Aprendendo Programação

iOS

Alasdair Allan

Novatec

Editor: Rubens Prates
Tradução: Rafael Zanolli
Revisão técnica: Aurelio Jargas
Revisão gramatical: Marta Almeida de Sá
Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-363-5

Histórico de impressões:

Julho/2013 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
Email: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

CAPÍTULO 1

Por que seguir o caminho nativo?

Quando o iPhone foi originalmente apresentado, em 2007, não havia nenhum SDK nativo. A Apple afirmava que um SDK desse tipo não era necessário e que aplicativos para o dispositivo deveriam ser construídos como aplicativos web utilizando JavaScript, CSS e HTML. Isso não foi bem digerido pela comunidade de desenvolvedores; eles queriam acesso direto ao hardware e integração com os próprios aplicativos da Apple.

Apenas alguns poucos meses depois do lançamento do iPhone, a comunidade de código aberto havia realizado algo que muitos consideravam impossível. Apesar de a Apple bloquear o dispositivo, desenvolvedores haviam obtido acesso, feito a engenharia reversa do SDK e seguido para construir uma cadeia de ferramentas de código-aberto gratuita que lhes permitia desenvolver aplicativos nativos para o dispositivo. Em determinado momento, estimava-se que mais de um terço dos iPhones no mercado havia sido desbloqueado com "jailbreak"[1] por seus usuários, permitindo que eles executassem esses aplicativos não sancionados de terceiros.

No entanto o livro que você tem em mãos não trata do SDK "hacker" de código aberto, pois, em março de 2008, menos de um ano depois do lançamento do primeiro iPhone, a Apple mudou de opinião publicamente e lançou um SDK nativo para uma comunidade de desenvolvedores que estava à espera. Se esse lançamento foi ou não uma resposta aos esforços da comunidade de código aberto, ou se esse foi – o tempo todo – exatamente o plano de uma Apple notoriamente reservada, isso provavelmente nunca saberemos.

1 N.T.: O Jailbreak é um processo que permite que aparelhos com o sistema iOS (iPhone, iPod Touch e iPad) executem aplicativos não autorizados pela fabricante Apple. Um aparelho com Jailbreak é capaz de baixar aplicativos anteriormente indisponíveis pela App Store por meio de instaladores não oficiais como o Cydia, assim como aplicações adquiridas de forma ilegal (fonte: Wikipédia).

Os prós e os contras

Quando o SDK nativo foi apresentado, muitas pessoas do setor argumentaram que ele era, na realidade, um passo atrás para os desenvolvedores. Eles sentiam que os aplicativos web eram suficientemente bons. Ao escrever um código especificamente para o iPhone em Objective-C, você estava tornando mais difícil portar seus aplicativos, e a portabilidade de um aplicativo web consistia mais ou menos em simplesmente reestilizá-lo utilizando um novo template CSS.

Parecia que os usuários dos aplicativos discordavam. Pode-se discutir porque isso ocorreu, mas é muito difícil desenvolver aplicativos web de aparência nativa que possam ser reutilizados em muitas plataformas diferentes, ainda que isso seja possível. Da mesma forma que os aplicativos do desktop Mac que foram portados do Windows tendem a incomodar como um polegar dolorido por não funcionarem como espera o usuário, aplicativos web, especialmente aqueles que devem ser utilizados em plataformas distintas, tendem a fazer o mesmo.

Apenas para deixar claro, isso não é uma crítica ao HTML, às CSS e ao JavaScript. O acervo de tecnologias da web é sólido e está amadurecendo, e a atração de termos uma base de código de multiplataforma será imediatamente óbvia para você como desenvolvedor. No entanto, para seus usuários, isso não é tão óbvio. A maioria dos argumentos apresentados em suporte ao uso de tecnologias web, ou de outros frameworks não nativos, é feita a partir da perspectiva do desenvolvedor. Ainda que isso não os torne inválidos, você deve analisar cuidadosamente esses argumentos pela perspectiva do usuário, pois nenhum deles trata realmente da principal consideração: a usabilidade.

Consumidores não comprarão seu aplicativo na plataforma deles simplesmente porque você suporta outras plataformas; em vez disso, eles querem um aplicativo que tenha o mesmo visual dos outros aplicativos da plataforma deles, que siga os mesmos paradigmas de interface do restante dos aplicativos com que eles estão acostumados e que esteja integrado a essa plataforma.

Se você integrar seu aplicativo ao ambiente do iOS, fizer uso das possibilidades que o hardware oferece e se certificar de que sua interface de usuário está otimizada para o dispositivo, a experiência de usuário será muito melhor. Tudo isso é possível utilizando tecnologias web, mas, na maior parte do tempo, será realmente difícil escrever aplicativos não nativos que funcionem bem em várias plataformas. É ainda mais difícil fazer com que eles pareçam nativos em meia dúzia de plataformas distintas.

Por que escrever aplicativos nativos?

A razão óbvia de utilizarmos um SDK nativo é fazer algo que não pode ser feito utilizando tecnologias web. A primeira geração de aplicativos de realidade aumentada é um exemplo disso; esses aplicativos necessitavam de uma integração mais próxima com os sensores do iPhone (por exemplo, GPS, acelerômetro, bússola digital e câmera) e não teriam sido possíveis sem esse acesso.

Ainda que o navegador Safari no iOS suporte as capacidades de localização geográfica que o HTML 5 fornece (*http://www.w3.org/TR/geolocation-API/*), isso não elimina o problema por completo. É duvidoso que todo o hardware específico da plataforma obtenha o mesmo tipo de tratamento em HTML 5, por isso é improvável que você veja a chegada de aplicativos web de realidade aumentada.

Se você vem de uma experiência com desenvolvimento web, pode estar interessado no framework multiplataforma PhoneGap (*http://phonegap.com/*). Esse framework fornece classes encapsuladoras nativas e permite que você desenvolva aplicativos nativos em HTML/JavaScript para várias plataformas móveis. Uma das plataformas às quais ele se direciona é o iOS.

Às vezes, não se trata de fazer algo que não pode ser feito, mas de fazer tudo mais rápido e de realizar a manipulação de erros no lado do cliente. Por exemplo, os aplicativos do iTunes e da App Store da Apple que são fornecidos com o iOS são, na realidade, aplicativos web encapsulados dentro de aplicativos nativos. Exatamente como o iTunes e a App Store no desktop Mac, o painel principal que você vê é, na realidade, apenas uma página web, mas a infraestrutura circundante é um aplicativo nativo. Isso significa que, ainda que o aplicativo não possa fazer muito sem uma conexão de internet, ele pode pelo menos ser inicializado sem problemas.

Porém esses são exemplos extremos. Muitos dos aplicativos da App Store combinam dados remotos e interfaces nativas. Sem acesso à rede, parte da UI está geralmente desabilitada. No entanto aplicativos nativos podem ser construídos de modo que ocorra com eles uma degradação elegante quando a conexão de rede do dispositivo desaparecer ou caso ela nunca esteja presente em primeiro lugar. O usuário ainda pode ver os elementos do aplicativo que não necessitam de uma conexão de rede para funcionar.

Às vezes, também devemos considerar aquilo de que um aplicativo não necessita. Se ele não necessita de uma conexão de rede, manter seu telefone conectado à rede para utilizá-lo, consumindo energia extra da bateria no processo, é um desperdício. Mesmo quando ele está conectado, o dispositivo nem sempre tem acesso

a uma conexão de internet rápida. Tudo que você puder fazer para minimizar a quantidade de dados que você precisa "puxar" pela conexão de rede vai melhorar a interação dos usuários com seu aplicativo. Isso significa gerar sua interface de usuário de modo local e preenchê-la com dados acessados da internet.

O desempenho de rede afetará a percepção do usuário da velocidade; renderizar sua interface de usuário enquanto uma solicitação web é feita para preenchê-la permite que seu aplicativo permaneça responsivo à interação do usuário, mesmo enquanto ele espera pela rede. Isso só pode ser algo bom.

Ainda nem sequer mencionei o desenvolvimento de jogos, e, com a Apple divulgando o iPod touch como o "iPod mais divertido de todos os tempos", isso é algo importante. Você não pode desenvolver os tipos de jogos que agora começam a aparecer na App Store utilizando tecnologias web. Ainda que este livro aborde o básico de como programar para dispositivos iOS, se você quiser se aprofundar na programação de jogos nessa plataforma, recomendo o livro *Tap, Move, Shake: Turning Your Game Ideas into iPhone & iPad Apps* (*http://shop.oreilly.com/product/0636920018414.do*) [em uma tradução livre, "Toque, movimente e balance: transformando suas ideias de jogos em aplicativos iPhone e iPad"], de Todd Moore (O'Reilly).

O ciclo de lançamento

Paul Graham, um de meus divulgadores de conhecimento favoritos, argumenta que a chegada dos softwares web mudou não apenas a experiência do usuário, mas também a experiência do desenvolvedor:

> Uma das mudanças mais importantes neste novo mundo é a forma como você realiza lançamentos. No mercado de software desktop, a realização de um lançamento é um enorme trauma, no qual toda a empresa sofre e se esforça para "empurrar para fora" um único e gigante fragmento de código. Comparações óbvias sugerem a si próprias, tanto com relação ao processo quanto com relação ao produto resultante.
>
> – De "The Other Road Ahead"
> (em uma tradução livre, "A outra estrada adiante"), de Paul Graham

Ele está exatamente certo. Trabalhando na nuvem, você raramente realiza um lançamento de software no velho sentido da palavra. Apesar dos benefícios, devo admitir que, na realidade, sinto saudades desse "empurrão final", no qual, geralmente com uma enorme ansiedade, você lança uma versão nova e melhorada de um fragmento de software. No entanto um problema com a escrita de aplicativos nativos é que fizemos um retorno ao ciclo de lançamento.

Com um software web você pode fazer lançamentos incrementais, corrigindo bugs quando eles ocorrem e se eles ocorrerem. Aplicativos nativos são muito mais parecidos com um software desktop.

Abordo os detalhes do envio de aplicativos para a App Store no capítulo 13. No entanto você deve se preparar agora para experimentar algum grau de dor. O processo de revisão é notoriamente obscuro, além de demorar algum tempo. Além disso, cada um de seus aplicativos deve passar por ele, e não apenas quando você enviar o aplicativo inicialmente para a loja, mas também para cada nova versão que você lançar. Geralmente, pode demorar até sete dias desde o envio de seu aplicativo até que ele seja aprovado (ou rejeitado) pela equipe de revisão, ainda que isso possa demorar muito mais. Apesar de alguns de meus aplicativos terem percorrido rapidamente o processo de envio em apenas alguns poucos dias, já tive aplicativos no processo de revisão por até quatro meses antes que recebessem aprovação.

Se você está interessado em saber como o período médio de revisão para aplicativos iOS e OS X muda com o passar do tempo, dê uma olhada no Average App Store Review Times (*http://reviewtimes.shinydevelopment.com/*). O site, criado por Dave Verwer, um reconhecido desenvolvedor iOS, rastreia o período médio de revisão da App Store para aplicativos enviados para a App Store tanto do iOS quanto do Mac, utilizando dados de crowdsourcing de desenvolvedores no Twitter.

Desenvolva e eles virão

Antes, neste capítulo, argumentei contra aplicativos web, pois eles requerem uma conexão de internet; evidentemente, essa não é a verdade completa. Você pode escrever um aplicativo web utilizando o Offline Application Cache disponível no HTML 5, ainda que haja limitações óbvias a essa abordagem. No entanto a grande vantagem de um aplicativo nativo, mesmo com a App Store superlotada de hoje, é a exposição. Se ninguém puder encontrar seu aplicativo, ninguém poderá pagar por ele, e a web é um lugar grande.

Uma vantagem significativa que um aplicativo nativo tem sobre um aplicativo web é a de que ele é mais fácil de ser encontrado por usuários potenciais e é muito mais fácil que paguem para obtê-lo. Isto é, se você conseguir fazer com que as pessoas paguem por aplicativos web. As pessoas não assinam por impulso um web service; elas compram por impulso na App Store.

No entanto não presuma que, se você desenvolver algo, os usuários aparecerão. A não ser que você seja realmente sortudo e que seu aplicativo se torne viral, você ainda terá de fazer o marketing dele. A App Store pode ser muito menor do que a web, mas ainda é um lugar bastante grande.

O marketing de seu aplicativo é como o marketing de qualquer outro produto: você precisará utilizar as ferramentas disponíveis e seus contatos para levar as novidades de seu software até seu mercado-alvo. A Apple fornece códigos promocionais para seu aplicativo (embora, quando este texto foi escrito, eles ainda funcionassem apenas na App Store norte-americana) que oferecem downloads gratuitos de seus aplicativos. Muitos desenvolvedores contatam blogs de alto perfil ou os inúmeros sites de catálogo de aplicativos e oferecem a eles cópias de revisão na esperança de que façam a publicidade do aplicativo. Se o aplicativo for bem projetado e útil, eles poderão muito provavelmente ficar interessados em revisá-lo.

Produza uma apresentação de vídeo mostrando como seu aplicativo funciona e como utilizá-lo. Dessa forma, também, aplicativos com bons recursos de suporte (como fóruns e sistemas de bug tracking) vendem mais cópias. Aplicativos com um bom design se destacam na loja e vendem mais cópias.

Um bom design frequentemente significa que você faz tudo do "jeitinho da Apple". Integre seu aplicativo adequadamente com os outros aplicativos do telefone. Não reinvente a roda – utilize os widgets e os elementos de interface padrão que são familiares a usuários do iOS.