EDIZIONE DIGITALE · ITALIANO

Il nuovo *creatore* tecnico.

ESTRATTO GRATUITO

Introduzione · Indice completo · Capitolo 2

AUTORE

Hernán Capucci

Edizione indipendente · 2026

Questo estratto include l'introduzione, l'indice completo e un capitolo rappresentativo del libro. L'obiettivo è permetterti di valutare il tono, l'approccio e il tipo di apprendimento prima di accedere all'edizione completa.

Cosa include questo estratto

- L'introduzione completa (Capitolo 0).
- L'indice completo del libro.
- Il Capitolo 2 per intero: *Come funziona internet*.
- Informazioni di continuità.

Indice completo

Capitolo 0 — Perché esiste questo libro

Parte: Introduzione **Extensión estimada:** 12 pagine (~3.200 parole) **Términos clave:** software, sistema, ecosistema digitale, IA generativa, deploy, repository

Un giorno chiedi all'IA di generarti qualcosa — codice per una funzionalità, un'automazione che colleghi due servizi, un frammento che risolva un problema concreto. Quello che ti restituisce sembra funzionare. Lo usi. E a un certo punto di quel processo affiora una domanda che forse non hai il coraggio di fare ad alta voce: come faccio a sapere se è fatto bene?

Quella domanda non è secondaria. È la differenza tra operare con criterio proprio e operare sperando che nessuno trovi l'errore.

Questo libro esiste perché tu possa rispondere a quella domanda.

Accesso senza comprensione

Qualche anno fa, costruire **software** richiedeva una formazione tecnica specifica. Imparare un linguaggio di programmazione, capire le strutture dati, configurare ambienti di sviluppo, leggere documentazione che dava per scontato che tu sapessi già molto. Era una barriera alta e reale.

Poi sono arrivati gli strumenti di **intelligenza artificiale generativa**. Improvvisamente, chiunque poteva scrivere un'istruzione in linguaggio naturale e ricevere codice funzionante, un'automazione configurata, un sistema collegato a servizi esterni. La barriera tecnica si è abbassata in modo significativo.

Ma si è abbassata la barriera tecnica, non quella concettuale.

Che uno strumento generi codice al posto tuo non significa che capisci cosa ha generato. Che una piattaforma pubblichi la tua applicazione con un clic non significa che capisci cos'è un deploy, cosa può rompersi o come tornare indietro quando qualcosa va storto. Che l'IA ti spieghi un errore non significa che sai perché è successo né come evitare che succeda di nuovo.

L'accesso è migliorato. La comprensione del **sistema** che c'è sotto, in molti casi, non ha tenuto lo stesso ritmo.

Creare non significa capire

Oggi ci sono più persone che costruiscono prodotti digitali senza formazione tecnica che in qualsiasi momento precedente. App, automazioni, integrazioni tra servizi, piattaforme di contenuti, processi interni digitalizzati.

Questa democratizzazione è reale ed è preziosa. Il problema compare quando qualcosa non funziona come previsto, o quando bisogna prendere decisioni sul sistema che si è costruito.

Un imprenditore che ha usato l'IA per costruire il suo negozio online non sa perché "è andato giù il database". Non sa se le informazioni dei suoi clienti sono protette. Non sa se il lavoro che gli ha consegnato lo sviluppatore che ha assunto è fatto bene o no. Non sa che domande fargli per scoprirlo.

Un professionista che ha automatizzato i suoi processi con l'IA non capisce con precisione cosa fa ogni parte di ciò che ha messo insieme. Accetta il risultato perché ieri funzionava. Quando smette di funzionare, non ha gli strumenti per capire da dove cominciare a cercare.

Una fondatrice che lavora con un team tecnico esterno annuisce quando le parlano di **API**, di **repository** o di architettura del sistema. Ma non capisce. E non chiede, perché non sa cosa chiedere.

In tutti questi casi il problema non è mancanza di intelligenza né di capacità. È mancanza del vocabolario e del quadro concettuale che permettono di operare nell'ecosistema digitale con criterio proprio.

L'IA come punto di partenza

L'intelligenza artificiale è ciò che ha reso urgente questo libro. È la sveglia.

Ma lo scopo del libro non è parlare di intelligenza artificiale. È parlare dei sistemi che esistono sotto qualsiasi progetto digitale, con l'IA o senza.

Una API non è un concetto legato all'IA. Esiste da decenni. Nemmeno un **repository** di codice è una novità. La differenza tra un ambiente di sviluppo e uno di produzione non l'ha inventata nessun modello linguistico. L'autenticazione degli utenti, i database relazionali, il ciclo di vita di un **deploy** — tutto questo esisteva prima dell'IA generativa e continuerà a esistere dopo.

Ciò che è cambiato con l'IA è che adesso queste cose sono alla portata di persone che prima non le toccavano. E questa accessibilità senza comprensione crea nuove dipendenze.

Dipendenza dallo strumento: se cambia, se si rompe, se dà risultati inattesi, non hai modo di valutarli. Dipendenza dallo sviluppatore che invece capisce: se cambia, se alza le tariffe, se sparisce, non puoi andare avanti senza di lui. Dipendenza da chi dice di saperne di più: senza un quadro tuo, non puoi distinguere la conoscenza reale dalle promesse vuote.

L'IA ha aperto una porta. Questo libro dà il vocabolario per capire cosa c'è dentro.

Il vocabolario che manca

C'è un equivoco frequente che vale la pena chiarire fin dall'inizio.

Il problema di cui parla questo libro non è che non sai programmare. Non saper programmare non è il problema. Un'enorme quantità di persone molto efficaci nel mondo digitale non sa programmare e non ha bisogno di impararlo.

Ciò che conta non è saper scrivere codice. Ciò che conta è saperlo leggere con criterio sufficiente per capire cosa fa. Ciò che conta è capire cos'è un **database** per poter prendere decisioni su come archiviare le informazioni. Sapere cos'è un repository per poter controllare ciò che uno sviluppatore ti ha consegnato. Sapere cos'è un **endpoint** per poter chiedere all'IA di costruirne uno con precisione. Sapere cos'è un ambiente di produzione per non rompere qualcosa dal vivo per errore.

Niente di tutto questo richiede di scrivere una sola riga di codice.

Pensa a questa situazione concreta: lo sviluppatore ti dice "è fallita la migrazione nell'ambiente di staging". Senza contesto, quella frase è rumore. Con il vocabolario di questo libro, capisci che "migrazione" è un cambiamento nella struttura del database, che "staging" è l'ambiente di prova precedente alla produzione, e che il problema probabilmente non ha ancora toccato gli utenti — ma va risolto prima di andare avanti. Quella differenza non richiede di programmare. Richiede di capire i concetti.

Quello che richiede, semmai, è vocabolario tecnico. Il vocabolario che permette di fare le domande giuste, interpretare le risposte, valutare le proposte, riconoscere i segnali d'allarme e prendere decisioni con criterio. Quel vocabolario non si acquisisce usando gli strumenti. Si

acquisisce capendo i concetti che ci stanno dietro.

È questo che il libro costruisce. Concetto per concetto, da zero, con esempi universali e senza dare per scontata nessuna conoscenza pregressa.

Per chi è questo libro

Questo libro è scritto per le persone che costruiscono cose digitali senza formazione tecnica e vogliono farlo con più criterio.

Questo include chi sta mettendo insieme il suo primo prodotto digitale con l'aiuto dell'IA e non capisce del tutto cosa sta costruendo. Chi lavora con un team tecnico esterno e vuole poter parlare con loro senza che ogni conversazione sia uno sforzo di traduzione. Chi usa strumenti di automazione nel lavoro di tutti i giorni e vuole capire cosa sta facendo davvero. Chi ha un'idea e ha bisogno di valutarla tecnicamente prima di impegnare tempo e denaro per svilupparla.

Non importa il settore né l'area di lavoro. Non importano nemmeno l'età né il livello di istruzione formale in tecnologia.

Quello che conta, invece, è che usi il computer con disinvoltura nel tuo lavoro o progetto. Che hai provato almeno qualche strumento di IA. E che vuoi capire meglio l'**ecosistema digitale** in cui operi.

Il punto di partenza può essere che non hai mai aperto un repository di codice. Che non hai mai configurato un ambiente di sviluppo. Che non hai chiaro qual è la differenza tra un server e il cloud. Nessuno di questi punti di partenza è un ostacolo. Questo libro comincia da lì.

Cosa non troverai

Prima di andare avanti, conviene essere espliciti su ciò che questo libro non fa.

Non insegna a programmare. Non è il manuale di nessun linguaggio di programmazione né di nessuno strumento specifico. Se è questo che cerchi, ci sono risorse migliori e più specializzate.

Non sostiene che l'IA fa tutto. Non è così. Ha capacità reali e limiti reali. Questo libro mostra

entrambe le cose con onestà.

Non promette che entro un tempo prestabilito avrai il tuo progetto pronto. I processi tecnici hanno una complessità e dei tempi propri. Minimizzarli sarebbe mentire.

Non dipende da nessuno strumento, piattaforma o servizio specifico. Quando si cita uno strumento concreto, è a titolo di esempio. I concetti che insegna valgono allo stesso modo con qualsiasi stack o ambiente tu usi.

Non parla di casi di successo particolari né di progetti reali come modelli da seguire. Gli esempi sono generici e universali per scelta.

Non dice "è facile" quando non lo è. Alcuni concetti di questo libro sono semplici. Altri richiedono attenzione e una seconda lettura. Quando qualcosa ha una complessità reale, il libro lo segnala.

Cosa troverai

Alla fine di questo libro, avrai una comprensione concreta di concetti che oggi forse conosci solo di nome.

Capirai cos'è un'applicazione vista da dentro: quale parte gestisce ciò che vedi sullo schermo, quale parte elabora la logica e i dati, e come queste due parti comunicano tra loro. Questo non ti farà saperla costruire da solo, ma ti farà capire di cosa stanno parlando quando te la descrivono o quando qualcosa si rompe.

Potrai aprire un repository di codice e leggerne la struttura: quali cartelle ci sono, cosa fa ogni parte del progetto, cosa dice la cronologia delle modifiche. Quell'informazione è disponibile in qualsiasi repository e oggi forse non sai di poterla usare per capire ciò che qualcuno ti ha consegnato.

Potrai scrivere un **prompt** con contesto sufficiente perché l'IA produca qualcosa di utile. La differenza tra un'istruzione vaga e un'istruzione precisa non è talento: è sapere di quale informazione ha bisogno lo strumento per lavorare bene.

Potrai controllare ciò che l'IA o uno sviluppatore ha generato con domande concrete: cosa fa esattamente questa parte? Cosa succede se questo campo arriva vuoto? C'è qualche sezione che accede a informazioni sensibili? Quelle domande non richiedono di saper programmare.

Richiedono di capire i concetti in gioco.

Potrai usare il glossario di questo libro come strumento di lavoro: aprire una definizione quando compare un termine sconosciuto, capirla, e tornare alla conversazione o al documento con più chiarezza di prima.

E potrai riconoscere quando una risposta — di una persona o di uno strumento — non è valutabile con le tue conoscenze attuali. Questo non significa sempre sapere la risposta giusta. Significa sapere quando la domanda che devi fare è migliore di quella che hai fatto.

Com'è organizzato

Il libro ha sei parti oltre a questo capitolo introduttivo, appendici pratiche e un glossario tecnico.

La **Parte I** copre i fondamenti: cos'è il software, come funziona internet, cos'è il modello client-server, cos'è un database e cos'è il codice. Sono i concetti più basilari dell'ecosistema digitale. Ogni parte successiva li dà per conosciuti.

La **Parte II** entra nell'architettura di un'applicazione: il **frontend**, il **backend**, le API, i database in profondità, l'autenticazione e le sessioni. È la struttura interna di qualsiasi prodotto digitale moderno.

La **Parte III** copre gli strumenti e i processi che fanno funzionare un progetto nel mondo reale: il controllo di versione con **Git**, il terminale dei comandi, gli ambienti di sviluppo, il deploy e la sicurezza di base.

La **Parte IV** è dedicata a lavorare con l'IA in modo intelligente: cosa può fare, dove fallisce in modo sistematico, come formulare un prompt efficace, come controllare ciò che produce, e come usare modelli, **agenti** e automazioni senza perdere il controllo del processo.

La **Parte V** mette insieme tutto nel contesto di costruire qualcosa di concreto: come prepararsi prima di iniziare, come comunicare con i team tecnici, come passare da un'idea a un prodotto funzionante.

La **Parte VI** sono appendici pratiche: checklist di riferimento rapido, comandi base di terminale e Git, modelli di prompt riutilizzabili e risorse per approfondire ogni area.

Alla fine del libro c'è un **glossario tecnico** con più di 200 termini spiegati in linguaggio chiaro, con esempi e senza gergo. Non è un'appendice decorativa. È una parte integrante del libro.

L'ordine delle parti non è arbitrario. Ognuna costruisce su quella precedente. Le parti su sistemi e strumenti hanno bisogno dei fondamenti della Parte I. Le parti su IA e costruzione hanno bisogno dell'architettura e degli strumenti. Se scegli di leggere in ordine, il motivo è questo.

Come usarlo

Ci sono due modi di leggere questo libro, ed entrambi sono validi.

Il primo è tutto d'un fiato, dall'inizio alla fine. È quello consigliato per chi ha poco o nessun contatto precedente con l'ecosistema digitale. Il percorso costruisce comprensione in modo cumulativo: ogni capitolo prepara il terreno per quello successivo.

Il secondo è come libro di consultazione. Qualcuno ti ha citato una API e non hai capito cos'è: apri il Capitolo 8. Stai per fare il tuo primo deploy e non sai cosa comporta: apri il Capitolo 14. Hai bisogno di capire cos'è l'**autenticazione** prima di una riunione sulla sicurezza: apri il Capitolo 10. Ogni capitolo può essere letto in modo indipendente, anche se i riferimenti incrociati indicano cosa conviene aver letto prima.

Il glossario funziona allo stesso modo. Quando un termine tecnico compare per la prima volta in un capitolo, è in **grassetto** con un riferimento al glossario. Nella versione digitale del libro, quel riferimento è un link attivo: puoi andare direttamente alla definizione e tornare al capitolo con un clic. Nella versione cartacea, il riferimento indica il numero di pagina.

Una nota sugli esempi: lungo tutto il libro troverai gli stessi tipi di contesti — un negozio online, un'app di gestione delle prenotazioni, una piattaforma di corsi. Sono situazioni generiche scelte perché non richiedono nessuna conoscenza pregressa di nessun settore. I concetti valgono esattamente allo stesso modo in qualsiasi altro tipo di progetto.

Una nota sugli strumenti: quando si cita uno strumento come esempio, è proprio questo, un esempio. I concetti che questo libro insegna non dipendono da nessuna piattaforma specifica. Se lo strumento che usi oggi cambia domani, i concetti restano validi.

Creare con criterio proprio

C'è una differenza tra chi usa gli strumenti digitali senza capirli e chi li usa sapendo cosa sta facendo.

Quella differenza non sta nel saper programmare. Sta nell'avere il vocabolario per fare le domande giuste. Il quadro concettuale per valutare le risposte. La capacità di accorgersi quando qualcosa non va, anche se non sai esattamente perché. L'autonomia di prendere decisioni senza dipendere completamente dagli altri per capire cosa sta succedendo.

È questo il profilo che il libro costruisce: il nuovo creatore tecnico. Qualcuno che usa l'IA, lavora con team tecnici, costruisce cose digitali — e lo fa con criterio proprio.

Non è un programmatore. Non ha la pretesa di esserlo. Ma non opera nemmeno alla cieca, non accetta risultati senza poterli valutare né dipende dal fatto che un altro gli spieghi tutto.

Capisce il sistema in cui lavora. Sa cosa chiedere, come valutare ciò che riceve e quando riconoscere che qualcosa non va. Quella comprensione non richiede di scrivere codice. Richiede di capire come funziona ciò che usi.

Cosa hai imparato in questo capitolo

- L'IA ha democratizzato l'accesso all'esecuzione tecnica, ma non la comprensione dell'ecosistema digitale.
- Non saper programmare non è il problema. La mancanza di un vocabolario tecnico minimo invece lo è.
- Questo libro non insegna a programmare. Insegna i concetti che permettono di creare con criterio, parlare con i team tecnici e controllare ciò che produce l'IA.
- Il libro ha sei parti, appendici pratiche e un glossario di più di 200 termini. Può essere letto tutto d'un fiato o usato come consultazione.
- Nella versione digitale, i termini del glossario sono link attivi in ogni capitolo.

Cosa viene dopo

Il Capitolo 1 parte dall'inizio: cos'è il software e perché conta capirlo anche se non lo scriverai mai. È il primo blocco della Parte I e la base di tutto ciò che viene dopo.

Capitolo 2 — Come funziona internet

Parte: Parte I — Fondamenti dell'ecosistema digitale **Extensión estimada:** 10 pagine (~2.800 parole) **Términos clave:** URL, DNS, indirizzo IP, dominio, server, hosting, HTTP, HTTPS, codice di stato

Scrivi un indirizzo nel browser e premi Invio. In meno di un secondo compare una pagina: testo, immagini, dati aggiornati.

Da dove è arrivato tutto questo? Come è arrivato fino al tuo schermo? Perché a volte non arriva?

Quelle domande non richiedono di saper programmare per avere una risposta. Richiedono di capire l'infrastruttura che collega qualsiasi dispositivo con qualsiasi server nel mondo. Quella comprensione è la base per poter diagnosticare perché qualcosa funziona — e perché a volte no.

Il percorso di una richiesta

Quando scrivi un indirizzo nel browser, ciò che stai scrivendo si chiama **URL** — Uniform Resource Locator, ovvero localizzatore uniforme di risorse. È l'indirizzo che identifica una risorsa specifica su internet: una pagina, un'immagine, un file, un dato.

Il browser deve trovare il server che ha quella risorsa. Ma i server non si identificano con i loro nomi leggibili: si identificano con dei numeri. Ogni server ha un **indirizzo IP** — una sequenza di numeri che lo individua in modo univoco nella rete, un po' come un indirizzo postale individua un edificio in una città.

Il problema è che tu non scrivi numeri. Scrivi `www.enciclopedia.org` o `notizie.esempio.com`. Perché il browser possa trovare il server corrispondente, deve tradurre quel nome in un indirizzo IP. Quel processo si chiama risoluzione **DNS** — Domain Name System, ovvero sistema dei nomi di dominio.

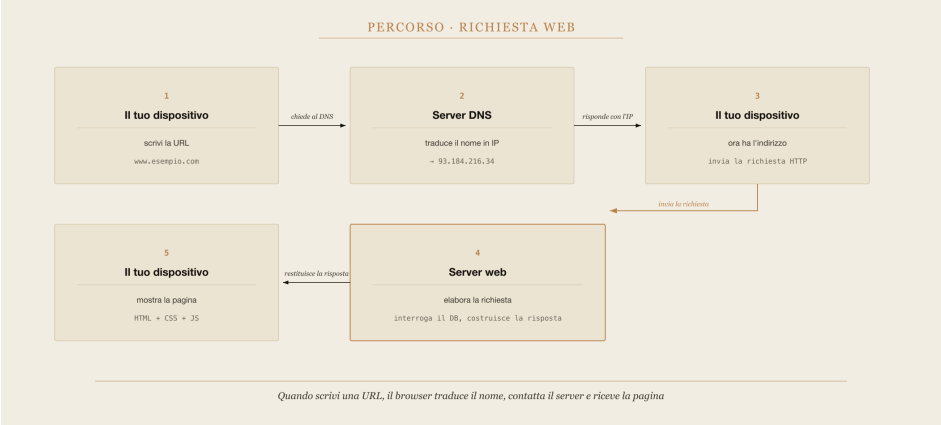

Figura 1. Quando scrivi un indirizzo, il browser non va in una nuvola magica: avvia una catena di traduzioni, richieste e risposte.

Il DNS funziona come l'elenco telefonico di internet. Il browser interroga un server DNS e gli chiede: "che IP ha `notizie.esempio.com`?" Il server DNS risponde con il numero. Il browser ha adesso l'indirizzo reale.

Con quell'indirizzo, il browser invia una richiesta al server. Quella richiesta viaggia attraverso la rete — cavi, fibre ottiche e connessioni wireless — fino ad arrivare al server che la aspetta. Il server la riceve, elabora ciò che deve elaborare, e restituisce una risposta. Quella risposta viaggia di nuovo fino al browser. Il browser interpreta il contenuto e lo mostra sullo schermo.

Tutto questo processo — interrogazione DNS, viaggio di andata, elaborazione, risposta di ritorno — avviene in frazioni di secondo in condizioni normali.

C'è un'ottimizzazione che vale la pena citare: la **cache**. Il browser non interroga il DNS ogni volta che visiti lo stesso sito. Salva la risposta — la traduzione da nome a IP — per un certo tempo. Lo stesso fa il sistema operativo. Questo rende più veloci le visite ripetute. Ma significa anche che, se il server ha cambiato indirizzo IP di recente, può darsi che il tuo dispositivo stia ancora usando l'indirizzo vecchio. Svuotare la cache del browser o aspettare che scada risolve quel problema.

Quando scrivi un indirizzo, il browser non va in una nuvola magica: avvia una catena di traduzioni, richieste e risposte.

Che cos'è un dominio

Un **dominio** è il nome leggibile che identifica un sito o un servizio su internet. `enciclopedia.org`, `notizie.esempio.com`, `lamiaapp.io` sono domini.

I domini non esistono da soli: qualcuno li registra. Ci sono aziende specializzate nella registrazione dei domini — si chiamano registrar — e il dominio si affitta per periodi di tempo, di solito un anno. Se non si rinnova, il dominio scade e torna disponibile perché un'altra persona lo registri.

La parte finale del dominio — `.com`, `.org`, `.io`, `.it` — si chiama dominio di primo livello o TLD (Top Level Domain). Indica il tipo di organizzazione o il paese di origine. Quella distinzione ha valore storico e di posizionamento, ma tecnicamente non cambia il funzionamento del sistema.

La parte che può confondere è la relazione tra dominio e server. Un dominio non è il server: è l'etichetta che punta al server. Uno stesso server può avere vari domini che puntano a lui. Uno stesso dominio può reindirizzare a server diversi a seconda del momento della giornata o della regione del mondo da cui si accede. Il DNS è il sistema che mantiene aggiornata quella relazione tra nomi e indirizzi.

Quando un dominio non è configurato correttamente — o quando è scaduto — il DNS non riesce a risolvere l'indirizzo. Il browser non sa a quale server andare. La pagina non si carica, anche se il server che la contiene funziona perfettamente.

Che cos'è un server — e che cos'è l'hosting

Un **server** è un computer progettato per ricevere richieste e restituire risposte. In sostanza, non è diverso da qualsiasi computer: ha processore, memoria, archiviazione e connessione alla rete.

La differenza pratica è che un server è acceso in modo permanente, connesso a internet con alta disponibilità, e configurato per gestire più richieste simultanee. Non ha schermo né tastiera. Non è sulla scrivania di nessuno. Vive in un data center — una struttura progettata per tenere molti server in funzione in modo continuo e sicuro.

L'**hosting** è il servizio che ti affitta spazio e risorse su quei server. Quando "pubblichi" qualcosa su internet — una pagina, un'applicazione, un file — quello che stai facendo è collocare quel contenuto su un server di un'azienda di hosting, configurare un dominio perché punti lì, e lasciare che il server lo consegni a chi lo richiede.

Ci sono diverse forme di hosting. In quella più basilare, condividi il server con altri siti — è più economico ma con risorse limitate. In forme più avanzate, hai un server virtuale o fisico esclusivo — più caro ma con maggiore controllo e capacità. Nel modello cloud, le risorse si assegnano dinamicamente a seconda della domanda.

Per chi costruisce qualcosa di digitale, capire cos'è l'hosting significa capire che il sistema che hai creato vive in un luogo fisico, che quel luogo ha dei limiti di capacità, e che se quel luogo si guasta — o se il dominio non punta correttamente a lui — il sistema non è accessibile anche se è costruito perfettamente.

Una distinzione utile: il server web è il componente che riceve le richieste HTTP e restituisce contenuto. Il server applicativo è quello che esegue la logica del sistema — cosa che si vedrà più in dettaglio nel Capitolo 3. In molti sistemi semplici, entrambe le funzioni girano sulla stessa macchina. In sistemi più grandi, sono separate. Quello che conta per ora è che "il server" non è una sola cosa monolitica: è un insieme di pezzi che lavorano insieme per rispondere alle richieste.

HTTP e HTTPS: la lingua del web

Quando il browser invia una richiesta e il server restituisce una risposta, entrambi devono parlare la stessa lingua. Quella lingua si chiama **protocollo**. Un protocollo è un insieme di regole che definisce come i messaggi vengono formattati, trasmessi e interpretati.

Il protocollo del web è **HTTP** — HyperText Transfer Protocol. Definisce la struttura delle richieste e delle risposte: quale informazione va per prima, come si indica il tipo di contenuto, come si comunica il risultato dell'operazione.

Una richiesta HTTP di base include: il metodo (cosa vuoi fare — ottenere informazioni, inviare dati, eliminare qualcosa), l'indirizzo della risorsa, e le intestazioni che contengono informazioni aggiuntive su chi sta chiedendo e che tipo di risposta accetta. Una risposta HTTP

include: il codice di stato (se ha funzionato o no), le intestazioni con informazioni sul contenuto, e il corpo — il contenuto vero e proprio.

HTTPS è la versione sicura di HTTP. La "S" sta per "Secure". La differenza tecnica è che in HTTPS tutta la comunicazione viaggia cifrata: i dati che il browser invia al server e quelli che il server restituisce non possono essere letti da chi intercetti la connessione lungo il percorso.

Quella cifratura è resa possibile da un **certificato SSL/TLS** — un file digitale che il server presenta al browser per dimostrare di essere chi dice di essere, e che attiva la cifratura della comunicazione. I browser moderni mostrano un lucchetto nella barra degli indirizzi quando la connessione usa HTTPS.

Perché conta per chi costruisce: qualsiasi sistema che gestisca dati degli utenti — password, informazioni personali, pagamenti — deve usare HTTPS senza eccezioni. Senza HTTPS, quei dati viaggiano in chiaro nella rete e possono essere catturati. Con HTTP, il browser può mostrare un avviso di "sito non sicuro" che allontana gli utenti e riduce la fiducia nel sistema.

> ***Concetto chiave: protocollo*** *Insieme di regole che definisce come due parti comunicano tra loro. HTTP è il protocollo che usano i browser e i server per scambiarsi informazioni sul web. HTTPS è la sua versione cifrata.*

Cosa risponde il server

Ogni volta che un server riceve una richiesta, restituisce una risposta che include un numero: il **codice di stato**. Quel numero indica cosa è successo alla richiesta.

I codici si raggruppano per categorie. Quelli che iniziano con 2 significano successo. Quelli che iniziano con 3 sono reindirizzamenti. Quelli che iniziano con 4 indicano un errore dal lato del client — qualcosa nella richiesta non va. Quelli che iniziano con 5 indicano un errore dal lato del server — il server ha ricevuto la richiesta ma non è riuscito a elaborarla correttamente.

I tre più importanti per chi lavora con i sistemi:

200 — OK. La richiesta è stata elaborata correttamente e il server ha restituito il contenuto atteso. È il risultato normale. Se vedi una pagina, il server ha risposto 200.

404 — Not Found. Il server ha ricevuto la richiesta, ma la risorsa che hai chiesto non esiste su quel server. L'indirizzo che hai usato non corrisponde a nessun file, pagina o dato disponibile.

Non è un errore del server: è che ciò che cercavi non è lì.

500 — Internal Server Error. Il server ha ricevuto la richiesta, ha provato a elaborarla, e qualcosa si è rotto nel suo stesso funzionamento. Il problema non sta in ciò che hai chiesto: sta in come il server ha gestito la richiesta. È un errore del codice o della configurazione del server.

Codice	Nome	Cosa è successo	Esempio quotidiano
200	OK	Il server ha trovato e consegnato ciò che hai chiesto.	Apri una pagina e si carica tutto senza problemi.
301 / 302	Reindirizzamento	La risorsa è a un altro indirizzo. Il browser ti porta lì automaticamente.	Una URL vecchia ti manda al nuovo dominio del sito.
400	Bad Request	La richiesta è arrivata mal formata; il server non è riuscito a interpretarla.	Invii un modulo con dati obbligatori incompleti.
401	Unauthorized	Non hai presentato credenziali valide. Il server non sa chi sei.	Provi a vedere contenuto privato senza aver effettuato l'accesso.
403	Forbidden	Le tue credenziali sono valide, ma non hai il permesso per quello.	Hai effettuato l'accesso, ma quella sezione non corrisponde al tuo ruolo.
404	Not Found	Ciò che hai chiesto non esiste su quel server.	Una URL è stata scritta male o non esiste più.

Codice	Nome	Cosa è successo	Esempio quotidiano
500	Internal Server Error	Il server ha ricevuto la tua richiesta ma si è rotto nell'elaborarla.	Provi a completare un acquisto e compare "errore inatteso".
503	Service Unavailable	Il server non può rispondere adesso; è sovraccarico o in manutenzione.	Un sito va giù durante una vendita di massa o un lancio.

Non serve memorizzare tutti i codici. L'importante è riconoscere lo schema: i 2xx di solito indicano successo, i 3xx reindirizzamento, i 4xx un problema dal lato della richiesta e i 5xx un problema dal lato del server.

Perché l'app "è andata giù"

"Non si carica la pagina", "il sistema è giù", "non funziona" — quando qualcosa smette di essere disponibile, di solito c'è una causa tecnica specifica. Capire i possibili punti di rottura permette di circoscrivere il problema prima di chiamare qualcuno che lo risolva.

Le cause più comuni, ordinate per dove si verificano:

Il server è giù. Il server ha smesso di rispondere. Può essere perché il servizio di hosting ha avuto un problema, perché il server si è riavviato per un aggiornamento, o perché il sistema è rimasto senza risorse (memoria, CPU) e ha smesso di funzionare. In questo caso, il browser non riceve risposta di nessun tipo.

Il dominio non si risolve. Il DNS non riesce a tradurre il nome del dominio in un indirizzo IP. Può essere perché il dominio è scaduto, perché la configurazione del DNS è stata cambiata male, o perché c'è un problema con il server DNS. Il risultato è simile al precedente: il browser non sa dove andare.

Il certificato è scaduto. Se il certificato HTTPS del server è scaduto, il browser blocca la connessione e mostra un avviso di sicurezza. La pagina tecnicamente esiste e il server funziona,

ma il browser protegge l'utente rifiutando la connessione. Il risultato visibile: una schermata di avviso, non la pagina.

Errore 500. Il server risponde ma con un errore interno. La pagina esiste, il server è acceso, ma il codice ha trovato un problema nell'elaborare la richiesta. L'utente vede un messaggio di errore al posto del contenuto atteso.

Problema di rete. La connessione tra il dispositivo dell'utente e il server è interrotta in qualche punto intermedio. Può essere la connessione a internet dell'utente, un fornitore di rete, o un punto dell'infrastruttura tra i due lati.

Distinguerli a grandi linee aiuta a sapere da dove cominciare:

- Se la pagina non carica niente e non c'è nessun avviso → probabilmente server giù o DNS che non si risolve.
- Se il browser mostra un avviso di sicurezza esplicito → certificato scaduto o HTTP senza cifratura.
- Se la pagina si carica con un messaggio di errore nel contenuto → probabilmente errore 500, problema del codice.
- Se solo tu non riesci ad accedere ma gli altri sì → probabilmente un problema di rete o di cache locale.

Sapere questo non significa poterlo risolvere direttamente. Significa poterlo comunicare con precisione a chi invece può farlo.

> ***Domande utili quando un sito non si carica***
>
> *Quando il sistema non è disponibile, queste domande aiutano a circoscrivere il problema prima di chiamare chi può risolverlo:*
>
> - *Il problema capita a tutti o solo a me?*
> - *Il dominio si risolve? Il nome punta a qualche indirizzo?*
> - *Il server risponde? Arriva qualcosa, anche solo un errore?*
> - *Che codice di errore c'è — 404, 500 — o non c'è risposta di nessun tipo?*
> - *A non funzionare è la produzione o solo un ambiente di prova?*
> - *Ci sono log dell'errore?*
> - *C'è stato qualche cambiamento recente — deploy, modifica di configurazione, rinnovo del dominio?*

C'è un caso che vale la pena citare perché è frequente e confuso: il sistema funziona per alcune persone e non per altre. Questo quasi mai significa che il server sia giù. Di solito indica un problema di propagazione del DNS — quando si è fatto un cambiamento nella configurazione del dominio, quel cambiamento ci mette tra minuti e ore ad arrivare a tutti i server DNS del mondo. Durante quel tempo, utenti diversi interrogano server DNS diversi e ottengono risposte diverse. Alcuni vedono il sito nuovo; altri continuano a vedere il vecchio o non vedono niente. È un comportamento atteso, non un errore da risolvere d'urgenza.

Come lo sentirai in una riunione

In una riunione con un team tecnico o con un fornitore, si usano frasi che danno per scontato che tutti capiscano di cosa si sta parlando. Queste sono le più comuni su questo tema, cosa significano nella pratica, e cosa puoi chiedere.

"Sembra un problema di DNS." Il nome del dominio non si sta risolvendo correttamente — non si riesce a tradurlo nell'indirizzo IP del server. Può essere una configurazione sbagliata o un cambiamento recente che ancora non si è propagato del tutto.

Domanda utile: "Capita a tutti gli utenti o solo ad alcuni?" Se è a tutti, probabilmente è un errore di configurazione. Se è solo ad alcuni, è più probabile che sia una propagazione in corso — un processo che ci mette ore e si risolve da solo.

Non conviene dare per scontato: che il server sia giù. Il DNS e il server sono pezzi distinti. Il server può funzionare perfettamente mentre il DNS si rompe.

"Il server risponde 500." Il server ha ricevuto la richiesta ma ha trovato un errore nell'elaborarla. Il problema sta nel codice o nella configurazione del sistema, non nella richiesta dell'utente né nella rete.

Domanda utile: "Ci sono log dell'errore?" I log sono il registro di ciò che ha fatto il sistema prima di rompersi. Un 500 quasi sempre lascia una traccia che identifica la causa. Senza log, la diagnosi diventa molto più difficile.

Non conviene dare per scontato: che sia un problema di internet o del dispositivo dell'utente. Un 500 significa che il server ha effettivamente risposto — solo che con un errore interno.

"Il certificato è scaduto." Il certificato HTTPS del server è scaduto. Il browser rileva che la connessione non può più essere garantita come sicura e blocca l'accesso per impostazione predefinita. Il sistema può funzionare perfettamente all'interno, ma nessun utente può entrare.

Domanda utile: "Quanto ci mette il rinnovo?" Nella maggior parte dei casi è un processo rapido. Quello che conta sapere è il tempo stimato di risoluzione e se ci sono già utenti colpiti a cui comunicare qualcosa.

Non conviene dare per scontato: che ci sia un problema di codice o di infrastruttura. È una scadenza amministrativa, simile alla scadenza di un documento. Non indica niente sulla qualità del sistema.

"È giù in produzione." Il sistema che usano gli utenti reali non è disponibile. Non è un ambiente di prova né di sviluppo: è quello con il traffico reale e i dati reali. Questa frase indica urgenza.

Domanda utile: "Si sta già lavorando per risolverlo? Qual è il tempo stimato?" Quelle due domande danno il contesto necessario per sapere se bisogna comunicare qualcosa agli utenti o aspettare in silenzio.

Non conviene dare per scontato: che il problema colpisca tutti allo stesso modo. A volte il blocco è parziale — colpisce solo una regione, un dispositivo o una funzionalità specifica.

Cosa hai imparato in questo capitolo

- Una URL è l'indirizzo che identifica una risorsa su internet. Per arrivarci, il browser interroga il DNS, che traduce il nome del dominio in un indirizzo IP.
- Un server è un computer sempre acceso che riceve richieste e restituisce risposte. L'hosting è il servizio che affitta quel server.

- HTTP è il protocollo che definisce come comunicano il browser e il server. HTTPS è la sua versione cifrata, obbligatoria per qualsiasi sistema che gestisca dati sensibili.
- I codici di stato sono la lingua che usa il server per indicare cosa è successo: 200 è successo, 404 è risorsa non trovata, 500 è errore interno del server.
- "L'app è andata giù" ha cause tecniche specifiche: server giù, DNS che non si risolve, certificato scaduto, errore di codice. Conoscerle permette di circoscrivere il problema.

Cosa viene dopo

Il Capitolo 3 entra nei due attori di quella conversazione che hai appena capito: il client e il server. Sai già come viaggia l'informazione tra loro. Quello che viene dopo è capire cosa fa ciascuno in quello scambio — e perché quella divisione è la struttura fondamentale di qualsiasi applicazione.

Questo estratto finisce qui.

Il libro completo prosegue con la mappa completa:
client, server, database, codice,
API, sicurezza, intelligenza artificiale e deploy.

L'edizione completa include
22 capitoli, appendici pratiche,
un glossario tecnico e le versioni PDF + EPUB.

elnuevocreadortecnico.com