1 Introduzione al web design per il mobile

Siete pronti per entrare nella grande arena del web design? Tutto questo libro riguarda il design per i dispositivi mobili, cioè il design del futuro. Vi guideremo nel processo di ideazione e realizzazione di un'applicazione web da zero. Daremo un'occhiata a quanto occorre considerare quando si progetta in un contesto mobile, costruendo la base della nostra applicazione utilizzando gli standard web e aggiungendo l'interattività. Infine attiveremo ed eseguiremo l'applicazione in un ambiente nativo, così che possa essere scaricata dai vari store online dedicati.

Il libro si concentra sulla creazione di app per device delle dimensioni di un telefono, sebbene molti dei concetti e delle tecniche possano essere applicati ad altri dispositivi e contesti mobili, come i tablet o i netbook.

Da un punto di vista tecnico, illustreremo tecnologie di creazione già note come HTML, CSS e JavaScript, che costituiranno la base di quasi tutto quello che tratteremo. Dovrete quindi avere una buona comprensione delle stesse per poter iniziare.

Cosa significa "mobile"?

Cominciamo con l'intenderci. Potreste chiedervi cosa si intende con *mobile*. La risposta non è univoca. A prima vista realizzare per il Web mobile potrebbe non sembrare così diverso che farlo per altre applicazioni o siti web: basta ottimizzare il tutto per la visualizzazione sui device mobili. I risvolti sono però più profondi e articolati.

Le discussioni sul Web mobile tendono a focalizzarsi sui dispositivi e le loro funzionalità, cose come l'ultimo iPhone, il recentissimo telefono Android o webOS. Poiché è un paesaggio in rapido cambiamento e quindi un momento eccitante per lo sviluppo web, è facile restare invischiati in diatribe sui requisiti tecnici e le soluzioni specifiche per i vari device. Questa prospettiva trascura però l'enorme opportunità offerta dal design mobile, che in ultima analisi ha a che fare con le persone, non con i dispositivi. La definizione data da Barbara Ballard nel suo libro *Designing the Mobile User Experience*, coglie nel segno (Hoboken, Wiley, 2007):

> Fondamentalmente, il termine *mobile* si riferisce all'utente, e non al dispositivo o all'applicazione.

Le persone, non le cose. La mobilità è qualcosa di più del liberarsi dai vincoli delle nostre scrivanie. È un contesto diverso, un'esperienza utente differente. Curiosamente, le persone usano le app mobili quando loro stesse sono "mobili", ed è questa natura "sempre e ovunque" del design mobile che rende tali applicazioni particolarmente utili, e quindi impegnative da progettare. Occorre pensare con attenzione ai destinatari e a quello che potrebbero volere o richiedere. L'obiettivo è creare applicazioni che brillino in questo senso. Per questa ragione in tutto il libro ci concentreremo sì sull'implementazione tecnica, ma tenendo sempre la definizione di Ballard come punto fermo dei processi decisionali.

Perché è importante?

Secondo alcune stime, nel 2013 il numero degli smartphone e di altri telefoni dotati di browser si aggirerà complessivamente attorno a 1,82 miliardi, a fronte degli 1,78 miliardi di PC (`http://www.gartner.com/it/page.jsp?id=1278413`). È

noto che non è facile trovare statistiche affidabili relative all'uso dei browser web; nondimeno, la tendenza è chiara. Secondo StatCounter, la quota "mobile" della navigazione complessiva del Web si aggira attualmente sul 4,36%, e per quanto questa cifra possa sembrare irrisoria, è bene ricordare che negli ultimi anni si è avuto un incremento eccezionale del 430%. E questo è solo l'inizio. In futuro non useremo certo di meno i telefoni e gli altri device mobili di quanto facciamo ora. Dispositivi mobili più potenti e un accesso più capillare alla Rete diventeranno la norma, e il contesto in cui questi device verranno utilizzati cambierà rapidamente. La probabilità che i nostri clienti potenziali sfruttino sempre più questi device è elevata. Se ignoriamo il fenomeno del Web mobile lo facciamo a nostro rischio e pericolo.

I nativi sono irrequieti

Quando si progetta per lo spazio mobile si deve affrontare una scelta inevitabile tra creare un'applicazione nativa o un'applicazione web. Vediamo alcune definizioni. A un'*applicazione web* si accede sul Web tramite il browser del device, navigando in un sito che offre una funzionalità simile a quella dell'app. Un'*applicazione nativa* viene invece creata per una piattaforma specifica (come Android o iOS, per esempio), e viene installata sul device in modo analogo a un'applicazione desktop. Queste applicazioni sono in genere messe a disposizione dei consumatori tramite un punto vendita specifico per la piattaforma. Tra i più famosi vi sono l'App Store di Apple per iPhone e iPad.

Diamo allora un'occhiata ai pro e contro delle due tipologie di applicazioni. Di norma, le app native offrono un'esperienza d'uso superiore, e tale differenza è ancora più marcata sui dispositivi più lenti. Le applicazioni native vengono costruite, ottimizzate e soprattutto compilate specificatamente per il device o la piattaforma su cui vengono eseguite. Su iOS, questo significa che sono scritte in Objective-C, e su Android in Java. Di contro, le web app sono *interpretate*, e cioè devono essere lette e capite immediatamente dai motori di rendering e JavaScript del browser. Per iOS, Android, BlackBerry, Symbian e webOS, il motore privilegiato è il progetto open source chiamato WebKit, lo stesso che alimenta Safari e Chrome. Per Windows Phone 7, il motore è attualmente una versione di Internet Explorer 7, sebbene Microsoft abbia annunciato di volerlo sostituire con il motore di rendering interno a Internet Explorer 9. Questo livello ulteriore tra il codice e il dispositivo fa sì che le web app non si eseguano mai come le app native, e questo può essere un problema quando si vuole realizzare un'app che richiede immagini in 3D ad alta risoluzione o che necessita di una grande potenza di calcolo. Tuttavia, se costruiamo qualcosa di più semplice, una web app farà il suo dovere con onore. Ci saranno sempre delle differenze a livello di prestazioni, ma saremo comunque in grado di garantire una buona esperienza agli utenti.

Il fatto che le applicazioni web debbano essere interpretate significa anche che si è vincolati ai limiti del motore. Laddove le app native possono accedere a tutto lo stack di metodi esposto dal sistema operativo, le applicazioni web possono comunicare con il sistema operativo solo tramite il motore del browser, e quindi si è limitati alle API rese disponibili dal browser. Per esempio, su iOS le applicazioni native possono accedere a un intero set di funzionalità che non sono disponibili attraverso Mobile Safari, come le notifiche push, la fotocamera, il microfono e i contatti dell'utente. Di conseguenza, non sarà possibile costruire un'applicazione web che consenta agli utenti di caricare foto su un servizio come Flickr o Facebook. Detto questo, il browser espone comunque un ventaglio di funzioni del device: l'API Geolocation permette di individuare la posizione dell'utente (se questi ci autorizza); l'API DeviceOrientation offre l'accesso al giroscopio e ai dati dell'accelerometro; con l'API Web Storage possiamo salvare i dati tra le sessioni di navigazione. Aggiungete audio e video in HTML5, gesti attraverso gli eventi *touch* del browser, transizioni e trasformazioni CSS e grafica 3D con WebGL, e il distacco nelle funzionalità di assottiglia. È tuttavia possibile che ci sia sempre qualcosa che non potremo usare (e di solito è la funzione più nuova ed eccezionale).

Ma allora, se le app native sono l'ideale, perché dovrebbe interessarci realizzare applicazioni web?

Il problema dell'essere nativi

Uno dei problemi principali della realizzazione di applicazioni native è la frammentazione.Visto che sono specifiche di una piattaforma, come si sceglie la piattaforma di destinazione? L'applicazione dovrà essere disponibile prima nell'App Store o sull'Android Market? E cosa dire del BlackBerry o di Windows Phone 7? Ricordate che per ogni piattaforma che desiderate supportare dovrete riscrivere l'applicazione. In un mondo ideale, realizzeremmo un'applicazione nativa per tutte queste piattaforme e altre ancora, ma nel mondo reale le nostre risorse sono limitate, per cui siamo costretti a scegliere quali piattaforme – o meglio, quali utenti – tralasciare. Nel caso di un'applicazione web, tuttavia, finché questi device si adatteranno a un browser web, potremo costruire un'unica applicazione partendo da una stessa base di codice che serve tutte queste piattaforme e altre. L'aspetto della frammentazione riguarda i browser, e quindi le web app, ma non è una novità per i designer del Web, e le differenze sono solitamente irrilevanti.

Un altro problema è quello della velocità di sviluppo. In quanto professionisti del Web, abbiamo una notevole esperienza nella costruzione e gestione delle applicazioni web. Conoscere un nuovo insieme di strumenti di sviluppo o affidarsi a qualcuno con queste competenze richiede tempo, fatica e denaro. Ci occorre quindi una ragione per giustificare tali seccature e spese che non sia una semplice sfida sulle competenze che già possediamo. Per contro, tali ragioni si fondano su

quanto è meglio per la nostra attività, e non per i nostri utenti (una buona motivazione). È quindi una questione di equilibrio: barattiamo cioè l'esperienza degli utenti con la familiarità, la velocità di sviluppo e la flessibilità della piattaforma. Ovviamente dovremo creare la migliore applicazione possibile a prescindere dalla piattaforma preferita dai nostri utenti, ma un'app che viene distribuita offre un'esperienza decisamente superiore di una che non vedrà mai la luce del giorno.

In tempi recenti alcune compagnie di alto profilo hanno soppesato questa equazione e convogliato i loro sforzi nel Web. 37signals, fornitore di numerose applicazioni di produttività basate sul Web, comprese Basecamp e Highrise, ha evitato la strada dell'app nativa quando ha rilasciato Basecamp mobile:

> Alla fine siamo giunti alla conclusione che dobbiamo limitarci a quello che sappiamo fare meglio: le web app. Conosciamo bene le tecnologie, abbiamo un ambiente di sviluppo e un flusso di lavoro eccellenti, possiamo controllare il ciclo di rilascio e tutti in 37signals possono eseguire il lavoro […] Lavoriamo in HTML/CSS/JS ogni giorno, e l'abbiamo fatto per anni. I guadagni che facciamo con il desktop possono ripercuotersi nel mobile e viceversa. Per noi è il cerchio perfetto.
> Jason Fried su *Signal vs. Noise*, 1 febbraio 2001
> `http://37signals.com/svn/posts/2761-launch-basecamp-mobile`

Per il team di 37signals, il problema non è l'impiego di risorse e denaro. Semplicemente hanno deciso che realizzare un'applicazione web fornisce un'esperienza migliore a più utenti, e farlo ricorrendo a tecnologie con cui hanno familiarità dà loro un maggiore controllo sull'applicazione nella sua globalità. Netflix è pervenuta a una conclusione analoga. La sua applicazione per la PlayStation 3 è scritta per intero con tecnologie web, e permette ai suoi sviluppatori di testarla e iterarla in continuazione, così da raggiungere i risultati migliori per gli utenti:

> La nostra missione principale è quella di sperimentare incessantemente le tecnologie che poi rilasciamo agli utenti. Testiamo ogni idea, così da poter misurare l'impatto sui nostri clienti […] È qui che entra in gioco HTML5. La tecnologia viene distribuita dai server Netflix ogni volta che lanciate la nostra applicazione. Questo significa che possiamo costantemente aggiornare, verificare e migliorare l'esperienza che offriamo. Abbiamo già effettuato diversi esperimenti sulla PS3, per esempio, e anche adesso stiamo lavorando intensamente su altro. I nostri clienti non devono eseguire una procedura manuale per installare il nuovo software ogni volta che apportiamo una modifica; semplicemente "succede".
> John Ciancutti, *The Netflix "Tech" Blog*, 3 dicembre 2010
> `http://techblog.netflix.com/2010/12/why-we-choose-html5-for-user.html`

Anche Facebook, una società con risorse di engineering tutt'altro che irrisorie (avendo prodotto l'applicazione numero uno per iPhone di tutti i tempi), ha delle difficoltà a gestire la frammentazione delle piattaforme, e sta adottando gli standard web come futuro della propria strategia "mobile" (`http://networkeffect.allthingsd.com/20110125/facebook-sets-mobile-sights-on-html5/`).

Le app web mobili offrono numerosi vantaggi rispetto a quelle native, e per quanto debbano affrontare alcune sfide di design, sviluppo e distribuzione, sono una potente soluzione multipiattaforma, accessibile e scalabile.

NOTA

Le API abilitano. Nonostante la decisione di 37signals di non dedicarsi internamente allo sviluppo di app native, nell'App Store attualmente ci sono non meno di dieci client nativi per la sua applicazione web Basecamp. L'API fornita da 37signals ha permesso che sviluppatori di terze parti costruissero le loro applicazioni native partendo da Basecamp, consentendo comunque alla compagnia di controllare il livello di interazione con i dati degli utenti. Un'API ben costruita fa sì che gli utenti possano realizzare delle app per voi, qualcosa che probabilmente non vi sareste mai aspettati.

Partire dall'inizio

"Dobbiamo avere un'app per iPhone". Sì, può essere, ma un'applicazione nativa per le diverse piattaforme non è sempre la soluzione migliore. La decisione deve basarsi su qualcosa di più di "ce l'hanno tutti". Occorre valutare se realizzare un'applicazione mobile sia la decisione più giusta per noi e per i nostri utenti (a prescindere dalla tecnologia). Se possedete una catena di caffetterie con 1000 punti vendita in tutto il mondo, un'applicazione mobile che aiuti i vostri clienti a trovare il negozio più vicino potrebbe essere un'ottima idea. Se però rappresentate l'emporio "vendi tutto" del quartiere, forse un'alternativa più semplice è più adatta.

Alle persone serve ciò che offriamo? Perché dovrebbero usare la nostra applicazione mentre sono "mobili"? Dove la useranno? Che esiti avrà sulla nostra attività?

Il modo migliore per rispondere a queste domande è quello di analizzare le informazioni di cui già disponete. Esaminate le statistiche correnti del vostro sito web: quanti visitatori lo visualizzano sui loro device mobili? Quali dispositivi usano? Che contenuti guardano? Questi dati possono fornire un'indicazione su ciò che le persone cercano in un contesto mobile. Ovviamente tali dati saranno influenzati dai vincoli del vostro sito web attuale, quindi qualsiasi conclusione estrapolerete costituirà solo una parte del processo decisionale.

E se non avete statistiche sui dati? Potreste provare a parlare con gli utenti: non c'è alcun pericolo nel chiedere alle persone cosa desiderano. Di solito la risposta è semplice: vogliono quello che già vendete il più in fretta possibile. Se siete dei fioristi, vorranno dei fiori, adesso. Avete un bar? Vorranno un caffè, subito. A prescindere dal prodotto o dal servizio, riuscire a realizzare un'applicazione che soddisfi queste richieste sarà enormemente gratificante per gli utenti (e produrrà denaro).

NOTA

L'effetto App Store. Il successo dell'App Store non va sottovalutato: c'è un vantaggio innegabile nel posizionare un'app in uno spazio così popolare, e avere la propria icona al centro dello schermo principale di un utente garantisce all'app un'esposizione che un segnalibro non dà. Anche il ritorno economico è molto chiaro: gli store online delle applicazioni portano clienti, e i clienti portano denaro. Il fatto di realizzare un'applicazione web mobile non implica che si debba trascurare le potenziali entrate economiche del prodotto. È qui che entra in gioco un'app ibrida, a metà tra una web e una nativa. Ma non anticipiamo i tempi; parleremo di questo e altro nel Capitolo 7.

Un'app non basta

Una delle ragioni principali per creare un'applicazione mobile usando le tecnologie web è che prima o poi verrà comunque il momento di farlo. Giustamente, gli utenti si aspettano che il sito web funzioni già sui loro device mobili. Il principio su cui ci si deve basare è che non si deve dare nulla per scontato riguardo alle potenzialità del dispositivo dell'utente, perché sbaglieremmo. Un'app nativa non è la soluzione al problema.

Abbiamo già visto che il design mobile ha a che fare con il contesto, ma dipende anche dalla velocità. Lo scopo è fornire agli utenti ciò che vogliono il più rapidamente possibile. Un'applicazione nativa, per quanto fantastica, è una buona idea solo se gli utenti ce l'hanno già installata. Chiedere agli utenti di scaricare da un punto vendita specifico un'applicazione separata può diventare un problema se questi sono in giro e devono affidarsi alla copertura di rete. Fornire una versione del nostro sito agli utenti mobili diventa quindi importante a prescindere dal fatto che questi abbiano a disposizione un'applicazione nativa. Cosa fare, dunque?

Opzione uno: niente

Non fare niente è un'opzione seria, da non accantonare. Le nuove generazioni di smartphone facilitano molto la navigazione nelle pagine grandi e complesse. Per esempio, la home page del *New York Times* contiene un'enorme quantità di informazioni sugli argomenti più disparati. Dietro le quinte, però, questo volume di dati ha un prezzo: il download della prima pagina di `http://nytimes.com/` occupa almeno 1 MB di dati, come rivela il Web Inspector di Chrome (Figura 1.1).

È vero, oggigiorno la copertura 3G è piuttosto buona, ma così chiediamo comunque ai nostri utenti di attendere quattro o cinque secondi prima di poter interagire con il nostro sito, e questo non è molto mobile né *user friendly*. Diventa quindi imperativo costruire un sito usando un markup significativo e ben strutturato. Quando si passa al mobile, l'adesione alle best practices convenzionali del web design dà ancora più frutti. Layout leggeri basati sui CSS, un design con il contenuto esposto e un focus sull'accessibilità e la usability contano ancora di più quando le dimensioni dello schermo, l'attenzione e la larghezza di banda sono ridotte.

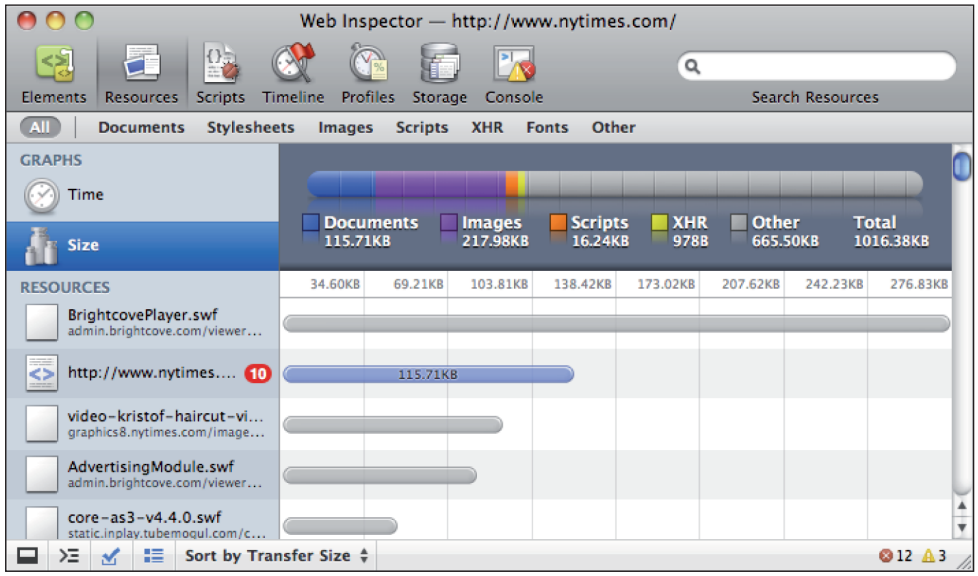

Figura 1.1 Il Web Inspector di Chrome rivela il costo effettivo delle pagine più complesse.

Opzione due: trasformare e andarsene

Se vi siete persi l'articolo fondamentale di Ethan Marcotte sul design reattivo, recuperatelo (`http://www.alistapart.com/articles/responsive-web-design/`).

L'espressione *web design reattivo* si riferisce all'arte dell'uso delle media query CSS, di immagini e layout a griglia fluidi che rispondono alla risoluzione del dispositivo (o della finestra del browser) dell'utente, adattando il design per fornire il formato migliore per una data risoluzione.

È una tecnica molto semplice, anche se a volte richiede molta applicazione, e vale la pena studiarla. Le *media query* sono un'estensione degli attributi dei tipi di media noti che sono stati utilizzati per anni per visualizzare i fogli di stile nelle pagine HTML. La differenza è che, invece di ammettere un unico contesto per le regole CSS, è possibile interrogare (*to query*) le caratteristiche fisiche del device dell'utente e poi distribuire diversi fogli di stile (o parti di essi) solo a quei dispositivi che rispondono ai criteri specificati. Nell'esempio che segue diciamo "Carica il file `mobile.css` solo se la finestra è larga almeno 480 pixel":

```
<link rel="stylesheet" type="text/css" media="screen and (max-width: 480px)"
➥href="mobile.css" />
```

Ovviamente le interrogazioni non riguardano esclusivamente la larghezza del device. Tra le numerose funzioni supportate nelle specifiche delle media query vi sono:

- larghezza e altezza (come nell'esempio precedente);
- risoluzione dello schermo;
- orientamento;
- proporzioni.

Tuttavia, la vera potenzialità sta nella capacità di combinare queste query in regole più complesse. Volete fornire determinati stili ad alta risoluzione con un orientamento orizzontale? Nessun problema:

```
<link rel="stylesheet" type="text/css" media="screen and
➥(min-resolution: 300dpi) and (orientation: landscape)"
➥  href="mobile.css" />
```

Questo approccio può essere utilizzato per creare un sito mobile eccezionale, un sito desktop eccellente e tutto quanto sta tra questi due estremi partendo dallo stesso codice di base.

Purtroppo la metodologia del design reattivo ha i suoi limiti. Riformattare il layout può rendere il nostro contenuto più gradevole su un dispositivo mobile, ma rimane qualcosa di "decorativo", e si rischia di costringere gli utenti a scaricare risorse che potrebbero non riuscire a vedere mai. Fortunatamente alcuni di questi problemi possono essere attenuati con un'attenta pianificazione.

Come prima cosa, costruite innanzitutto per il mobile. Sebbene la tendenza sia quella di usare le media query come mezzo per aggiungere un tocco in più a un sito desktop pressoché completo, si dovrebbe fare esattamente il contrario. Per raggiungere un'esperienza desktop più potente, partite da una base semplice e sovrapponete la complessità:

```
<link rel="stylesheet"
➥ media="screen and (min-width: 939px)" href="desktop.css" />
```

Il concetto è quello del miglioramento progressivo. Non si può evitare di inviare lo stesso HTML a tutti i device; tuttavia, se badiamo al modo in cui costruiamo e distribuiamo i fogli di stile, potremo garantire un'esperienza ottimale agli utenti mobili, fornendo direttamente loro il contenuto che è probabile che cercheranno. In questo modo riduciamo al minimo i costi senza atrofizzare l'esperienza desktop.

Opzione tre: da soli per sempre

Un'opzione più comune è quella di costruire un sito web del tutto separato per gli utenti mobili. Un sito del genere si trova solitamente su un sottodominio `m.` o `mobile.` (o su entrambi). Con questo metodo si può modellare un'esperienza focalizzata esclusivamente sulle esigenze degli utenti mobili. L'ideale. Più o meno. Ci sono infatti degli inconvenienti a questo approccio. Avere un sito mobile distinto significa solitamente doversi affidare a una qualche forma di individuazione dello

user agent per decidere quale dispositivo riceve una data versione del sito, per esempio "distribuisci il sito mobile agli utenti di iPhone, distribuisci la versione desktop agli utenti di Firefox" e così via. Apparentemente valida, questa sorta di *sniffing* dello user agent (cioè l'individuazione del browser dell'utente in base alle informazioni che questo dà di sé nelle richieste al nostro server) è notoriamente inaffidabile. Lo user agent può essere cambiato facilmente in molti browser, e spesso per aggirare specificatamente proprio questa tecnica. Anche qualora riuscissimo a far sì che il nostro sito mobile rilevi correttamente il giusto gruppo di browser mobili, potremmo creare dei problemi imprevisti agli utenti dei device che ancora non conosciamo.

La soluzione è semplice: consentire agli utenti di seguire un percorso autonomo inserendo un link al nostro sito standard nella versione mobile (e viceversa) e rispettare questa decisione. Non c'è nulla di sbagliato a incoraggiare gli utenti a partire dal sito mobile, ma non dovremmo comunque impedire loro di arrivare a tutto ciò a cui accedono normalmente.

Facebook è un buon esempio di comportamento corretto nel consentire agli utenti di spostarsi tra il sito standard e quello mobile. Facebook ha due siti mobili: touch.facebook.com, per gli utenti mobili degli smartphone touch, e m.facebook.com, per gli utenti dei device non touch. Entrambi i siti consentono di eseguire le normali operazioni e interazioni che ci si aspetta da Facebook: leggere e rispondere ai messaggi, postare aggiornamenti sullo stato e vedere la bacheca che è al cuore del sito. Tuttavia non possiamo fare tutto quello che il sito standard permette, per esempio caricare fotografie o modificare il nostro profilo. Se dobbiamo assolutamente eseguire un'operazione che è ammessa solo sul sito standard (o che viene effettuata meglio su questo), i due siti mobili di Facebook forniscono un comodo link nel piè di pagina per spostarsi tra le versioni. La chiave è quella di concedere sempre agli utenti un accesso semplice al sito completo, senza escluderli da alcuna funzionalità. Separate, non segregate.

Una nota sui framework

Quando si fanno ricerche sulla costruzione delle applicazioni web per i dispositivi mobili ci si imbatte spesso in progetti che hanno l'intento di fornire framework multipiattaforma per il mobile. I principali sono Sencha Touch (`http://www.sencha.com/products/touch/`) e JQuery Mobile (`http://jquerymobile.com/`). Non ci addentreremo nelle specifiche della loro implementazione, ma vale comunque la pena dar loro un'occhiata per decidere se possono o meno adattarsi ai nostri scopi.

Entrambi sono fondamentalmente framework JavaScript. Le applicazioni costruite con Sencha Touch dipendono per intero dal fatto che i device degli utenti abbiano un buon motore JavaScript, cosa che sappiamo non succedere sempre. Se

si visualizza un'applicazione costruita in Sencha Touch senza JavaScript, si ottiene una shell vuota che non fa nulla. JQuery Mobile, invece, sceglie l'approccio più user friendly del miglioramento progressivo; le sue applicazioni sono costruite in HTML semplice, e il loro comportamento è impostato sul livello superiore. Questo è un altro compromesso (cosa non rara nel mondo del mobile!). Sencha Touch utilizza il metodo JavaScript perché garantisce prestazioni migliori, in virtù del fatto che la logica dell'applicazione è costruita in JavaScript e non sul DOM. A prescindere dagli approcci, la cosa da ricordare su questi framework è che entrambi implementano un intero set di funzioni che replicano il comportamento e l'operatività delle applicazioni native. Se non le utilizzate tutte, quindi, pagherete dei costi inutili.

Questo ci porta a uno dei primi principi da considerare quando si progetta una web app: la *uncanny valley* (la zona perturbante). Secondo questa teoria della robotica, più un robot antropomorfo cresce nella sua somiglianza con la figura umana, più il suo aspetto causa repulsione negli essere umani. L'idea può essere applicata anche alle interfacce: se assomiglia a un'anatra, fa lo stesso verso di un'anatra ma non è un'anatra, come verrà vista dagli utenti? Replicando il *look and feel* di un'applicazione nativa, i framework di un'applicazione mobile creano certe aspettative, che, per definizione, sono impossibili da soddisfare. La soluzione è semplice: basta accettare i limiti. Non occorre fingere di essere fuori dall'ambito della normale interfaccia del browser. Il mobile non è una maledizione, bensì un'opportunità di prendere una decisione attiva sul modo di presentarsi. La versione mobile di Twitter non cerca di emulare le applicazioni native del social network, ma fa quello che ci si aspetta che faccia: fornire più informazioni possibili nel modo più rapido possibile.

Rimbocchiamoci le maniche

Tutta questa discussione è interessante, ma questo non dovrebbe essere un libro sulla costruzione di applicazioni web mobili? Sì, lo è, e per quanto sia importante capire perché si scelga una data strategia, è ancora più divertente creare qualcosa di concreto. Cosa realizzeremo? Il caso vuole che siamo stati avvicinati da un potenziale cliente che ci ha chiesto di creare una versione mobile del famoso sito web StarTrackr. Si tratta di un sito "cerca VIP" che permette agli utenti di registrare quando e dove incappano in una celebrità, un esempio perfetto di un'operazione da Web mobile. Ripassiamo le opzioni a nostra disposizione: possiamo non fare niente (ma il cliente non apprezzerebbe), possiamo modellare un design reattivo al mobile o possiamo creare una versione mobile separata (ma correlata) del sito. Dipende tutto da quello che vogliamo ottenere (o meglio, che il cliente vuole ottenere).

Per StarTrackr, il cliente vuole che l'utente sia in grado di:

- vedere i punti (spot) più vicini (i luoghi dove è stata vista una celebrità) e i vari avvistamenti in quei punti;
- trovare le sue celebrità preferite e vedere dove sono state avvistate di recente;
- aggiungere un nuovo avvistamento.

La natura di queste attività ci dice chiaramente che stiamo parlando di costruire un'applicazione web, non un semplice sito. In parole povere, le applicazioni web servono a eseguire delle attività, i siti web a consumare delle informazioni. Capire questa distinzione e il modo in cui le tecniche illustrate si adattano a un'opzione o all'altra è cruciale. Possiamo rendere le finestre più belle, ma se il nostro intento è quello di creare un'esperienza avvincente e contestuale per i nostri utenti, dovremo passare a un'applicazione mobile separata. E allora andiamo.