Introduzione

"Come creare un'app" è sicuramente una delle frasi più ricorrenti digitate nei motori di ricerca. Inutile soffermarsi sul perché sia un argomento così richiesto: se vi trovate a sfogliare questo libro vi sarete già dati sicuramente una risposta e presumo non sia poi così distante dal perché ho voluto procedere a scrivere questo libro. Quello della domanda e dell'offerta è un modello che ha sempre regolamentato qualsiasi mercato; oggi è in costante crescita quello tecnologico. Il mondo del lavoro nel campo informatico non è mai stato così fiorente e le competenze maggiormente ricercate sono quelle legate allo sviluppo: ancor di più, sviluppo web e mobile. Lo scopo ultimo di questo libro è quello di accompagnare il lettore passo dopo passo nell'apprendimento di uno strumento specifico, che nasce per facilitare la creazione di applicazioni web e mobile. Tutto questo è stato pensato e realizzato tenendo conto del fatto che la finalità è quella di realizzare qualcosa di reale, tangibile: una vera applicazione.

A chi si rivolge questo libro

Iniziare a leggere un libro del genere senza possedere approfondite conoscenze informatiche è sicuramente una strada in salita. Diverso è per chi ha già dimestichezza con lo sviluppo di applicazioni web e vuole approfondire l'uso di un framework, Angular in questo caso, di cui magari ha già apprezzato le potenzialità o, più semplicemente, ha già sentito parlare. Chi ha già nozioni di programmazione è sicuramente favorito rispetto a chi si affaccia per la prima volta sul mondo dello sviluppo. Quanto a un neofita immagino due importanti motivazioni che lo possono spingere verso una lettura di questo tipo: la passione per l'informatica, lo sviluppo in particolare, oppure la ricerca di una specifica competenza, da "spendere" nel mondo del lavoro. In entrambi i casi si è mossi da una grande forza e voglia di fare; se le due motivazioni coincidono… tanto meglio! Questo volume è stato pensato per ambedue le categorie: lo sviluppatore e il neofita. Il primo approfondirà le ultime tecnologie, combattendo con il dubbio atavico instillato dal nuovo che avanza; il secondo apprenderà le tecnologie più moderne. In ogni caso, questa lettura sarà assolutamente un valore aggiunto al bagaglio tecnologico di ognuno. Rispetto a una qualsiasi lettura accademica, questa si presenterà come la più pragmatica possibile. Sarà privilegiato l'aspetto tecnico a quello teorico: procederemo passo dopo passo, approfondendo i vari aspetti che ruotano attorno all'argomento trattato, accom-

pagnandovi nella comprensione di uno strumento che vi consentirà di raggiungere un prestigioso traguardo: sviluppare la vostra prima applicazione web e/o mobile. Quello che faremo insieme è un viaggio specifico, lineare, il più possibile chiaro. Utilizzeremo una serie di strumenti innovativi e creeremo insieme un'applicazione funzionale e completa, con le caratteristiche che la tecnologia e il mercato attuale richiede, accumulando, nel frattempo, specifiche competenze. Questo è ciò che faremo.

Scenario

Comprendere lo scenario che ci circonda è il primo passo. Il panorama legato al mondo della programmazione è composto da innumerevoli software, che consentono di sviluppare applicazioni partendo da strutture già consolidate, agevolando di molto il lavoro dello sviluppatore, da ogni punto di vista. I software di cui stiamo parlando vengono chiamati *framework*. La scelta di utilizzare un framework abbatte drasticamente i tempi di realizzazione e, in generale, la quantità di codice scritto. Ancora più importante: alcuni framework offrono un valido aiuto nell'approccio logico all'applicazione stessa.
Nonostante la nostalgia che molti sviluppatori, me compreso, provano per i tempi in cui sforare l'allocazione di un byte provocava l'arrivo dell'Apocalisse, nella maggior parte dei casi fare a meno di un framework risulterebbe una scelta errata.
La realtà è che la tecnologia web e mobile vive in un ambiente così eterogeneo che la scelta di sviluppare da zero un applicativo che tenga conto di tutte le differenti implementazioni da parte dei vari browser e dispositivi mobili risulterebbe davvero troppo onerosa. Questa problematica è espressa perfettamente dal termine *cross-platform*, che senza alcun dubbio rappresenta la base da cui partire quando si lavora con il Web e, ancora oggi, rimane uno degli aspetti più macchinosi dello sviluppo.

Il buon vecchio jQuery

Entriamo nel dettaglio e vediamo che cos'è un framework. Uno dei maggiori esempi legati allo sviluppo del Web in relazione alla nascita di un framework è sicuramente jQuery.

Figura I.1 Il logo del progetto jQuery.

jQuery è, ma soprattutto è stato, uno dei framework più usati, che hanno caratterizzato lo sviluppo web degli ultimi dieci anni. Come ogni framework, anche jQuery garantisce un accesso più semplice allo sviluppo web e, in particolare, alla manipolazione degli elementi che compongono la pagina. Grazie al suo utilizzo, si passò dall'accedere direttamente alle risorse del browser per manipolare un determinato elemento, per esempio per eliminare un input box, al delegare jQuery, con una notevole semplificazione del lavoro.

La complessità del fatto di utilizzare per la manipolazione della pagina l'interfaccia nativa fornita dal browser è stata, in pratica, soppiantata dall'utilizzo di jQuery, grazie alla sua semplicità di utilizzo. Le capacità tecniche legate alla conoscenza del funzionamento del browser vengono così dirottate verso la conoscenza specifica del framework. Quello che dovrebbe fare un buon framework è *astrarre*. Se potesse parlare, direbbe: "Tu preoccupati di studiarmi, che al resto penso io". È innegabile che sia molto più semplice studiare un framework, piuttosto che comprendere nel dettaglio il lato nascosto di una pagina web. Il passo tecnologico successivo è stato imposto dalla naturale espansione del Web e dalla conseguente necessità di incrementare le possibilità di interazione con l'utente. L'obiettivo principale è sempre quello di migliorare costantemente la cosiddetta "user experience", l'esperienza d'uso dell'utente. Le applicazioni web sono diventate sempre più complesse e ricordano sempre di più delle vere e proprie applicazioni desktop. Lo sviluppo del Web in questi anni ha fatto sì che anche la struttura di base si evolvesse, dall'uso di jQuery per semplici script si è passati ai modelli progettuali, denominati "design pattern".

Il nostro framework

Il primo framework di nuova generazione ad aver catturato l'interesse dell'intera comunità di sviluppo software è stato proprio Angular. Il nostro framework implementa un design pattern che impone un'architettura dalla struttura precisa, semplifica molto la scrittura del codice e fornisce una serie di librerie per implementare le più comuni interazioni tra applicazione e utente. Insieme ad Angular sono nati parecchi framework, ognuno con svariate peculiarità e specifici punti di forza, ma Angular è quello che racchiude in sé di più il concetto di framework, offrendo sia un design architetturale ben definito, sia una completa suite di funzionalità per la user-interface. In altre parole Angular aiuta molto lo sviluppatore, fornendo un approccio sia logico, nella creazione dell'applicativo obbligato dal suo design, sia pratico, nel disegnare l'interfaccia web. Il panorama framework legato al mondo del Web è attualmente diviso tra due principali framework: Angular, ideato in casa Google, e React, creato da Facebook. La mia personalissima opinione (e non solo mia, a onor del vero) è che il primo strumento sia più completo e, nonostante la curva d'apprendimento sia ripida, una volta compresa la sua logica, l'uso del framework vi garantirà l'immediata capacità di sviluppare i primi applicativi. Il secondo, React, non è propriamente un framework, ma piuttosto una libreria nata per creare interfacce web e, a differenza di Angular, non impone allo sviluppatore vincoli architetturali. Paradossalmente, nonostante React possa risultare al primo impatto più immediato nell'uso, senza una buona esperienza nello sviluppo di un progetto di medie dimensioni, il suo utilizzo potrebbe rivelarsi una lama a doppio taglio. Diversamente, Angular facilita senza alcun dubbio l'avvio del progetto, organizzando meglio la sua struttura. Più procederete nella lettura di queste pagine, meglio vi risulterà chiaro il senso di questa affermazione.

Premessa

Prima di buttarci a capofitto sugli aspetti pratici, i primi capitoli cercheranno di aiutarvi il più possibile nell'apprendimento di tutto ciò che è propedeutico alla scrittura del codice. Predisporremo il sistema operativo, installando tutto ciò che serve per iniziare a sviluppare e solo dopo arriverà la parte più divertente e creativa: l'uso di Angular.

Le basi della programmazione che sono necessarie alla realizzazione di un'applicazione web o mobile sono sicuramente la conoscenza di HTML5, CSS3 e JavaScript. TypeScript, che non deve spaventare, verrà introdotto poiché è necessario all'apprendimento di Angular, che armonizza il tutto. Riuscire ad affrontare tutti gli aspetti tecnici senza dilungarci in più volumi è un'impresa piuttosto ardua. Ciò che faremo sarà approfondire tutti gli aspetti del caso, passo dopo passo, sempre in relazione al codice che stiamo scrivendo. L'idea è quella di affrontare l'argomento fornendo tutte le informazioni e le nozioni necessarie, affinché possiate approfondire, in autonomia e contestualmente alla lettura, ciò che ancora non conoscete. Tutte le informazioni che sono facilmente reperibili tramite le pagine man, le documentazioni ufficiali (e non), là dove possibile verranno tralasciate, dando maggiormente spazio a un metodo tecnico quanto logico.

Angular è la scelta giusta

Angular nasce nel 2010: sviluppato in origine da Google, è un progetto open source supportato da un'enorme community, che partecipa attivamente al suo sviluppo. Angular si è presentato al grande pubblico come un framework JavaScript. Essendo JavaScript un linguaggio *client-side*, il framework sarà eseguito esclusivamente dal client. L'evoluzione del Web ha fatto sì che negli ultimi anni il carico di elaborazione delle informazioni si dirottasse dal server al client. Per intenderci, mentre prima molte operazioni venivano eseguite dall'altro capo del nostro dispositivo, mentre oggi si tende a fare il contrario. Questo perché la necessità di essere più veloci nello scambio dati con dispositivi di qualsiasi genere che si interfacciano a Internet è diventata vitale, e inoltre la potenza di calcolo raggiunta permette di elaborare i dati direttamente a bordo del dispositivo, limitando sempre di più gli scambi di dati tra client e server.

App single page

Angular è usato per la creazione di applicazioni single-page. SPA, acronimo di *Single Page Application*, descrive le applicazioni che sono fruibili in una unica "pagina", senza dovere necessariamente ricaricare tutti i contenuti al cambio di link. Le applicazioni single-page forniscono un'esperienza simile alle applicazioni desktop e questo è sicuramente un vantaggio in termini di efficienza. Questa tipologia di applicazione, come detto, lavora sulla stessa *view* e il browser non dovrà rielaborare ogni volta tutte le informazioni affinché esse vengano riprodotte a video. Le risorse vengono caricate dinamicamente, e dalla view principale ci si potrà muovere all'interno dell'applicazione tramite un sistema di navigazione prestabilito.

NOTA

Il termine view indica semplicemente l'interfaccia, la pagina web.

Dietro una view è presente una struttura logica composta da dati e codice, che definisce i comportamenti delle viste. L'organizzazione delle view in relazione al codice JavaScript a sostegno della parte logica è definita in un design pattern ben preciso, che Angular impone di seguire.

Data binding

La peculiarità principale che ha attratto fin da subito gli utilizzatori del framework è il "two-way data binding": in pratica Angular collega nativamente la view al codice. Questo "collegamento" permette di tenere costantemente sincronizzate entrambe le parti. Immaginiamo di aver definito un archivio e di aver creato una tabella a video con la lista degli ultimi venti fascicoli creati. Nella parte logica, che definiremo meglio successivamente, avremo un dato che corrisponde ai fascicoli. Il data binding non è altro che una sincronizzazione costante tra i dati e l'interfaccia. In questo caso tra i fascicoli e la specifica view che li mostrerà a video. Il *two-way data binding*, ancor di più, si propaga in entrambe le direzioni. Per intenderci, se volessimo eliminare un fascicolo direttamente dalla view, tramite una precisa interazione (come un clic sul pulsante predisposto), il fascicolo scomparirà dalla view e verrebbe eliminato anche dai dati. E viceversa: se eliminassimo il fascicolo dai dati, tramite il codice, questo verrà a sua volta eliminato anche dalla view.

Il nuovo Angular

Il two-way data binding si è rivelato nel tempo una scelta non poi così vincente, almeno nella prima versione. Il suo uso in applicazioni con una grossa mole di dati e diverse interazioni web, causava un crollo delle prestazioni, ancor di più in dispositivi mobili. La nuova versione di Angular (a oggi Angular 2.x) gestisce il binding in maniera più intelligente. La nuova logica è il data binding *one-way*, mentre il data binding two-way non è più una peculiarità imposta, ma una scelta opzionale. Il data binding one-way, monodirezionale, consente allo sviluppatore di gestire il collegamento con maggiore autonomia poiché, come è facile intuire, non sempre è necessario tenere sincronizzati dati e view in entrambi i versi, ma piuttosto in una direzione precisa, utilizzando le risorse necessarie solo quando è effettivamente necessario l'uso del data binding two-way (bidirezionale).

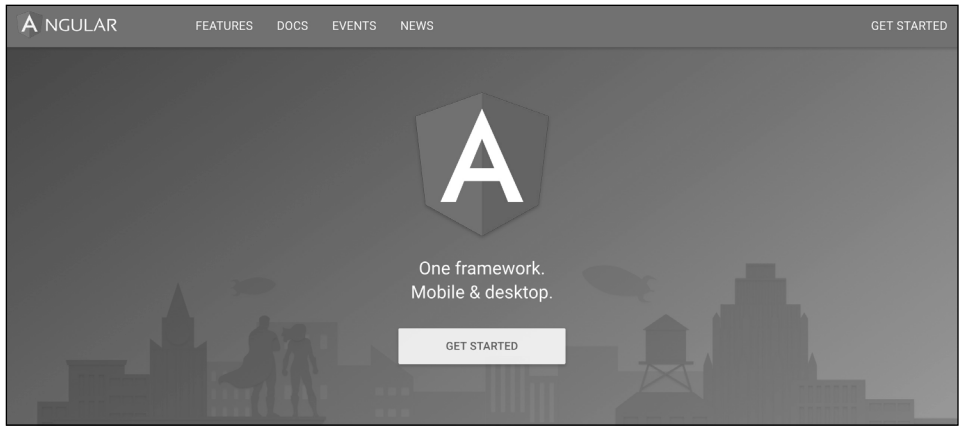

Figura I.2 Home page del nuovo progetto Angular (https://angular.io/).

L'evoluzione di Angular alla versione 2.x è stata un'occasione per ristrutturare totalmente il framework, che si basava sul vecchio standard EcmaScript. Gli sviluppatori hanno riscritto completamente il cuore del framework, rendendo Angular completamente modulare. Inoltre hanno lasciato, al momento, la possibilità di utilizzare tre linguaggi differenti: Java-

Script, TypeScript e Dart. Questo perché, nel corso della sua rifattorizzazione, sono state seguite più strade, percorrendo in conclusione quella che ha portato a TypeScript come linguaggio di sviluppo del framework. Questa motivazione spinge noi stessi a utilizzare TypeScript, a prescindere dalla documentazione ufficiale, carente negli altri linguaggi.

Rifattorizzazione

Processo di riscrittura migliorativa delle fondamenta di un software.

Un altro aspetto che lo legava al passato era l'uso interno di jqLite, una sotto-versione di jQuery, abbandonato in favore delle chiamate native per l'accesso al DOM, per incrementare ancora di più la velocità di rendering.

DOM

Struttura interna di una pagina HTML. L'accesso al DOM consente di manipolare gli elementi presenti nella pagina.

Ancora oggi il team di Angular sta sviluppando nuove funzionalità che verranno integrate dalla versione 4. Igor Minar, il team leader di Angular, ha anticipato che la futura release 4.x sarà concepita per riallineare il versioning che si era biforcato tra la versione 2.x e quella che avrebbe dovuto essere il seguito naturale, la versione 3.x. Questo disallineamento, dovuto a scelte interne probabilmente poco attente, si è rivelato una strada sbagliata. La versione 4 avrà quindi il compito di riportare compattezza nello sviluppo del progetto. In ogni caso non c'è da aspettarsi una rifattorizzazione com'è avvenuta dalla versione 1 alla versione 2, per adeguare Angular allo sviluppo di applicazioni sempre più complesse ed efficienti.

NOTA

Nell'evoluzione dalla versione 1.x alla 2.x, Angular ha anche cambiato nome e sito di riferimento. La versione 1.x è infatti denominata AngularJS e si può trovare all'indirizzo `https://angularjs.org`. La versione 2.x ha quindi assunto il nome attuale, Angular, e l'indirizzo di riferimento è `https://angular.io`. In questo libro facciamo riferimento ad Angular, versione 2.x.

I contenuti

La struttura del libro, come abbiamo appena visto, segue le necessità del lettore, aiutandolo a comprendere tutto il necessario e poi a utilizzare Angular. Innanzitutto installeremo l'ambiente di sviluppo composto da un software per la scrittura del codice e il software necessario alla gestione delle dipendenze legate al framework. Una volta preparata la macchina, installeremo bootstrap e sass, indispensabili per la realizzazione delle view, e Angular CLI, che facilita molto lo startup di un progetto Angular. Proseguiremo approfondendo

la nuova versione di JavaScript, ES6, con un'attenzione particolare al funzionamento delle classi. Dopo aver approfondito ES6 utilizzeremo TypeScript, facendo conoscenza degli aspetti base necessari all'uso di Angular. Una volta fatta la conoscenza di TypeScript inizieremo lo studio di Angular, soffermandoci passo dopo passo su tutti gli aspetti del framework, dalla logica alla pratica, con esempi correlati da codice. Terminato lo studio degli aspetti principali, metteremo in pratica quanto acquisito, realizzando una vera e propria applicazione web, con tanto di inserimento dati, foto e geolocalizzazione, offrendo la possibilità di cercare intorno a noi dei punti di interesse. In questa fase verranno inoltre approfonditi temi come l'ottimizzazione del codice o la pubblicazione dell'applicazione in produzione, attraverso la compilazione del codice. L'ultimo tema trattato sarà quello di un framework per applicazioni mobile, come Ionic2, che si basa su Angular. Tutto questo vi consentirà, senza troppa fatica, di apprendere una buona base di partenza per leggere la documentazione online con più semplicità e procedere in autonomia verso lo sviluppo di applicazioni web, mobile o entrambe.

- Capitolo 1, *Ambiente di sviluppo* – Seguiamo il lettore nell'installazione di tutto ciò che servirà per produrre tutti gli esempi. Introduciamo boostrap e un'estensione css. Il primo step è caratterizzato dal classico "Hello World".
- Capitolo 2, *Angular CLI* – Analizziamo il pacchetto Angular CLI apprezzandone le caratteristiche peculiari. È composto da una suite di software di ultima generazione e verrà analizzato file per file. Il precedente "Hello World" verrà ora realizzato con l'ausilio di Angular.
- Capitolo 3, *Un po' di teoria* – Introduzione agli argomenti necessari alla comprensione dell'architettura di Angular e al suo utilizzo. Concetti di pattern architetturali, la scelta e l'evoluzione del linguaggio JavaScript fino al superset.
- Capitolo 4, *Introduzione a EcmaScript 2015* – Breve introduzione alle novità del nuovo standard EcmaScript, che di fatto rende il linguaggio JavaScript adatto a progetti di grandi dimensioni.
- Capitolo 5, *TypeScript, diamogli fiducia* – Esempio dopo esempio vedremo quanto sia immediato l'apprendimento di TypeScript. Verranno presentati esempi pratici orientati strettamente a ciò che verrà sviluppato nei capitoli successivi.
- Capitolo 6, *Design Angular* – Presentazione dettagliata dell'architettura Angular. Svisceriamo tutti gli elementi che lo compongono, accompagnando il tutto con codice dimostrativo.
- Capitolo 7, *Binding e comunicazione* – Il binding dei dati è probabilmente il fulcro sul quale si basa lo sviluppo in Angular. Questo capitolo analizza ogni possibile collegamento tra la view e il modello. Inoltre approfondisce l'aspetto della condivisione dei dati tra elementi differenti.
- Capitolo 8, *Prima fase pratica* – Una volta ottenuta una buona visione d'insieme, procediamo creando un applicativo d'esempio. L'applicazione ingloberà le principali funzionalità che sono presenti nelle app più comuni.
- Capitolo 9, *Acceleriamo un po'* – Proseguiamo lo sviluppo dell'app implementando alcune funzionalità avanzate. Approfondimenti e accorgimenti tecnici aiuteranno il lettore a comprendere meglio le potenzialità offerte dal framework.
- Capitolo 10, *Ionic2* – Angular e Ionic2 consentono di realizzare app mobile che poco hanno da invidiare alle app native. Il framework Ionic annoverava nel 2015 più di

1,3 milioni di app create. A oggi è sicuramente uno tra i più popolari e apprezzati mobile framework.

- Capitolo 11, *Consigli conclusivi* – Alla fine di un lungo percorso si traggono sempre delle conclusioni. L'ultimo capitolo si guarda indietro e racconta ciò che abbiamo volutamente dimenticato di raccontare, consigliando ciò che deve essere assolutamente approfondito prima di lanciarsi nello sviluppo di una nuova applicazione. Dispenseremo gli ultimi consigli su come affrontare i problemi più comuni e suggeriremo alcune valide guide di riferimento.

Codice degli esempi

A questo libro sono affiancate alcune risorse reperibili online.
Tutto il codice degli esempi è liberamente disponibile ai seguenti indirizzi:

- `https://bitbucket.org/vincenzo_giacchina/angularbook/`
- `http://www.angular-italia.it/src/angularbook.zip`

Il sorgente è costituito dalle tre applicazioni presentate nel libro:

- HelloW, è la base Angular CLI che può essere utilizzata per iniziare a esaminare la struttura base creata dal framework.
- SaveBooks è l'applicativo più completo e consiglio lo studio della sua struttura che prevede l'uso dei servizi, direttive e componenti.
- HelloW-Ionic2 viene riproposto come applicazione Android in Ionic2.

Questi applicativi devono essere valutati esclusivamente come caso di studio. Il consiglio è quello di replicare anche lo stesso identico codice ma non limitarsi alla sua modifica. Il copia incolla, per quanto sia un'operazione comune, rischia, almeno nella fase iniziale di apprendimento, di limitare la memorizzazione dei processi chiave dell'uso di un framework.
Infine, per qualsiasi dubbio o problema è possibile fare riferimento al gruppo Facebook Angular Italia:

- `https://www.facebook.com/groups/AngularItalianCommunity` che estende e completa l'attività della community
- `http://www.angular-italia.it`

Buona lettura.