constantemente.

Devemos entender que, se nosso design falhar, isso não será uma *falha pessoal*. Devemos explorar o que deixamos passar ou as perguntas que podemos ter deixado de fazer. Se nosso design não funcionar, é sinal de que falhamos em obter as informações corretas; não significa que falhamos em ser programador e que deveríamos considerar outra profissão.

Ter a postura correta permitirá que você fique menos na defensiva quando receber *feedbacks*. Também o protegerá de se sentir desestimulado ou se sentir um fracasso, caso você crie algo de que os usuários não gostem.

Os usuários podem ser exigentes, mas minha experiência tem mostrado que, quando nós os incluímos no processo de desenvolvimento, eles têm maior compreensão acerca de como informar o que está faltando. Além do mais, eles têm melhor compreensão do que estão pedindo.

Os usuários que já trabalharam comigo no design de um aplicativo acabam adquirindo conhecimento com as dificuldades e as limitações técnicas. Devemos reconhecer que o fato de nossos usuários não serem formados em computação não significa que eles não sejam aptos a nos ajudar na solução de problemas técnicos.

Esteja preparado para reiniciar

Os desenvolvedores têm uma tendência de ser acumuladores de código, de se apaixonar pelo código e querer mantê-lo para sempre. Nós o deixamos em nossos aplicativos porque nunca se sabe se poderemos precisar dele ou se iremos querer atribuir outro propósito a ele. Nossos aplicativos tornam-se um armazém virtual para todas as nossas ideias de codificação.

Certamente, redefinir o propósito de um código tem seus méritos, e não há necessidade de reinventar a roda a cada projeto. No entanto isso tem suas desvantagens. Se instituirmos continuamente o mesmo código para cada projeto, correremos o risco de

implementar a mesma solução para todos os problemas. Pode se tornar mais difícil prover novas soluções para problemas de usabilidade. Acabamos avaliando toda solicitação de usuário como um exercício para provar que “temos código para ela”.

Se nosso aplicativo não atender às necessidades dos usuários, devemos fazer algumas perguntas básicas:

- O que estamos tentando realizar?
- Como estamos agregando valor?
- O que nossos usuários adoram?
- O que nossos usuários detestam?

O imenso desafio de criar um aplicativo faz com que seja fácil desviar-se dessas perguntas. Apesar disso, criar um ótimo aplicativo pode significar que teremos de voltar ao início. Pode significar que teremos de começar de novo e retornar à nossa visão. Pode ser necessário redefinir nossas pressuposições.

Sei como é difícil jogar código fora. Sei que você deve estar pensando “Levei meses para desenvolver este código! Agora você quer que eu o jogue fora porque os usuários não precisam dele?”.

Sim. É exatamente isso que eu quero que você faça.

Quando colocamos nosso código antes de nossos usuários, nós nos afastamos da visão que tínhamos inicialmente. Nossa tarefa consiste em criar excelentes aplicativos, e não um excelente código. Se seu código não der suporte à sua visão para o aplicativo, então ele deve ser descartado. É simples assim.

Desafie-se e à sua equipe a reavaliar a visão de seu projeto. Reserve continuamente um tempo em seu ciclo de desenvolvimento para garantir que está voltado na direção correta. Certifique-se de que as escolhas são as melhores para seus usuários e que não tenham sido feitas porque você tem medo de apagar código.

Seja corajoso e esteja disposto a reavaliar os pressupostos para garantir que ainda têm mérito à luz das novas descobertas.

É por isso que é tão importante ter uma definição de missão para a equipe ou um manifesto para seu projeto. Ao ter uma definição clara do que você está tentando realizar, você economizará o tempo que seria desperdiçado por sair do rumo. Retorne frequentemente à sua definição de missão e não hesite em usá-la quando estiver tomando decisões difíceis de design.

Últimas reflexões

Gostaria de fazer uma última observação antes de concluir. Alguns de nós, desenvolvedores, temos a tendência de ser egoístas. Há um nível de importância atribuída a si próprio e de poder que provêm da criação de aplicativos dos quais os usuários dependem. Com um clique de mouse ou uma tecla, podemos mudar tudo!

Por outro lado, muitos de nós sentimos uma pressão incrível para sermos *experts* e sabermos de tudo. Muitas vezes, nos vemos sozinhos, e é fácil se sentir desiludido ou frustrado. Em pouco tempo, seus usuários acabam se transformando no *inimigo*. Eles representam a horda revoltada, à sua porta, gritando “Quando isso ficará pronto?”.

Respire fundo. O design centrado no usuário exige habilidades afáveis. Você deve ouvir pacientemente os seus usuários e desejar genuinamente o seu *feedback*. Não estou dizendo que você deva *concordar* com o que eles estão dizendo, mas você deve, no mínimo, valorizar o que eles dizem. Lembre-se de que, no final, nossos usuários têm muito a nos ensinar.

Já tive usuários que me deram sugestões tão embaraçosamente simples que não sei como eu não havia pensado nelas antes. Afinal de contas, espera-se que eu seja o *expert*! A verdade é que é impossível ver tudo. Desenvolver aplicativos exige um foco tão intenso nos mínimos detalhes que tenho que abdicar de minha habilidade de perceber e de pensar em tudo.

Para fazer o trabalho corretamente é preciso ter ajuda, e não há

grupo melhor de pessoas para nos ajudar do que os próprios usuários.

Essa é a beleza da abordagem do design centrado no usuário. Ela nos protege do fracasso na abertura da cortina, o momento em que abrimos a cortina e mostramos nosso aplicativo a um grupo insatisfeito de usuários. Ela nos proporciona um roteiro que manterá nossa empatia com os usuários por focarmos continuamente em suas experiências.

O que oferecemos neste livro é uma rápida introdução. Há uma infinidade de conhecimentos no mundo da usabilidade e da experiência de usuário. Siga os *experts* e aprenda com eles. Se precisar de ajuda para encontrar alguns *experts*, consulte o capítulo 11.

Estimule sua própria criatividade ao reservar tempo para explorar novidades. Crie um plano para ajudá-lo a implementar sua estratégia de design centrado no usuário. Reúna *feedback* de seus usuários e faça perguntas específicas a eles sobre o domínio do problema. Crie protótipos para testar considerações iniciais sobre o design e permita que seus usuários avaliem.

Por fim, conduza estudos detalhados e organizados de usabilidade. Faça medidas da usabilidade de seu aplicativo observando os usuários quando estiverem tentando executar as tarefas. Documente seus comentários e use os dados para dar suporte às suas decisões de design.

Implemente a metodologia do design centrado no usuário apresentada neste livro, e você estará buscando o que há de melhor para seus usuários. Quando souberem que estão no foco de seu aplicativo, os usuários responderão continuamente com sua preferência e sua lealdade.

CAPÍTULO 11

Outros recursos

Neste livro, escrevi bastante sobre estimular sua criatividade seguindo os *experts* em cada área. Esta seção contém alguns dos melhores recursos e algumas das melhores ferramentas que já encontrei. Se você descobrir algo que esteja faltando, por favor, envie um email para *books@travislowdermilk.com*.

Twitter

O Twitter representa um recurso inestimável para todos os mais recentes acontecimentos no mundo da usabilidade e da experiência de usuário (UX, ou User Experience). Considere a criação de uma conta no Twitter para seguir estes ótimos recursos:

Aarron Walter @aarron

Diretor de UX na MailChimp e autor de *Designing for Emotion* (A Book Apart).

A Book Apart @abookapart

Editora que publica alguns livros muito bons sobre design e desenvolvimento para web. Esses livros me inspiraram a escrever este!

Dan Cederholm @simplebits

Cofundador do Dribbble, um site para demonstração e compartilhamento para designers, e do estúdio de design para web SimpleBits LLC. Dan é um designer extremamente talentoso e oferece uma grande quantidade de ideias sobre o mundo da usabilidade e do design.

Design Observer @designobserver

O grupo hospeda um site expansivo sobre o mundo do design (*http://www.designobserver.com*). Eles também produzem uma série incrível de podcasts chamada “Design Matters” (Design é Importante), apresentada por Debbie Millman.

EffectiveUI @effectiveui

Agência de design de experiência de usuário com sede em Denver, Colorado, que usa o Twitter para ajudar as pessoas a ficar por dentro das principais tendências em design e das mudanças no mercado. Foi responsável também pelo livro *Effective UI* (O’Reilly), que usei para preparar a seção sobre estudos de usabilidade.

Human Factors International @humanfactors

Um líder mundial em design centrado no usuário que mantém você conectado com as tendências emergentes apresentando artigos relacionados.

Luke Wroblewski @lukew

Um líder internacionalmente reconhecido em produtos digitais e autor de *Mobile First* (A Book Apart), que argumenta que os sites devem ser concebidos para levar dispositivos móveis em consideração.

Measuring Usability @MeasuringU

Jeff Sauro é autor de *Quantifying the User Experience: Practical Statistics of User Research* (Morgan Kaufmann) e fundador da Measuring Usability, uma empresa de pesquisas que ajuda outras empresas a quantificar suas experiências de usuário.

Nick Campbell @nickvegas

Campbell é um designer divertido e incrivelmente talentoso. Possui um site de design de animações bem como um podcast de design em geral chamado Greyscalegorilla

(*http://www.greyscalegorilla.com*).

Nielsen Norman Group @NNgroup

Fundado por Jakob Nielson e Don Norman, esse grupo é a agência líder em usabilidade e design centrado no usuário.

Roman Mars @romanmars

Roman é apresentador do 99% Invisible (*http://www.99percentinvisible.org*), uma série de podcasts absolutamente fantástica sobre design.

Typographica.org @typographica

Do editor Stephen Coles, sobre fontes e tipografia.

User Interface Engineering @UIE

Uma empresa de consultoria, líder em pesquisas e treinamento, especializada em UX e em usabilidade de sites e de produtos. Tem sugestões ótimas para todo o espectro de problemas de usabilidade.

UX Magazine @uxmag

Uma publicação online sobre o complexo campo da experiência de usuário. É composta de escritores bastante talentosos que escrevem artigos instigantes e cativantes.

Ferramentas para prototipagem

Ha muitos aplicativos disponíveis para prototipagem, mas esses dois parecem corresponder aos padrões do mercado no momento:

Balsamiq Mockups

Com foco em agilidade, o Balsamiq (*http://www.balsamiq.com*) é uma ótima ferramenta para apresentar design, mockups e layouts iniciais. O Balsamiq usa um efeito de desenho feito à mão em todos os seus widgets, o que cria uma experiência que parece ser conceitual e iterativa.

Axure RP Pro

O Axure RP Pro (*http://www.axure.com*) proporciona uma grande dose de sofisticação para uma ferramenta de prototipagem. O Axure permite usar widgets, e, ao ajustar o modo de esboçar, você pode mudar rapidamente a fidelidade de seu protótipo. Além do mais, o Axure RP Pro oferece eventos robustos e um framework condicional para definir ações básicas que ocorrem em resposta a cliques e pressionamentos de teclas.

Sites

alistapart.com

Uma revista que explora design, estratégia de conteúdo, desenvolvimento, padrões e práticas. O foco principal está no desenvolvimento para web.

abookapart.com

Criada pelo pessoal do *A List Apart*, A Book Apart publicou uma coleção de livros que focam em tópicos como HTML5, CSS, estratégia de conteúdo e muito mais. O que há de melhor nesses livros são seu tamanho e o tom informal. São fáceis de ler e são divertidos!

boxesandarrows.com

Dedicado à prática e a discussões sobre design, design para interação, arquitetura de informações e negócios sobre design.

netmagazine.com

.net é uma revista para designers e desenvolvedores para web. Os assuntos variam de tutoriais sobre as últimas tendências em design a entrevistas com os maiores designers da Web.

smashingmagazine.com

Uma das editoras líderes globais em publicações online no campo de design e de desenvolvimento para web. Esse site representa a

culminância de artigos diários e dicas dos principais *experts* do mercado em design de UX.

useit.com/alertbox

De autoria de Jakob Nielsen, um dos *experts* mais importantes no campo da usabilidade e da experiência de usuário. O site está repleto de artigos com suas opiniões sobre as tendências mais recentes em design. Se você quiser estar onde a discussão sobre usabilidade começa, esse é o lugar.

uxmatters.com

Oferece ideias e inspiração a profissionais de todos os níveis, à medida que fazem sua jornada pelo mundo da experiência de usuário.

APÊNDICE A
Exemplo de template para projetos

Template

Nós nos referimos a nosso template como Ciclo de Vida do Desenvolvimento de Software (CVDS). Ele não representa um CVDS totalmente abrangente. Muitas empresas se utilizam de processos mais elaborados para desenvolvimento e documentação. Assim como no que se refere a todas as recomendações presentes neste livro, esse template deve ser usado como ponto de partida, e você deverá alterá-lo para que se torne adequado às suas necessidades.

Título do projeto

Template de Projeto Ciclo de Vida do Desenvolvimento de Software

Definição da missão

Resumo do Ciclo de Vida do Desenvolvimento de Software

O propósito deste documento é coletar, definir e organizar os detalhes e os requisitos do projeto. O template foi concebido para ser iterativo, o que significa que ele irá evoluir ao longo da vida do projeto.

O template inclui as seguintes seções-chave:

1. Detalhes do projeto: inclui o resumo do projeto. Foi concebido para reunir as seguintes informações:
2. Título**:** o título ou codinome do projeto.
3. Descrição**:** um resumo definindo o objetivo do projeto.
4. Pessoas-chave: pessoas que estão envolvidas ou que solicitaram o projeto.
5. Avaliação de impacto: o impacto observado ou que se espera do produto final resultante do projeto. Quem será afetado por esse projeto? Quantos usuários? Quais processos do negócio serão alterados como resultado desse projeto?
6. **Requisitos de usuário:** os requisitos do projeto, conforme especificados pelo usuário. A seção de requisitos de usuário inclui uma assinatura para expressar concordância. À medida que os requisitos evoluírem, muitas iterações das páginas contendo os requisitos de usuário poderão ser incluídas.
7. Páginas contendo especificações (requisitos funcionais): os detalhes técnicos do projeto, ou os produtos que serão desenvolvidos e/ou projetados para atender às necessidades do usuário.
8. **Modelos de dados:** Diagramas de Fluxo de Dados e/ou Diagramas de Estrutura de Dados.
9. Processos de dados: são os processos de dados ou roteiros dos quais o projeto depende. A lista inclui o nome do processo, uma

descrição do local em que está localizado, o que ele faz, entre outros itens, e o tempo de execução/duração. Essa tabela será atualizada durante a vida do produto.

10. **Protótipos:** cópias de telas de baixa ou alta fidelidade, mockups, modelos etc.
11. Notas para manutenção: notas adicionais, pertinentes para dar suporte a esse produto.

Detalhes do projeto

Preencha os vários detalhes abaixo para descrever seu projeto. Caso sejam feitas alterações, use texto tachado e insira a data em que a alteração foi feita. Em seguida você poderá acrescentar o título revisado, as descrições etc.

IMPORTANTE: não apague dados, pois eles são o registro oficial do desenvolvimento de seu projeto!

Título

Título do projeto ou do produto

Descrição

Descrição do projeto ou do produto

Nome(s) da(s) pessoa(s)-chave

Nome(s) da(s) principais pessoa(s)-chave

Avaliação de impacto

Um resumo do impacto do projeto/produto

Requisitos de usuário

Preencha os vários requisitos de usuário abaixo. Caso sejam feitas alterações, use texto tachado e insira a data em que a alteração foi feita. Em seguida você poderá adicionar os requisitos de usuário novos ou revisados.

IMPORTANTE: não apague requisitos de usuário, pois eles são o registro oficial do desenvolvimento de seu projeto!

	Requisito de usuário	Descrição	Nome do solicitante
1			
2			
3			
4			
5			

Confirmo que esses requisitos de usuário atendem às necessidades de minha solicitação. Se houver mudanças nos requisitos de usuário, notificarei o desenvolvedor por meio de uma nova solicitação. Estou ciente de que fazer alterações nesses requisitos de usuário pode provocar atraso no prazo de entrega estimado para esse projeto.

_____

Assinatura da pessoa-chave Data

Página de especificações (requisitos funcionais)

Preencha os vários requisitos funcionais abaixo. Caso sejam feitas alterações, use texto tachado e insira a data em que a alteração foi feita. Em seguida, você poderá adicionar os requisitos funcionais novos ou revisados.

IMPORTANTE: não apague requisitos funcionais, pois eles são o registro oficial do desenvolvimento de seu projeto!

	Requisito funcional	Descrição	Atende ao requisito de usuário número
1			
2			
3			
4			
5			

Confirmo que esses requisitos funcionais atendem às necessidades de minha solicitação. Se houver mudanças nos requisitos funcionais, notificarei o desenvolvedor por meio de uma nova solicitação. Estou ciente de que fazer alterações nesses requisitos funcionais pode provocar atraso no prazo de entrega estimado para esse projeto.

_____

Assinatura da pessoa-chave Data

Modelos de dados e de fluxo de trabalho

Inclua diagramas que mostrem a estrutura de dados em alto nível. A intenção desse processo é documentar como os dados são estruturados para dar suporte ao projeto.

Processos de dados

	Nome do processo	Descrição	Tempo de execução/frequência
1			
2			
3			
4			
5			

Protótipos

Inclua protótipos de alta e/ou baixa fidelidade. Você pode incluir esboços iniciais ou capturas de tela. O objetivo da prototipagem é dar aos usuários a oportunidade de revisar modelos conceituais iniciais. Pode ser útil incluir até mesmo os primeiros designs conceituais, para que você possa manter um registro do progresso de seu aplicativo.

Notas para manutenção

Inclua notas que serão necessárias para prover suporte a seu produto. Essa parte do documento deve ser atualizada durante a vida do produto. À medida que os requisitos para suporte sofrerem alterações, complemente esse documento com uma data ao lado de cada entrada.

Exemplo de persona

Dan Welks

Idade: 29

Estado civil: casado, 1 filho (2 anos)

Residência: Austin, Texas

Profissão: designer de web

Hobbies: leitura (em sua maior parte, blogs técnicos), tocar violão, jogar videogames e fotografia.

Itens preferidos: iPad, iPhone, rádios AM/FM antigos e o novo automóvel SUV que ele e sua esposa acabaram de comprar.

Necessidades: Dan carrega seu iPad onde quer que vá. Ele adoraria ter uma maneira de fazer anotações ou criar esboços rápidos para o design de um site .

Atualmente, Dan esboça suas ideias em papel ou em um quadro branco. Às vezes, ele tira uma foto de seus desenhos com seu iPad, mas isso não permite que ele faça alterações posteriormente.

Dan gostaria que houvesse uma maneira de organizar todas as suas anotações e seus rabiscos em um só lugar. Ele tentou carregar consigo um caderno e uma caneta, mas é desconfortável carregar esses itens, além de carregar seu iPad.

Crenças: Dan experimentou outros apps para desenhar, esboçar e fazer anotações em seu iPad. Nenhum deles foi muito satisfatório. Ele gosta do Sketchbook Pro, mas acha que as ferramentas são

complexas e intimidadoras. Ele já tem o Photoshop, de modo que, se quiser criar algo mais detalhado, ele simplesmente irá usá-lo.

Dan acredita que a melhor maneira de desenhar e fazer anotações é usando papel e caneta. Ele acha que desenhar em um iPad não proporciona agilidade e não dá a impressão de naturalidade.

Roteiro de exemplo para um estudo de usabilidade

Este é um modelo de roteiro para um estudo de usabilidade, e seu objetivo é ajudar você a ter um ponto de partida. Obviamente, as tarefas e o foco de seu estudo serão determinados por você. No entanto este template dará uma ideia da linguagem e da organização de um roteiro. Após a introdução, você poderá criar um roteiro similar para cada tarefa.

Essencialmente, cada roteiro deve ter o mesmo tom e a mesma linguagem da introdução. A única diferença está no fato de que ele terá instruções específicas para as tarefas que você observará.

Introdução ao estudo

Obrigado por concordar em participar do teste do *nome do app*, um aplicativo que permitirá que você *breve descrição do aplicativo*.

Hoje, vamos testar três recursos principais do aplicativo:

- *Recurso*
- *Recurso*
- *Recurso*

O estudo deve durar *tempo estimado*.

A primeira tarefa exigirá que você *breve descrição da primeira tarefa*. A seguir, você será solicitado a *breve descrição da segunda tarefa*. Por fim, observarei você a *breve descrição da terceira tarefa*.

Antes de cada tarefa, descreverei os objetivos específicos da tarefa para você. Cada tarefa começará quando eu disser “Início da tarefa” e terminará quando eu disser “Fim da tarefa”. Não poderei responder a perguntas específicas durante a tarefa; porém posso esclarecer qualquer instrução. Antes de iniciar cada tarefa, perguntarei se você tem alguma dúvida sobre as instruções.

Por favor, narre seu processo de raciocínio expressando seus pensamentos em voz alta. Por exemplo, se você for *ação apropriada*, você deve dizer em voz alta “Vou *ação apropriada*”. Isso me ajudará a entender o que você está tentando fazer e aumentará a eficiência deste estudo.

Por favor, lembre-se de que não estamos testando você neste estudo. Estamos testando somente a capacidade do aplicativo de ajudá-lo na execução de sua tarefa. Todas as anotações, a documentação e os comentários serão totalmente anônimos.

APÊNDICE B
Referências

Citação de Steve Jobs sobre grupos de foco

Bloomberg Business Week, de 12 de maio de 1998. Steve Jobs on Apple's Resurgence: "Not a One-Man Show" (Steve Jobs sobre o ressurgimento da Apple: "Não é um espetáculo de um homem só"). *http://buswk.co/Yf7HCQ*.

Citação de Steve Jobs sobre a intersecção entre artes liberais e tecnologia

Jobs, Steve. 27 de janeiro de 2010. Apresentação do iPad pela Apple. San Francisco, CA.

O que é usabilidade?, Coletando requisitos de usuário, Criando requisitos funcionais, Princípio de feedback, Consistência, Lei de Fitt, Estudando usuários, Escalas de Likert, O que são estudos de usabilidade?

Rogers, Yvonne, Helen Sharp e Jenny Preece. 2011. *Design de interação: além da interação homem-computador*. 3. ed. Bookman.

Usabilidade não é perda de tempo (nem de dinheiro), Usando protótipos

Hollis, Billy. 21 de junho de 2012. Entrevista pessoal.

Lund, Arnold M. Maio/junho de 1997. "Another approach to justifying the cost of usability" (Outra abordagem para justificar o custo da usabilidade). *Interactions*.

Saber quando ouvir (e quando não ouvir) os usuários, O obcecado por controle

Goodman, Elizabeth, Mike Kuniavsky e Andrea Moed. 2012.

Observing the user experience: A practitioner's guide to user research. Waltham: Morgan Kaufmann.

Lidando com diferentes tipos de usuários: O "Advogado do Diabo"

Kelley, Tom e Jonathan Littman. 2005. *As 10 faces da inovação: as estratégias para turbinar a criatividade*. Campus.

Lidando com a negatividade e a atribuição de erros

Heath, Chip e Dan Heath. 2010. *A guinada: maneiras simples de operar grandes transformações*. Best Business.

Planejar

Anderson, Jonathan, John McRee, Robb Wilson e a equipe da EffectiveUI. 2010. *Effective UI*. Cambridge: O'Reilly Media, Inc.

Documentando protótipos

Fitzpatrick, Brian W. e Ben Collins-Sussman. 2012. *Equipes de software: um guia para o desenvolvedor de software se relacionar melhor com outras pessoas*. São Paulo: Novatec.

Criando um manifesto pessoal

Ingebretsen, Robby. 14 de abril de 2012. "Paper, manifestos and why you need one to be great at anything" (Paper, manifestos e por que é preciso ter um para ser ótimo em tudo). *http://nerdplusart.com/paper-manifestos-and-why-you-need-one-to-be-great-at-anything*.

Paper para iPad, da FiftyThree, Consistência

Hamburger, Ellis. 29 de março de 2012. "Paper: the next great iPad app, from the brains behind Courier" (Paper: o próximo excelente app para iPad, dos cérebros responsáveis pelo Courier). *The Verge*. *http://www.theverge.com/2012/3/29/2909537/paper-drawing- ipad-app-fiftythree-brains-behind-courier*.

Exercitando restrições

Buxton, Bill. 12 de janeiro de 2007. "Multi-touch systems that I

have known and loved" (Sistemas multi-touch que conheci e amei). *http://www.billbuxton.com/multitouchOverview.html*.

Construindo uma narrativa

Walker, Julian. 1 de maio de 2012. Entrevista pessoal.

Criatividade exige coragem (e trabalho árduo)

Hogan, Blaine. 2011. *Untitled: Thoughts on the creative process*. Alpharetta: Clark. *http://creativecollective.is/portfolio/untitled-thoughts-on-the-creative-process/*.

Roube (quero dizer, empreste) de outros

Kleon, Austin. 2012, *Steal like an artist: 10 things nobody told you about being creative*. Nova York: Workman Publishing Company.

Heffron, Jack. 2003. *The writer's idea workshop: How to make your good ideas great*. Cincinnati: Writer's Digest Books.

Criatividade exige questionamento

Lehrer, Jonah. *Imagine: How creativity works*. Houghton Mifflin Harcourt Publishing Company. Livro digital.

Pegue um lápis (da apresentação de Jeff Gothelf no Agile UX NYC 2012)

Constable, Giff. 29 de fevereiro de 2012. "Increase collaboration through sketching" (Aumente a colaboração por meio de esboços). *http://giffconstable.com/2012/02/increase-collaboration-through-sketching/*.

Liberdade de criação

Pink, Daniel H. 2009. *Drive: The surprising truth about what motivates us*. Nova York: Riverhead Books.

Entendendo sua meta

Weir, Jeff. 20 de maio de 2012. Entrevista pessoal.

Princípio de proximidade (Princípio da Gestalt), Modelos mentais e metáforas, Princípio da visibilidade e Lei de Hick

Lidwell, William, Kritina Holden e Jill Butler. 2003. *Princípios universais do design*. Bookman.

Disponibilidade e restrições

Norman, Donald A. 1988. *O design do dia-a-dia*. Rocco.

Lei de Hick, Lei de Fitt

Wickens, Christopher, John Lee, Yili Liu e Sallie Gordon Becker. 2004. *An introduction to human factors engineering*. Pearson Education, Inc.

Quantos usuários serão necessários?

Nielsen, Jakob. 19 de março de 2000. “Why you only need to test with 5 users” (Por que você só precisa fazer testes com cinco usuários). *http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/*.

Distorção da concordância nas respostas

Lavrakas, Paul J. 2008. Encyclopedia of Survey Research Methods. Sage Publications, Inc. Referência online. *http://bit.ly/10PorRi*.

Testes A/B

Christian, Brian. 25 de abril de 2012. “The A/B test: Inside the technology that’s changing the rules of business” (O teste A/B: por dentro da tecnologia que está mudando as regras dos negócios). *Wired Magazine*. *http://www.wired.com/business/2012/04/ff_abtesting/*.

Atwood, Jeff. 20 de janeiro de 2010. “Groundhog Day, or, the problem with A/B testing” (Feitiço do Tempo, ou o problema com os testes A/B). *Coding Horror: Programming and Human Factors*. *http://www.codinghorror.com/blog/2010/07/groundhog-day-or-the-problem-with-ab-testing.html*.

O que são estudos de usabilidade?, Criando um plano de testes, De que você precisará?, Conduzindo o estudo

Pernice, Kara, Amy Schade e Jakob Nielsen. *Intranet Usability Guidelines, vol. 1: Understanding and Studying Users (Test Data, User Behavior, and Methodology 2nd Edition)*. Nielsen Norman Group. Documento eletrônico. *http://www.nngroup.com*.

O'REILLY

Padrões para Kubernetes

Elementos reutilizáveis no design de aplicações nativas de nuvem

Bilgin Ibryam
Roland Huß

novatec

Padrões para Kubernetes

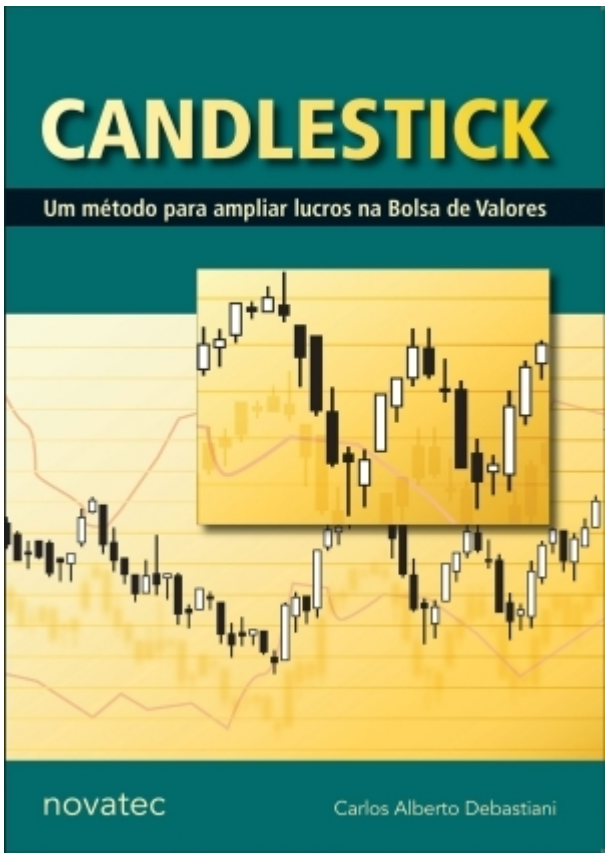
Ibryam, Bilgin

9788575228159

272 páginas

O modo como os desenvolvedores projetam, desenvolvem e executam software mudou significativamente com a evolução dos microsserviços e dos contêineres. Essas arquiteturas modernas oferecem novas primitivas distribuídas que exigem um conjunto diferente de práticas, distinto daquele com o qual muitos desenvolvedores, líderes técnicos e arquitetos estão acostumados. Este guia apresenta padrões comuns e reutilizáveis, além de princípios para o design e a implementação de aplicações nativas de nuvem no Kubernetes. Cada padrão inclui uma descrição do problema e uma solução específica no Kubernetes. Todos os padrões acompanham e são demonstrados por exemplos concretos de código. Este livro é ideal para desenvolvedores e arquitetos que já tenham familiaridade com os conceitos básicos do Kubernetes, e que queiram aprender a solucionar desafios comuns no ambiente nativo de nuvem, usando padrões de projeto de uso comprovado. Você conhecerá as seguintes classes de padrões: • Padrões básicos, que incluem princípios e práticas essenciais para desenvolver aplicações nativas de nuvem com base em contêineres. • Padrões comportamentais, que exploram conceitos mais

específicos para administrar contêineres e interações com a plataforma. • Padrões estruturais, que ajudam você a organizar contêineres em um Pod para tratar casos de uso específicos. • Padrões de configuração, que oferecem insights sobre como tratar as configurações das aplicações no Kubernetes. • Padrões avançados, que incluem assuntos mais complexos, como operadores e escalabilidade automática (autoscaling).


CANDLESTICK
Um método para ampliar lucros na Bolsa de Valores
novatec
Carlos Alberto Debastiani

Candlestick

Debastiani, Carlos Alberto

9788575225943

200 páginas

A análise dos gráficos de Candlestick é uma técnica amplamente utilizada pelos operadores de bolsas de valores no mundo inteiro. De origem japonesa, este refinado método avalia o comportamento do mercado, sendo muito eficaz na previsão de mudanças em tendências, o que permite desvendar fatores psicológicos por trás dos gráficos, incrementando a lucratividade dos investimentos. Candlestick – Um método para ampliar lucros na Bolsa de Valores é uma obra bem estruturada e totalmente ilustrada. A preocupação do autor em utilizar uma linguagem clara e acessível a torna leve e de fácil assimilação, mesmo para leigos. Cada padrão de análise abordado possui um modelo com sua figura clássica, facilitando a identificação. Depois das características, das peculiaridades e dos fatores psicológicos do padrão, é apresentado o gráfico de um caso real aplicado a uma ação negociada na Bovespa. Este livro possui, ainda, um índice resumido dos padrões para pesquisa rápida na utilização cotidiana.

Compre agora e leia


AVALIANDO
EMPRESAS
INVESTINDO
EM AÇÕES
A APLICAÇÃO PRÁTICA DA
ANÁLISE FUNDAMENTALISTA NA
AVALIAÇÃO DE EMPRESAS
novatec
CARLOS ALBERTO DEBASTIANI
FELIPE AUGUSTO RUSSO

Avaliando Empresas, Investindo em Ações

Debastiani, Carlos Alberto

9788575225974

224 páginas

Avaliando Empresas, Investindo em Ações é um livro destinado a investidores que desejam conhecer, em detalhes, os métodos de análise que integram a linha de trabalho da escola fundamentalista, trazendo ao leitor, em linguagem clara e acessível, o conhecimento profundo dos elementos necessários a uma análise criteriosa da saúde financeira das empresas, envolvendo indicadores de balanço e de mercado, análise de liquidez e dos riscos pertinentes a fatores setoriais e conjunturas econômicas nacional e internacional. Por meio de exemplos práticos e ilustrações, os autores exercitam os conceitos teóricos abordados, desde os fundamentos básicos da economia até a formulação de estratégias para investimentos de longo prazo.

Compre agora e leia


Marcos Abe
MANUAL DE
ANÁLISE
TÉCNICA
ESSÊNCIA E ESTRATÉGIAS AVANÇADAS
TUDO O QUE UM INVESTIDOR PRECISA SABER PARA
PROSPERAR NA BOLSA DE VALORES ATÉ EM TEMPOS DE CRISE
novatec

Manual de Análise Técnica

Abe, Marcos

9788575227022

256 páginas

Este livro aborda o tema Investimento em Ações de maneira inédita e tem o objetivo de ensinar os investidores a lucrarem nas mais diversas condições do mercado, inclusive em tempos de crise. Ensinará ao leitor que, para ganhar dinheiro, não importa se o mercado está em alta ou em baixa, mas sim saber como operar em cada situação. Com o Manual de Análise Técnica o leitor aprenderá: - os conceitos clássicos da Análise Técnica de forma diferenciada, de maneira que assimile não só os princípios, mas que desenvolva o raciocínio necessário para utilizar os gráficos como meio de interpretar os movimentos da massa de investidores do mercado; - identificar oportunidades para lucrar na bolsa de valores, a longo e curto prazo, até mesmo em mercados baixistas; um sistema de investimentos completo com estratégias para abrir, conduzir e fechar operações, de forma que seja possível maximizar lucros e minimizar prejuízos; - estruturar e proteger operações por meio do gerenciamento de capital. Destina-se a iniciantes na bolsa de valores e investidores que ainda não desenvolveram uma metodologia própria para operar lucrativamente.


Fundos de
Investimento
Imobiliário
ASPECTOS GERAIS E
PRINCÍPIOS DE ANÁLISE
novatec
Roni Antônio Mendes

Fundos de Investimento Imobiliário

Mendes, Roni Antônio

9788575226766

256 páginas

Você sabia que o investimento em imóveis é um dos preferidos dos brasileiros? Você também gostaria de investir em imóveis, mas tem pouco dinheiro? Saiba que é possível, mesmo com poucos recursos, investir no mercado de imóveis por meio dos Fundos de Investimento Imobiliário (FIIs). Investir em FIIs representa uma excelente alternativa para aumentar o patrimônio no longo prazo. Além disso, eles são ótimos ativos geradores de renda que pode ser usada para complementar a aposentadoria. Infelizmente, no Brasil, os FIIs são pouco conhecidos. Pouco mais de 100 mil pessoas investem nesses ativos. Lendo este livro, você aprenderá os aspectos gerais dos FIIs: o que são; as vantagens que oferecem; os riscos que possuem; os diversos tipos de FIIs que existem no mercado e como proceder para investir bem e com segurança. Você também aprenderá os princípios básicos para avaliá-los, inclusive empregando um método poderoso, utilizado por investidores do mundo inteiro: o método do Fluxo de Caixa Descontado (FCD). Alguns exemplos reais de FIIs foram estudados neste livro e os resultados são apresentados de maneira clara e didática, para que você aprenda a conduzir os próprios estudos e tirar as próprias

conclusões. Também são apresentados conceitos gerais de como montar e gerenciar uma carteira de investimentos. Aprenda a investir em FIIs. Leia este livro.

Compre agora e leia