# 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 scriverlo. 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 mobili. 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 richiedono, 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 una sfida impossibile. 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

Esaminiamo ora cos'è una libreria e come differisce da un framework. Uno dei maggiori esempi legati allo sviluppo del Web è sicuramente jQuery, che, pur essendo una libreria, ha influenzato l'evoluzione dell'ecosistema web e indirettamente contribuito all'emergere dei framework. La sua popolarità e l'efficienza nell'astrazione della manipolazione del DOM (*Document Object Model*) hanno evidenziato la necessità di strumenti più strutturati e integrati, come i framework, per gestire applicazioni web sempre più complesse.


**Figura I.1** Il logo del progetto jQuery.

jQuery è stata, e per molti aspetti continua a essere, una delle librerie JavaScript più impiegate, che ha notevolmente influenzato lo sviluppo web nell'ultimo decennio, forse anche ventennio. Come ogni buona libreria, jQuery ha reso lo sviluppo web più acces-

sibile, specialmente per quanto riguarda la manipolazione degli elementi di una pagina. Anziché accedere direttamente alle risorse native del browser per manipolare un elemento specifico, come per esempio modificare un input box, è possibile delegare tale compito a jQuery, semplificando enormemente il lavoro grazie alla sua semplicità di utilizzo.
La necessità di conoscere a fondo il funzionamento interno del browser è stata sostituita dall'apprendimento di jQuery. Questa libreria ha agito da strumento di astrazione, rendendo più semplice l'interazione con il DOM.
Sebbene jQuery non sia un framework, ha svolto un ruolo importante nell'astrazione e nella semplificazione dello sviluppo web, influenzando indirettamente lo sviluppo dei framework. Con la continua espansione del Web e la crescente necessità di migliorare l'interazione con l'utente, l'obiettivo è sempre stato quello di migliorare costantemente la *user experience*. Le applicazioni web si sono evolute diventando sempre più complesse, simili a veri e propri software desktop. Questo sviluppo ha portato a un'evoluzione della struttura di base delle applicazioni web, passando dall'uso di jQuery per semplici script a modelli progettuali più strutturati, noti come "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, giunto a oggi alla versione 17, 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. Il pattern, assolutamente moderno per via della sua modularità, offre importanti funzionalità (*feature*) per la gestione degli stati, il *routing*, l'integrazione con servizi e molto altro.
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 maggiormente 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. In effetti, anche React con la successiva implementazione degli *hook* ha guadagnato maggiore espressività, semplicità dal punto di vista dell'implentazione del codice e migliore riutilizzabilità.
Diversamente, Angular facilita senza alcun dubbio l'avvio del progetto, organizzando meglio la sua struttura. Più procederete nella lettura di queste pagine, più vi risulterà chiaro il senso di questa affermazione.

## Premessa

Prima di buttarci a capofitto sugli aspetti pratici, i capitoli iniziali 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 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. 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 da Google, è un progetto open source supportato da un'enorme community che partecipa attivamente al suo sviluppo. Angular ha segnato l'inizio di un'era, in cui JavaScript, e più specificamente TypeScript, un suo potente superset, sono diventati protagonisti nella creazione di esperienze utente dinamiche e ricche. Angular è un framework che crea applicazioni che vengono elaborate all'interno del browser e per questo motivo viene definito un framework lato client, nonostante offra anche la possibilità di eseguire la renderizzazione lato server.
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, oggi si tende a fare il contrario. Il motivo di questo cambiamento è duplice: da un lato, la necessità di interazioni più rapide e fluide con l'utente finale; dall'altro, l'aumento esponenziale della potenza di calcolo dei dispositivi client. Angular si inserisce perfettamente in questo contesto, permettendo una gestione efficiente e performante lato client.
Con il susseguirsi delle versioni, Angular ha continuato a evolversi, ottimizzando le sue prestazioni, soprattutto per quanto riguarda le *Progressive Web Apps* (PWA). Le strategie di lazy loading, introdotte per moduli e componenti, hanno reso possibile un caricamento delle risorse più snello e veloce, ottimizzandone l'utilizzo e migliorando l'esperienza utente. Ma c'è di più: Angular Universal entra in gioco per colmare il gap lato server, specialmente per esigenze legate alla SEO (*Search Engine Optimization*)e all'interazione con le *preview* dei social network. Questo potente strumento consente il rendering lato server (SSR) delle applicazioni Angular, portando un netto miglioramento nelle prestazioni e nell'ottimizzazione per i motori di ricerca. È una caratteristica che non solo esalta le qualità di Angular, ma risponde anche a esigenze moderne di visibilità e accessibilità web.
In conclusione, Angular non è solo un framework, ma una vera e propria piattaforma che si adatta e risponde alle mutevoli esigenze del web moderno. Che si tratti di creare applicazioni reattive, gestire complesse interazioni utente o ottimizzare la SEO, Angular si conferma una scelta strategica, versatile e all'avanguardia.

## App single page

Angular è uno strumento ideale per lo sviluppo di applicazioni *single page*, comunemente note come SPA (*Single Page Application*). Questo tipo di applicazioni si caratterizza per l'offerta di un'esperienza utente fluida e interattiva, molto simile a quella delle applicazioni desktop. In una SPA, l'interazione avviene in una "pagina" unica, dove i contenuti vengono caricati o sostituiti dinamicamente senza necessità di ricaricare l'intera pagina a ogni cambio di link. Come si può immaginare, ciò si traduce in una maggiore efficienza e in un'interfaccia utente più reattiva.

**NOTA**

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

Ogni view in una SPA è sostenuta da una struttura logica che include dati e codice TypeScript. Questa struttura definisce il comportamento dell'interfaccia utente. Angular guida gli sviluppatori attraverso un design pattern specifico, che determina come organizzare efficacemente queste view e il codice correlato. Le prime versioni di Angular, conosciute come AngularJS, si basavano principalmente sul modello MVC (*Model-View-Controller*). Tuttavia, le versioni più recenti di Angular hanno adottato un approccio basato su componenti e servizi. Questo cambiamento ha reso le applicazioni non solo più semplici da testare e mantenere, ma ha anche migliorato la gestione della logica di business. Inoltre, Angular ora si integra perfettamente con una serie di nuovi strumenti e piattaforme, come Angular CLI per l'automazione del workflow e RxJS per la programmazione reattiva, fondamentale nella gestione delle operazioni asincrone. Queste evoluzioni fanno di Angular uno strumento estremamente potente e adatto alla creazione di applicazioni web moderne, performanti e facilmente manutenibili.

## 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 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 im-

posta, 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).

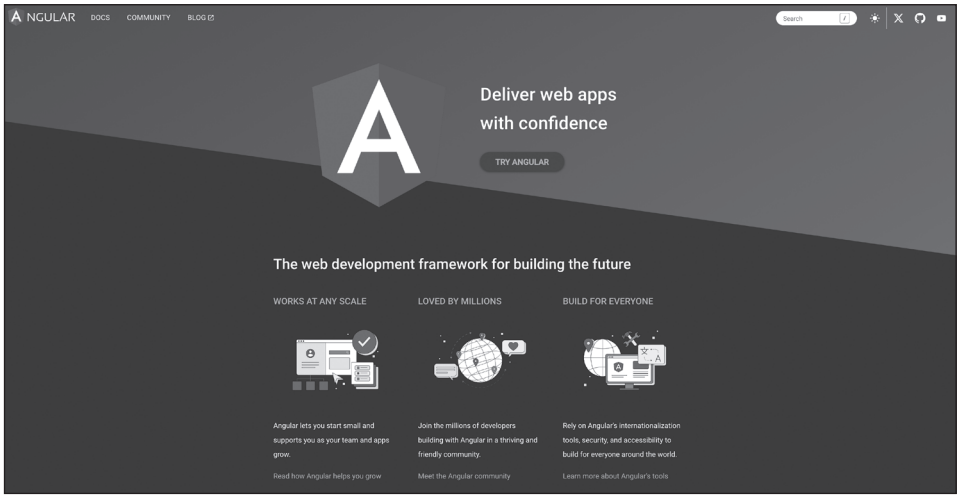


**Figura I.2** Home page del nuovo progetto Angular (https://angular.io/).

L'evoluzione di Angular alla nuova versione 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. Sebbene Angular supportasse inizialmente tre linguaggi differenti, ovvero JavaScript, TypeScript e Dart, la scelta di adottare TypeScript come linguaggio principale del framework è stata guidata dalla sua affinità con le pratiche di sviluppo moderne e una migliore integrazione con gli attuali strumenti di sviluppo. In conclusione, TypeScript è diventato il linguaggio di riferimento per lo sviluppo durante il processo di rifatorizzazione da AngularJS ad Angular. 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.

### Rifattorizzazione

Processo di riscrittura migliorativa delle fondamenta di un software.

### DOM

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

Le versioni più recenti di Angular offrono un approccio avanzato alla manipolazione del DOM, sfruttando una combinazione di tecniche atte a migliorare le prestazioni. L'uso di un'architettura basata su componenti permette di costruire applicazioni web dove ogni componente gestisce il proprio stato e la propria porzione del DOM, limitando così le interazioni dirette e migliorando l'efficienza. Inoltre, Angular si avvale di un sistema di rilevamento delle modifiche, noto come *change detection*, che aggiorna il DOM in maniera selettiva e solo all'occorrenza. Tale meccanismo garantisce che le modifiche vengano applicate nel modo più efficiente possibile. Un altro strumento a disposizione degli sviluppatori è Renderer2, un livello di astrazione fornito da Angular per interagire con il DOM. Renderer2 consente di manipolare il DOM in modo sicuro e compatibile con diverse piattaforme, riducendo i rischi di vulnerabilità e migliorando la portabilità del codice. Questo facilita inoltre il testing, permettendo agli sviluppatori di apportare modifiche in un ambiente controllato senza toccare gli elementi nativi. Come ulteriore punto, Angular supporta il Shadow DOM, che permette di incapsulare stili e markup dei componenti, prevenendo i conflitti di stile e marcando una netta separazione tra il componente e il contesto globale in cui è inserito. Questo approccio aiuta a mantenere l'isolamento tra i componenti e favorisce la creazione di elementi personalizzati riutilizzabili. Altri aspetti, come aperd esempio la compilazione del codice, consentono di efficientare l'interazione con il DOM, ma vedremo tutto questo più avanti con il giusto livello di dettaglio.

**NOTA**

Nell'evoluzione dalla versione 1.x alla successiva, 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`. Dalla versione successiva ha quindi assunto il nome attuale, Angular, e l'indirizzo di riferimento è `https://angular.io`.

## Introduzione alla nuova edizione

La prima edizione di questo volume fu pubblicata nel 2017, poco dopo che Angular avesse introdotto la sua innovativa versione 2.x. Questa edizione aggiornata alla versione 17 esplora le versioni evolute del framework. Dopo anni di sviluppo su una piattaforma tanto cruciale e diffusa, possiamo testimoniare e documentare progressi notevoli. La tecnologia di base è rimasta chiaramente la stessa. La sua architettura a componenti rimane un design fondamentale, così come la conseguente modularità del codice e il principio di riutilizzo. Il suo design rimane, a oggi, un tratto distintivo del framework. Un altro aspetto cardine è l'uso di TypeScript, che rimane un saldo compagno di viaggio nella vita del framework, così come il sistema di *dependency injection*, che rimane un importante approccio nella gestione del codice. Altri aspetti, come per esempio direttive, binding, routing e navigazione, lazy loading, decoratori, form e Angular CLI, rimangono coerenti nell'evoluzione del framework. Per quanto gli aspetti base e lo studio del framework, da una media distanza d'osservazione, siano rimasti comprensibili anche affrontandoli con la prima edizione del suddetto volume, è scontato che Angular ha subito negli anni una costante evoluzione atta a migliorare notevolmente le prestazioni e il livello generale di sicurezza. Numerose funzionalità sono state implementate nel susseguirsi delle versioni

del framework, e ciò ha consentito agli sviluppatori di creare applicazioni sempre più complesse, performanti, sicure e scalabili.
Rispetto alla versione trattata nel 2017, a oggi sono state implementate nuove funzionalità e migliorie varie, tra le quali, non in ordine di importanza, la sostituzione del motore di rendering introdotto in Angular 9 chiamato Ivy, che ha consentito al framework di fare un salto di qualità in termini di efficienza e dimensione del bundle. Inoltre, è stato ottimizzato il processo di creazione dei bundle in JavaScript, esterni ad Angular, anche grazie all'ausilio di esbuild, garantendo così la possibilità di velocizzare il processo di *building* e *bundling*.

> **NOTA**
>
> Il processo di bundling si occupa di combinare diversi file sorgenti in un unico file. È un processo molto utilizzato nel mondo web e garantisce una maggiore efficienza.

In termini di efficienza, anche lo sviluppo dell'interfaccia utente ha visto delle evoluzioni interessanti. Tra le più importanti vi sono sicuramente lo sviluppo di funzionalità legate alla parte di rendering lato server e la possibilità dell'uso del framework Tailwind CSS, introdotto grazie all'uso di esbuild e del tool ngx-tailwind. Anche Angular Material ha accompagnato le successive versioni del framework, arricchendo il suo catalogo componenti per uno sviluppo sempre più funzionale e accattivante dell'interfaccia utente. Dal punto di vista dello sviluppo del codice sorgente vi è sicuramente l'introduzione dei componenti standalone che consentono la creazione di applicazioni indipendenti dai moduli Angular; questo aspetto consente una notevole possibilità di riusabilità del codice oltre a un incremento delle performance e una migliore gestione del codice, in quanto il componente sarà indipendente dall'applicazione stessa. Un altro aspetto che facilita di molto lo sviluppo del codice è l'introduzione di un binding all'interno delle rotte, che di fatto, almeno nei casi meno complessi, rende superfluo l'uso di ulteriori classi. Lo stesso approccio più snello è utilizzato anche per accedere dinamicamente ai dati del router all'interno del componente. In tema di dinamismo, una nuova funzionalità riguarda la facilità di gestione dei cambiamenti di stato dell'applicazione tramite l'uso di Angular Signals. Infine, anche il tema sicurezza è stato approfondito e sviluppato lungo il percorso di vita del framework, con l'implementazione di una prevenzione più accurata dagli attacchi *Cross-Site Scripting* (XSS). Insomma, tante nuove funzionalità che chi ha già esperienza con le versioni precedenti del framework troverà molto interessanti. Questa nuova edizione si propone di guidarvi attraverso le nuove caratteristiche di Angular con la stessa chiarezza e profondità che avete apprezzato nella prima edizione, fornendo tutti gli strumenti necessari per sfruttare al meglio le potenzialità di questo eccezionale framework.

## I contenuti

Questo libro è strutturato per rispondere alle esigenze dei lettori, guidandoli nella comprensione e nell'utilizzo di Angular. Inizieremo con l'installazione dell'ambiente di sviluppo, che include il software per la scrittura del codice e gli strumenti necessari per gestire le dipendenze di Angular. Una volta preparato il nostro ambiente di lavoro, installeremo Bootstrap e Sass, indispensabili per la realizzazione delle view. Ci

concentreremo poi su Angular CLI, fondamentale per l'avvio di un progetto Angular e la sua gestione. Proseguiremo approfondendo le nuove versioni di ECMAScript, con un'attenzione particolare a un approccio moderno al linguaggio JavaScript. Dopo aver approfondito JavaScript affronteremo TypeScript, facendo conoscenza degli aspetti base orientati all'uso di Angular. Acquisita una solida conoscenza di TypeScript, ci immergeremo nello studio di Angular, esaminando tutti gli aspetti del framework, dalla teoria alla pratica, con esempi di 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 diversi temi legati al design del codice. L'ultimo argomento trattato sarà lo sviluppo di applicazioni mobili, con un focus particolare su framework come Ionic, noto per il suo pieno supporto ad Angular. Questo vi permetterà di acquisire con facilità una solida base di conoscenza per iniziare a sviluppare le vostre prime applicazioni. Inoltre, potrete integrare conoscenze più avanzate, attingendo con semplicità dalla documentazione online, e procedere in modo autonomo nello sviluppo di applicazioni sempre più sofisticate, sia web che mobili.

- Capitolo 1, *Ambiente di sviluppo* – Seguiamo il lettore nell'installazione di tutto ciò che servirà per produrre tutti gli esempi. Introduciamo Bootstrap e Sass. 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* – Breve introduzione alle novità del nuovo standard ECMAScript, che di fatto rende il linguaggio JavaScript adatto a progetti di grandi dimensioni.
- Capitolo 5, *TypeScript* – 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 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, *Ionic* – Angular e Ionic consentono di realizzare app mobili. Il framework Ionic rimane, soprattutto per il legame con Angular e lo sviluppo web in generale, una soluzione semplice ed efficace.
- 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 sul sito di Apogeo all'indirizzo `https://bit.ly/apo-sane` e al seguente indirizzo:

`https://bitbucket.org/vincenzo_giacchina/workspace/projects/AN`

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 di servizi, direttive e componenti;
- HelloW-Ionic viene riproposto come applicazione Android in Ionic.

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 e 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.

Buona lettura.