Introduzione

Negli ultimi anni si è visto il lento e graduale passaggio dal web statico, denominato Web 1.0, fino a quello che oggi chiamiamo Web 3.0.
Inizialmente il Web era costituito principalmente da siti statici, pagine dense di immagini e informazioni testuali, senza offrire la possibilità di interazioni con l'utente. I siti web erano creati principalmente con l'ausilio di linguaggi di markup come HTML e, successivamente, CSS, e il contenuto veniva presentato in modo lineare.
La navigazione da parte dell'utente avveniva attraverso link ipertestuali (la vera rivoluzione concettuale di quegli anni), e non esisteva la possibilità di personalizzare il contenuto presentato.
Successivamente si è assistito a un secondo stadio dello sviluppo web: il Web 2.0. In questa fase, il Web è diventato molto più interattivo e i siti hanno cominciato a offrire funzionalità avanzate come la condivisione dei contenuti e la collaborazione in tempo reale. Grazie a linguaggi di programmazione come JavaScript per la parte front-end e PHP per la parte back-end, i siti web sono passati dall'essere ambienti statici a presentare contenuti sempre più dinamici e personalizzabili.
Oltre a questo, nel tempo, i siti web hanno iniziato a sfruttare le API (*Application Programming Interface*) per integrarsi con altre applicazioni e fornire funzionalità avanzate come la geolocalizzazione e la condivisione sui social media.
Tutto questo innovare e progredire a livello tecnologico ci ha portato a quello che oggi viene comunemente chiamato Web 3.0.
In questo momento, il Web, definito anche Web semantico, mira a creare un'esperienza utente più personalizzata, decentralizzata e sicura, e offre un livello di interazione e intelligenza mai visto prima. L'intelligenza artificiale (IA) sta prendendo sempre più piede, ed è ormai una realtà consolidata e a disposizione di tutti.
E tra tutte le innovazioni che il Web 3.0 ha portato spicca di sicuro la tecnologia della *blockchain*, la quale ha stravolto completamente il concetto su cui si basa la quasi totalità delle applicazioni a oggi conosciute.
Fino a qualche anno fa, il termine blockchain era sconosciuto a molti, mentre oggi, complice anche la diffusione negli ultimi anni delle criptovalute, sta diventando sempre più presente e sta guadagnando sempre più spazio all'interno di progetti di innovazione digitale.
Sicuramente, il campo in cui questa tecnologia ha visto il suo più ampio sviluppo è quello legato al mondo finanziario. Le caratteristiche di decentralizzazione e sicurezza su cui è basata l'infrastruttura della blockchain hanno permesso il diffondersi in larga scala di un sistema monetario (le criptovalute) non più legato a entità centrali (le banche), ma che

potesse essere gestito dal singolo individuo che diventa in questo modo proprietario senza alcun vincolo del suo patrimonio.
Ma la blockchain non è solamente questo! La sua architettura basata su un registro condiviso e distribuito, e la possibilità di implementare i cosiddetti *smart contract* all'interno della chain, rende possibile la creazione di applicazioni innovative e sicure in molti ambiti, da quello finanziario a quello artistico, da progetti di sicurezza a sistemi di identità digitale.
Gli smart contract, definiti anche "contratti intelligenti", come vedremo, non sono nati insieme alla blockchain, ma grazie alla visione di un brillante programmatore russo (Vitalik Buterin); sono basati su un linguaggio di programmazione specifico che permette di definire le regole e le condizioni che devono essere rispettate per l'esecuzione del contratto.
Uno dei principali vantaggi è la loro capacità di automatizzare processi, eliminando la necessità di intermediari e rendendo le transazioni più sicure ed efficienti. Inoltre, consentono di creare applicazioni decentralizzate, in cui i dati sono distribuiti su nodi della rete e le decisioni sono prese in modo democratico senza un'autorità centrale.
In questo libro esploreremo il funzionamento di questi contratti: inizieremo con una panoramica sulla tecnologia blockchain, dopodiché esamineremo il linguaggio di programmazione Solidity, il più utilizzato per lo sviluppo di smart contract.
Una volta acquisite le competenze necessarie per la scrittura di uno smart contract, vedremo quali strumenti utilizzare per la stesura del codice e per il rilascio su una blockchain privata (per testarne le funzionalità) e sulla blockchain pubblica. Analizzeremo a fondo Remix, un framework gratuito online, e una serie di prodotti collaterali legati allo sviluppo e al mondo blockchain in generale (come per esempio Ganache e l'*hot wallet* MetaMask).
Procederemo, attraverso casi di studio di difficoltà progressiva, nella scrittura di contratti sempre più complessi fino ad arrivare ad analizzare lo sviluppo di una nuova moneta digitale e alla creazione di arte digitale, composta da quelli che vengono comunemente chiamati NFT (*Non Fungible Token*).
Successivamente ci concentreremo sulla creazione dell'interfaccia web, utilizzando la libreria web3.js, che ci permette di interagire e di sfruttare le funzionalità che lo smart contract mette a disposizione.
Alla fine del libro avrete una conoscenza completa del mondo blockchain, di tutto quello che ci ruota attorno e, cosa ben più importante, sarete completamente in grado di strutturare e sviluppare autonomamente un'applicazione decentralizzata, per quanto riguarda sia la parte back-end che la parte front-end.

Suddivisione dei capitoli

Il Capitolo 1 presenta una panoramica sulla blockchain in generale, sul suo funzionamento e su come a oggi è possibile sviluppare programmi intelligenti, chiamati smart contract, che permettono di eseguire delle azioni in autonomia.
Il Capitolo 2 è uno studio del linguaggio Solidity. Viene presentata l'intera sintassi logica e grammaticale che occorre conoscere per creare smart contract funzionanti.
Il Capitolo 3 presenta tre strumenti per lo sviluppo di smart contract: Remix, Ganache e MetaMask. Remix è un framework online completamente gratuito con il quale sarà possibile scrivere, compilare, rilasciare e testare i propri smart contract sia in ambiente di test che direttamente sulla mainnet di Ethereum; verranno analizzate tutte le funzio-

nalità utili al lettore per prendere confidenza con lo strumento e utilizzarlo per gli scopi prefissati da questo libro. Ganache è una rete di test privata che vi permetterà di testare in locale i vostri contratti. MetaMask è un portafoglio digitale che permette di mettere in comunicazione gli smart contract con le applicazioni web.
Dal Capitolo 4 si entra nel vivo della programmazione: attraverso casi di studio di difficoltà progressiva, si cercherà di portare il lettore ad avere piena conoscenza sullo sviluppo di smart contract attraverso il linguaggio Solidity.
Il Capitolo 5 introduce il lettore alla programmazione front-end, vale a dire lo sviluppo dell'interfaccia web che dovrà interagire con lo smart contract presente sulla blockchain. Il lettore verrà istruito su come interrogare uno smart contract e modificare i dati presenti sulla blockchain utilizzando la libreria JavaScript web3.js.
Il Capitolo 6 presenta le reti di test pubbliche, utili per lo sviluppo di applicazioni che non possono essere testate su reti private come Ganache. Sviluppare una nuova moneta, oppure creare un NFT, sono tutte operazioni che necessitano di una blockchain pubblica. Per tale ragione, la conoscenza e l'utilizzo di testnet pubbliche risulta indispensabile per una completa padronanza delle tecniche di programmazione legate al mondo Web 3.0.
Il Capitolo 7 fornisce una panoramica dettagliata su cosa sono i *token fungibili* e porta il lettore alla creazione di una propria criptomoneta digitale.
Il Capitolo 8 fornisce una panoramica dettagliata su cosa sono i *token non fungibili* (NFT) e porta il lettore alla creazione di un'intera collezione di arte digitale composta da essi.
Il Capitolo 9 tratta in maniera dettagliata degli oracoli e della loro utilità all'interno di un sistema decentralizzato, offrendo al lettore gli strumenti per poter richiedere in tempo reale la quotazione di qualsiasi criptovaluta.
Il Capitolo 10 presenta un intero progetto di una DApp che permette, fra le altre cose, di trasferire fondi da un indirizzo all'altro. L'intero progetto, sia per quanto riguarda la parte relativa allo smart contract, sia per quanto riguarda la parte front-end, viene analizzato in ogni singola istruzione affinché il lettore abbia piena conoscenza di come tutti gli elementi che compongono una DApp cooperano insieme per ottenere il risultato finale.
Il Capitolo 11 contiene le considerazioni finali, un breve glossario e gli indirizzi di alcune risorse utili.

Scaricare i file di codice di esempio

Potete scaricare i file di codice di esempio dal sito di Apogeo all'indirizzo `https://bit.ly/apo-saw3`.

Convenzioni utilizzate nel libro

In tutto il testo, potreste imbattervi in brevi paragrafi aggiuntivi inerenti l'argomento trattato. Questi sono di sei tipologie diverse:

ATTENZIONE

Situazioni a cui fare particolare attenzione.

APPROFONDIMENTO

Brevi approfondimenti riguardanti il tema trattato.

CURIOSITÀ

Curiosità riguardanti il tema trattato.

DEFINIZIONE

Spiegazione dettagliata di un determinato termine.

NOTA

Informazioni aggiuntive sul contesto corrente.

SUGGERIMENTO

Piccoli accorgimenti che permettono di migliorare la qualità del codice che si sta scrivendo.

Durante la stesura del testo si è cercato di rendere la materia il più esauriente ed esplicativa possibile facendo attenzione a non introdurre errori sia di forma che di sostanza. Qualora doveste riscontrare anomalie, oppure per comunicare qualsiasi tipo di idea e suggerimento valido per una prossima edizione, lo potete fare scrivendo all'indirizzo `alex.baldini72@gmail.com`.
Che il viaggio nel mondo della programmazione decentralizzata abbia inizio.