DIGITALE AUSGABE · DEUTSCH

Der neue *creator* technische.

LESEPROBE

Einleitung · Vollständiges Inhaltsverzeichnis · Kapitel 2

AUTOR

Hernán Capucci

Unabhängige Ausgabe · 2026

Diese Leseprobe enthält die Einleitung, das vollständige Inhaltsverzeichnis und ein repräsentatives Kapitel des Buches. So kannst du Ton, Ansatz und Art des Lernens einschätzen, bevor du zur vollständigen Ausgabe greifst.

Was diese Leseprobe enthält

- Die vollständige Einleitung (Kapitel 0).
- Das vollständige Inhaltsverzeichnis des Buches.
- Das komplette Kapitel 2: *Wie das Internet funktioniert*.
- Hinweise zur Kontinuität.

Vollständiges Inhaltsverzeichnis

Kapitel 0 — Warum es dieses Buch gibt

Eines Tages bittest du die KI, dir etwas zu erzeugen — Code für eine Funktion, eine Automatisierung, die zwei Dienste verbindet, ein Schnipsel, das ein konkretes Problem löst. Was

sie dir zurückgibt, scheint zu funktionieren. Du verwendest es. Und irgendwann in diesem Prozess taucht eine Frage auf, die du dich vielleicht nicht laut zu stellen traust: Woher weiß ich, ob das richtig ist?

Diese Frage ist nicht nebensächlich. Sie ist der Unterschied zwischen Handeln mit eigenem Urteilsvermögen und Handeln in der Hoffnung, dass niemand den Fehler findet.

Dieses Buch gibt es, damit du sie beantworten kannst.

Zugang ohne Verständnis

Vor einigen Jahren erforderte es eine spezielle technische Ausbildung, **Software** zu bauen. Eine Programmiersprache lernen, Datenstrukturen verstehen, Entwicklungsumgebungen einrichten, Dokumentation lesen, die voraussetzte, dass du schon viel weißt. Das war eine hohe und reale Hürde.

Dann kamen die Werkzeuge der **generativen künstlichen Intelligenz**. Auf einmal konnte jeder eine Anweisung in natürlicher Sprache schreiben und funktionierenden Code erhalten, eine eingerichtete Automatisierung, ein an externe Dienste angebundenes System. Die technische Hürde sank deutlich.

Aber gesunken ist die technische Hürde, nicht die konzeptionelle.

Dass ein Werkzeug Code für dich erzeugt, heißt nicht, dass du verstehst, was es erzeugt hat. Dass eine Plattform deine Anwendung mit einem Klick veröffentlicht, heißt nicht, dass du verstehst, was ein Deployment ist, was schiefgehen kann oder wie du zurückrollst, wenn etwas schiefläuft. Dass die KI dir einen Fehler erklärt, heißt nicht, dass du weißt, warum er auftrat oder wie du verhinderst, dass er erneut auftritt.

Der Zugang wurde besser. Das Verständnis des **Systems** darunter hielt in vielen Fällen nicht Schritt.

Bauen ist nicht dasselbe wie verstehen

Heute bauen mehr Menschen ohne technische Ausbildung digitale Produkte als je zuvor. Apps, Automatisierungen, Integrationen zwischen Diensten, Content-Plattformen, digitalisierte interne Abläufe.

Diese Demokratisierung ist real und sie ist wertvoll. Das Problem taucht auf, wenn etwas nicht wie erwartet funktioniert oder wenn Entscheidungen über das gebaute System anstehen.

Ein Gründer, der mit KI seinen Onlineshop gebaut hat, weiß nicht, warum „die Datenbank abgestürzt ist". Er weiß nicht, ob die Daten seiner Kunden geschützt sind. Er weiß nicht, ob die Arbeit, die der beauftragte Entwickler abgeliefert hat, gut gemacht ist oder nicht. Er weiß nicht, welche Fragen er stellen muss, um es herauszufinden.

Eine Fachkraft, die ihre Abläufe mit KI automatisiert hat, versteht nicht genau, was jeder Teil dessen tut, was sie zusammengebaut hat. Sie akzeptiert das Ergebnis, weil es gestern funktioniert hat. Wenn es nicht mehr funktioniert, fehlen ihr die Mittel, um zu verstehen, wo sie suchen soll.

Eine Gründerin, die mit einem externen Technikteam arbeitet, nickt, wenn von **APIs**, von **Repositories** oder von Systemarchitektur die Rede ist. Aber sie versteht es nicht. Und sie fragt nicht nach, weil sie nicht weiß, was sie fragen soll.

In all diesen Fällen ist das Problem weder mangelnde Intelligenz noch fehlende Fähigkeit. Es ist das Fehlen des Vokabulars und des begrifflichen Rahmens, die es erlauben, im digitalen Ökosystem mit eigenem Urteilsvermögen zu handeln.

Die KI als Ausgangspunkt

Die künstliche Intelligenz war es, die dieses Buch dringlich machte. Sie ist der Weckruf.

Aber das Ziel des Buches ist nicht, über künstliche Intelligenz zu sprechen. Es geht darum, über die Systeme zu sprechen, die unter jedem digitalen Projekt liegen — mit KI oder ohne sie.

Eine API ist kein KI-Konzept. Sie existiert seit Jahrzehnten. Ein **Repository** für Code ist auch nichts Neues. Den Unterschied zwischen einer Entwicklungsumgebung und einer Produktivumgebung hat kein Sprachmodell erfunden. Die Authentifizierung von Nutzern, die relationalen Datenbanken, der Lebenszyklus eines **Deployments** — all das gab es schon vor der generativen KI und wird es auch danach geben.

Was sich mit der KI geändert hat, ist, dass diese Dinge nun in Reichweite von Menschen liegen, die sie vorher nicht angefasst haben. Und diese Zugänglichkeit ohne Verständnis schafft neue Abhängigkeiten.

Abhängigkeit vom Werkzeug: Wenn es sich ändert, wenn es versagt, wenn es unerwartete Ergebnisse liefert, hast du keine Möglichkeit, sie zu beurteilen. Abhängigkeit vom Entwickler, der es versteht: Wenn er wechselt, wenn er seine Preise erhöht, wenn er verschwindet, kommst du ohne ihn nicht weiter. Abhängigkeit von dem, der vorgibt, mehr zu wissen: Ohne eigenen Rahmen kannst du echtes Wissen nicht von leeren Versprechen unterscheiden.

Die KI hat eine Tür geöffnet. Dieses Buch liefert das Vokabular, um zu verstehen, was dahinter liegt.

Das Vokabular, das fehlt

Es gibt eine häufige Verwechslung, die man von Anfang an ausräumen sollte.

Das Problem, um das es in diesem Buch geht, ist nicht, dass du nicht programmieren kannst. Nicht programmieren zu können ist nicht das Problem. Eine riesige Zahl sehr wirkungsvoller Menschen in der digitalen Welt kann nicht programmieren und muss es auch nicht lernen.

Worauf es ankommt, ist nicht, Code schreiben zu können. Worauf es ankommt, ist, ihn mit genug Urteilsvermögen lesen zu können, um zu verstehen, was er tut. Worauf es ankommt, ist, zu verstehen, was eine **Datenbank** ist, um Entscheidungen darüber treffen zu können, wie Informationen gespeichert werden. Zu wissen, was ein Repository ist, um prüfen zu können, was ein Entwickler dir abgeliefert hat. Zu wissen, was ein **Endpunkt** ist, um die KI präzise bitten zu können, einen zu bauen. Zu wissen, was eine Produktivumgebung ist, um nicht versehentlich etwas im laufenden Betrieb kaputtzumachen.

Nichts davon erfordert, auch nur eine einzige Zeile Code zu schreiben.

Stell dir diese konkrete Situation vor: Der Entwickler sagt dir „die Migration in der Staging-Umgebung ist fehlgeschlagen". Ohne Kontext ist dieser Satz nur Rauschen. Mit dem Vokabular dieses Buches verstehst du, dass eine „Migration" eine Änderung an der Struktur der Datenbank ist, dass „Staging" die Testumgebung vor der Produktion ist und dass das Problem die Nutzer wahrscheinlich noch nicht betroffen hat — aber gelöst werden muss, bevor es weitergeht. Dieser Unterschied erfordert kein Programmieren. Er erfordert, die Konzepte zu verstehen.

Was er sehr wohl erfordert, ist technisches Vokabular. Das Vokabular, das es erlaubt, die richtigen Fragen zu stellen, Antworten zu deuten, Vorschläge zu beurteilen, Warnsignale zu erkennen und Entscheidungen mit Urteilsvermögen zu treffen. Dieses Vokabular eignet man

sich nicht durch das Benutzen von Werkzeugen an. Man eignet es sich an, indem man die Konzepte versteht, die dahinterstehen.

Genau das baut dieses Buch auf. Konzept für Konzept, von Grund auf, mit universellen Beispielen und ohne Vorkenntnisse vorauszusetzen.

Für wen dieses Buch ist

Dieses Buch ist für Menschen geschrieben, die digitale Dinge ohne technische Ausbildung bauen und das mit mehr Urteilsvermögen tun wollen.

Dazu gehört, wer mithilfe von KI sein erstes digitales Produkt zusammenbaut und nicht ganz versteht, was er da baut. Wer mit einem externen Technikteam arbeitet und mit ihm sprechen können will, ohne dass jedes Gespräch eine Übersetzungsleistung wird. Wer im Arbeitsalltag Automatisierungswerkzeuge nutzt und verstehen will, was sie wirklich tun. Wer eine Idee hat und sie technisch beurteilen muss, bevor er Zeit und Geld in ihre Umsetzung steckt.

Die Branche oder der Arbeitsbereich spielt keine Rolle. Auch Alter oder formale Ausbildung im Bereich Technik spielen keine Rolle.

Worauf es sehr wohl ankommt: dass du in deiner Arbeit oder deinem Projekt sicher mit dem Computer umgehst. Dass du mindestens ein KI-Werkzeug ausprobiert hast. Und dass du das **digitale Ökosystem**, in dem du dich bewegst, besser verstehen willst.

Der Ausgangspunkt kann sein, dass du nie ein Repository für Code geöffnet hast. Dass du nie eine Entwicklungsumgebung eingerichtet hast. Dass dir nicht klar ist, was der Unterschied zwischen einem Server und einer Cloud ist. Keiner dieser Ausgangspunkte ist ein Hindernis. Dieses Buch beginnt genau dort.

Was du nicht finden wirst

Bevor es weitergeht, ist es gut, klar zu sagen, was dieses Buch nicht leistet.

Es lehrt nicht das Programmieren. Es ist kein Handbuch zu irgendeiner Programmiersprache oder zu irgendeinem bestimmten Werkzeug. Wenn du das suchst, gibt es dafür bessere und spezialisiertere Quellen.

Es behauptet nicht, dass die KI alles macht. Tut sie nicht. Sie hat reale Fähigkeiten und reale Grenzen. Dieses Buch zeigt beides ehrlich.

Es verspricht nicht, dass du in einer bestimmten Frist dein fertiges Projekt hast. Technische Prozesse haben ihre eigene Komplexität und ihre eigene Zeit. Sie kleinzureden wäre gelogen.

Es hängt von keinem bestimmten Werkzeug, keiner Plattform und keinem Dienst ab. Wenn ein konkretes Werkzeug erwähnt wird, dann als Beispiel. Die Konzepte, die hier vermittelt werden, gelten genauso mit jedem Stack und jeder Umgebung, die du nutzt.

Es spricht nicht von bestimmten Erfolgsgeschichten oder von realen Projekten als Vorbildern, denen man folgen soll. Die Beispiele sind bewusst generisch und universell.

Es sagt nicht „das ist einfach", wenn es das nicht ist. Manche Konzepte in diesem Buch sind schlicht. Andere erfordern Aufmerksamkeit und ein erneutes Lesen. Wenn etwas reale Komplexität hat, weist das Buch darauf hin.

Was du sehr wohl finden wirst

Wenn du dieses Buch beendet hast, wirst du ein konkretes Verständnis von Konzepten haben, die du heute vielleicht nur dem Namen nach kennst.

Du wirst verstehen, was eine Anwendung von innen ist: welcher Teil das verwaltet, was du auf dem Bildschirm siehst, welcher Teil die Logik und die Daten verarbeitet und wie diese beiden Teile miteinander kommunizieren. Das wird dich nicht in die Lage versetzen, sie allein zu bauen, aber sehr wohl zu verstehen, wovon die Rede ist, wenn man sie dir beschreibt oder wenn etwas schiefgeht.

Du wirst ein Repository für Code öffnen und seine Struktur lesen können: welche Ordner es gibt, was jeder Teil des Projekts tut, was der Änderungsverlauf erzählt. Diese Informationen stehen in jedem Repository bereit, und heute weißt du vielleicht nicht, dass du sie nutzen kannst, um zu verstehen, was dir jemand abgeliefert hat.

Du wirst einen **Prompt** mit genug Kontext schreiben können, damit die KI etwas Nützliches hervorbringt. Der Unterschied zwischen einer vagen und einer präzisen Anweisung ist kein Talent: Es ist das Wissen darum, welche Informationen das Werkzeug braucht, um gut zu arbeiten.

Du wirst prüfen können, was die KI oder ein Entwickler erzeugt hat, mit konkreten Fragen: Was genau tut dieser Teil? Was passiert, wenn dieses Feld leer ankommt? Gibt es einen Abschnitt, der auf sensible Informationen zugreift? Diese Fragen erfordern kein Programmieren. Sie erfordern, die beteiligten Konzepte zu verstehen.

Du wirst das Glossar dieses Buches als Arbeitswerkzeug nutzen können: eine Definition aufschlagen, wenn ein unbekannter Begriff auftaucht, sie verstehen und mit mehr Klarheit als zuvor ins Gespräch oder ins Dokument zurückkehren.

Und du wirst erkennen können, wann eine Antwort — von einer Person oder von einem Werkzeug — mit deinem aktuellen Wissen nicht beurteilbar ist. Das bedeutet nicht immer, die richtige Antwort zu kennen. Es bedeutet zu wissen, wann die Frage, die du stellen musst, besser ist als die, die du gestellt hast.

Wie es aufgebaut ist

Das Buch hat sechs Teile sowie dieses einleitende Kapitel, praktische Anhänge und ein technisches Glossar.

Teil I behandelt die Grundlagen: was Software ist, wie das Internet funktioniert, was das Client-Server-Modell ist, was eine Datenbank ist und was Code ist. Das sind die grundlegendsten Konzepte des digitalen Ökosystems. Jeder folgende Teil setzt sie als bekannt voraus.

Teil II geht in die Architektur einer Anwendung: das **Frontend**, das **Backend**, die APIs, die Datenbanken im Detail, die Authentifizierung und die Sitzungen. Es ist die innere Struktur jedes modernen digitalen Produkts.

Teil III behandelt die Werkzeuge und die Prozesse, die ein Projekt in der realen Welt zum Laufen bringen: Versionskontrolle mit **Git**, das Befehlsterminal, die Entwicklungsumgebungen, das Deployment und die grundlegende Sicherheit.

Teil IV ist dem klugen Arbeiten mit KI gewidmet: was sie kann, wo sie systematisch versagt, wie man einen wirkungsvollen Prompt formuliert, wie man prüft, was sie hervorbringt, und wie man Modelle, **Agenten** und Automatisierungen nutzt, ohne die Kontrolle über den Prozess zu verlieren.

Teil V fügt alles im Kontext zusammen, etwas Konkretes zu bauen: wie man sich vorbereitet, bevor man anfängt, wie man mit Technikteams kommuniziert, wie man von einer Idee zu einem funktionierenden Produkt kommt.

Teil VI besteht aus praktischen Anhängen: Checklisten zum schnellen Nachschlagen, grundlegende Befehle für Terminal und Git, wiederverwendbare Prompt-Vorlagen und Ressourcen zur Vertiefung in jedem Bereich.

Am Ende des Buches steht ein **technisches Glossar** mit über 200 Begriffen, in klarer Sprache erklärt, mit Beispielen und ohne Fachjargon. Es ist kein dekorativer Anhang. Es ist ein integraler Bestandteil des Buches.

Die Reihenfolge der Teile ist nicht willkürlich. Jeder baut auf dem vorherigen auf. Die Teile zu Systemen und Werkzeugen brauchen die Grundlagen aus Teil I. Die Teile zu KI und zum Bauen brauchen die Architektur und die Werkzeuge. Wenn du dich entscheidest, der Reihe nach zu lesen, ist das der Grund.

Wie du es nutzt

Es gibt zwei Wege, dieses Buch zu lesen, und beide sind gültig.

Der erste ist von vorne bis hinten am Stück. Er ist empfohlen für alle, die wenig oder gar keinen vorherigen Kontakt mit dem digitalen Ökosystem hatten. Der Weg baut das Verständnis Schritt für Schritt auf: Jedes Kapitel bereitet den Boden für das nächste.

Der zweite ist als Nachschlagewerk. Jemand hat dir eine API erwähnt und du hast nicht verstanden, was das ist: Du schlägst Kapitel 8 auf. Du wirst dein erstes Deployment machen und weißt nicht, was das bedeutet: Du schlägst Kapitel 14 auf. Du musst verstehen, was **Authentifizierung** ist, bevor du in ein Meeting über Sicherheit gehst: Du schlägst Kapitel 10 auf. Jedes Kapitel lässt sich eigenständig lesen, auch wenn die Querverweise zeigen, was du vorher gelesen haben solltest.

Das Glossar funktioniert genauso. Wenn ein technischer Begriff zum ersten Mal in einem Kapitel auftaucht, steht er in **Fettschrift** mit einem Verweis auf das Glossar. In der digitalen Version des Buches ist dieser Verweis ein aktiver Hyperlink: Du kannst direkt zur Definition springen und mit einem Klick ins Kapitel zurückkehren. In der gedruckten Version gibt der Verweis die Seitenzahl an.

Eine Anmerkung zu den Beispielen: Im Lauf des Buches wirst du auf immer wieder dieselben Arten von Kontexten stoßen — einen Onlineshop, eine App zur Terminverwaltung, eine Kursplattform. Es sind generische Situationen, gewählt, weil sie keinerlei Vorkenntnisse aus irgendeiner Branche erfordern. Die Konzepte gelten genau gleich bei jeder anderen Art von Projekt.

Eine Anmerkung zu den Werkzeugen: Wenn ein Werkzeug als Beispiel erwähnt wird, dann ist es genau das, ein Beispiel. Die Konzepte, die dieses Buch vermittelt, hängen von keiner

bestimmten Plattform ab. Wenn das Werkzeug, das du heute nutzt, morgen wechselt, bleiben die Konzepte gültig.

Mit eigenem Urteilsvermögen bauen

Es gibt einen Unterschied zwischen jemandem, der digitale Werkzeuge benutzt, ohne sie zu verstehen, und jemandem, der sie benutzt und weiß, was er tut.

Dieser Unterschied liegt nicht im Programmierenkönnen. Er liegt darin, das Vokabular zu haben, um die richtigen Fragen zu stellen. Den begrifflichen Rahmen, um Antworten zu beurteilen. Die Fähigkeit, zu erkennen, wann etwas nicht stimmt, auch wenn du nicht genau weißt, warum. Die Eigenständigkeit, Entscheidungen zu treffen, ohne völlig von anderen abhängig zu sein, um zu verstehen, was vor sich geht.

Das ist das Profil, das dieses Buch aufbaut: der neue technische Creator. Jemand, der KI nutzt, mit Technikteams arbeitet, digitale Dinge baut — und das mit eigenem Urteilsvermögen tut.

Er ist kein Programmierer. Er will keiner sein. Aber er handelt auch nicht im Blindflug, akzeptiert keine Ergebnisse, die er nicht beurteilen kann, und ist nicht darauf angewiesen, dass ein anderer ihm alles erklärt.

Er versteht das System, in dem er arbeitet. Er weiß, was er fragen muss, wie er beurteilt, was er bekommt, und wann er erkennt, dass etwas nicht stimmt. Dieses Verständnis erfordert kein Schreiben von Code. Es erfordert, zu verstehen, wie das funktioniert, was du nutzt.

Was du in diesem Kapitel gelernt hast

- Die KI hat den Zugang zur technischen Ausführung demokratisiert, aber nicht das Verständnis des digitalen Ökosystems.
- Nicht programmieren zu können ist nicht das Problem. Das Fehlen eines technischen Mindestvokabulars sehr wohl.
- Dieses Buch lehrt nicht das Programmieren. Es vermittelt die Konzepte, die es erlauben, mit Urteilsvermögen zu bauen, mit Technikteams zu sprechen und zu prüfen, was die KI hervorbringt.
- Das Buch hat sechs Teile, praktische Anhänge und ein Glossar mit über 200 Begriffen. Es lässt sich am Stück lesen oder als Nachschlagewerk nutzen.

- In der digitalen Version sind die Glossarbegriffe in jedem Kapitel aktive Hyperlinks.

Was als Nächstes kommt

Kapitel 1 beginnt ganz von vorn: was Software ist und warum es sich lohnt, sie zu verstehen, auch wenn du sie nie schreiben wirst. Es ist der erste Baustein von Teil I und die Grundlage für alles, was danach kommt.

Kapitel 2 — Wie das Internet funktioniert

Du gibst eine Adresse in den Browser ein und drückst Enter. In weniger als einer Sekunde erscheint eine Seite: Text, Bilder, aktuelle Daten.

Woher kam das alles? Wie ist es auf deinen Bildschirm gelangt? Warum kommt es manchmal nicht an?

Diese Fragen lassen sich beantworten, ohne programmieren zu können. Sie verlangen, die Infrastruktur zu verstehen, die jedes Gerät mit jedem Server der Welt verbindet. Dieses Verständnis ist die Grundlage dafür, diagnostizieren zu können, warum etwas funktioniert — und warum manchmal nicht.

Der Weg einer Anfrage

Wenn du eine Adresse in den Browser eingibst, heißt das, was du eingibst, **URL** — Uniform Resource Locator, also einheitlicher Ressourcen-Lokalisierer. Es ist die Adresse, die eine bestimmte Ressource im Internet identifiziert: eine Seite, ein Bild, eine Datei, ein Datum.

Der Browser muss den Server finden, der diese Ressource hat. Aber die Server identifizieren sich nicht über ihre lesbaren Namen: Sie identifizieren sich über Zahlen. Jeder Server hat eine **IP-Adresse** — eine Zahlenfolge, die ihn eindeutig im Netz verortet, ähnlich wie eine Postadresse ein Gebäude in einer Stadt verortet.

Das Problem ist, dass du keine Zahlen eingibst. Du gibst `www.enzyklopaedie.org` oder `nachrichten.beispiel.com` ein. Damit der Browser den entsprechenden Server finden kann, muss er diesen Namen in eine IP-Adresse übersetzen. Dieser Prozess heißt **DNS**-Auflösung — Domain Name System, also Domainnamensystem.

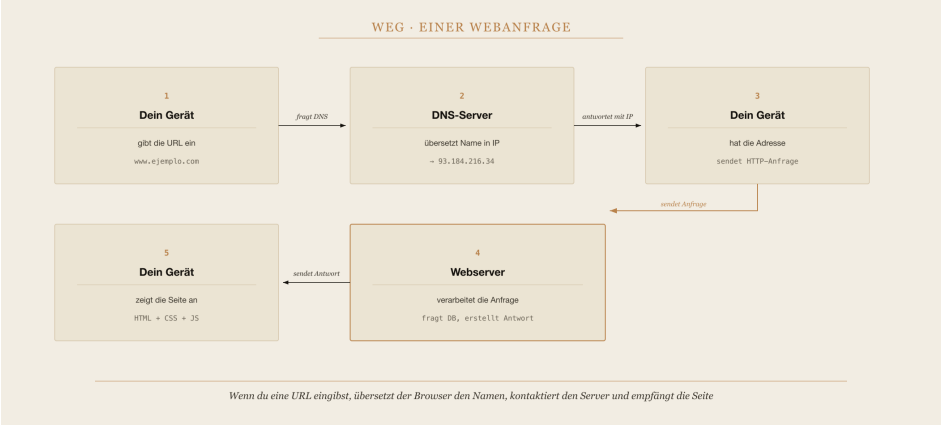

Abbildung 1. Wenn du eine Adresse eingibst, geht der Browser nicht zu einer magischen Wolke: Er startet eine Kette von Übersetzungen, Anfragen und Antworten.

Das DNS funktioniert wie das Telefonbuch des Internets. Der Browser fragt einen DNS-Server und fragt ihn: „Welche IP hat `nachrichten.beispiel.com`?" Der DNS-Server antwortet mit der Zahl. Der Browser hat jetzt die echte Adresse.

Mit dieser Adresse sendet der Browser eine Anfrage an den Server. Diese Anfrage reist durchs Netz — über Kabel, Glasfasern und Funkverbindungen — bis sie den Server erreicht, der sie erwartet. Der Server empfängt sie, verarbeitet, was er verarbeiten muss, und gibt eine Antwort zurück. Diese Antwort reist zurück bis zum Browser. Der Browser interpretiert den Inhalt und zeigt ihn auf dem Bildschirm an.

Dieser ganze Prozess — DNS-Abfrage, Hinreise, Verarbeitung, Antwort zurück — geschieht unter normalen Bedingungen in Sekundenbruchteilen.

Es gibt eine Optimierung, die es wert ist, erwähnt zu werden: den **Cache**. Der Browser fragt das DNS nicht jedes Mal ab, wenn du dieselbe Seite besuchst. Er speichert die Antwort — die Übersetzung von Name zu IP — für eine Weile. Dasselbe tut das Betriebssystem. Das macht wiederholte Besuche schneller. Aber es bedeutet auch, dass dein Gerät, wenn der Server kürzlich seine IP-Adresse geändert hat, vielleicht noch die alte Adresse verwendet. Den Cache des Browsers zu leeren oder zu warten, bis er abläuft, löst dieses Problem.

Wenn du eine Adresse eingibst, geht der Browser nicht zu einer magischen Wolke: Er startet eine Kette von Übersetzungen, Anfragen und Antworten.

Was eine Domain ist

Eine **Domain** ist der lesbare Name, der eine Seite oder einen Dienst im Internet identifiziert. `enzyklopaedie.org`, `nachrichten.beispiel.com`, `meineapp.io` sind Domains.

Domains existieren nicht von selbst: Jemand registriert sie. Es gibt auf die Registrierung von Domains spezialisierte Unternehmen — sie heißen Registrare —, und die Domain wird für bestimmte Zeiträume gemietet, in der Regel ein Jahr. Wird sie nicht verlängert, läuft die Domain ab und steht wieder zur Verfügung, damit eine andere Person sie registriert.

Der hintere Teil der Domain — `.com`, `.org`, `.io`, `.de` — heißt Top-Level-Domain oder TLD (Top Level Domain). Er zeigt die Art der Organisation oder das Herkunftsland an. Diese Unterscheidung hat historischen Wert und Wert für die Positionierung, aber technisch ändert sie die Funktionsweise des Systems nicht.

Der Teil, der verwirren kann, ist die Beziehung zwischen Domain und Server. Eine Domain ist nicht der Server: Sie ist das Etikett, das auf den Server zeigt. Ein und derselbe Server kann mehrere Domains haben, die auf ihn zeigen. Ein und dieselbe Domain kann je nach Tageszeit oder Weltregion, aus der zugegriffen wird, auf verschiedene Server umleiten. Das DNS ist das System, das diese Beziehung zwischen Namen und Adressen aktuell hält.

Wenn eine Domain nicht korrekt konfiguriert ist — oder wenn sie abgelaufen ist —, kann das DNS die Adresse nicht auflösen. Der Browser weiß nicht, zu welchem Server er gehen soll. Die Seite lädt nicht, auch wenn der Server, der sie enthält, perfekt funktioniert.

Was ein Server ist — und was das Hosting ist

Ein **Server** ist ein Computer, der dafür entworfen wurde, Anfragen zu empfangen und Antworten zurückzugeben. Im Grunde unterscheidet er sich nicht von jedem anderen Computer: Er hat Prozessor, Speicher, Speicherplatz und Netzanbindung.

Der praktische Unterschied ist, dass ein Server dauerhaft eingeschaltet ist, mit hoher Verfügbarkeit ans Internet angebunden und darauf ausgelegt, mehrere Anfragen gleichzeitig zu bedienen. Er hat weder Bildschirm noch Tastatur. Er steht auf niemandes Schreibtisch. Er lebt in einem Rechenzentrum — einer Einrichtung, die dafür entworfen ist, viele Server kontinuierlich und sicher am Laufen zu halten.

Das **Hosting** ist der Dienst, der dir Platz und Ressourcen auf diesen Servern vermietet. Wenn du etwas im Internet „veröffentlichst" — eine Seite, eine Anwendung, eine Datei —, dann

legst du diesen Inhalt auf einem Server eines Hosting-Unternehmens ab, konfigurierst eine Domain so, dass sie dorthin zeigt, und lässt den Server ihn an jeden ausliefern, der ihn anfragt.

Es gibt verschiedene Formen des Hostings. In der einfachsten teilst du dir den Server mit anderen Seiten — günstiger, aber mit begrenzten Ressourcen. In fortgeschritteneren Formen hast du einen eigenen virtuellen oder physischen Server — teurer, aber mit mehr Kontrolle und Kapazität. Im Cloud-Modell werden die Ressourcen je nach Nachfrage dynamisch zugewiesen.

Für jemanden, der etwas Digitales baut, heißt zu verstehen, was Hosting ist, zu verstehen, dass das System, das du erschaffen hast, an irgendeinem physischen Ort lebt, dass dieser Ort Kapazitätsgrenzen hat und dass das System, wenn dieser Ort ausfällt — oder wenn die Domain nicht korrekt auf ihn zeigt —, nicht erreichbar ist, auch wenn es perfekt gebaut ist.

Eine nützliche Unterscheidung: Der Webserver ist die Komponente, die die HTTP-Anfragen empfängt und Inhalt zurückgibt. Der Anwendungsserver ist der, der die Logik des Systems ausführt — was wir in Kapitel 3 genauer ansehen werden. In vielen einfachen Systemen laufen beide Funktionen auf derselben Maschine. In größeren Systemen sind sie getrennt. Was vorerst zählt, ist, dass „der Server" nicht eine einzelne, monolithische Sache ist: Er ist ein Satz von Teilen, die zusammenarbeiten, um Anfragen zu beantworten.

HTTP und HTTPS: die Sprache des Webs

Wenn der Browser eine Anfrage sendet und der Server eine Antwort zurückgibt, müssen beide dieselbe Sprache sprechen. Diese Sprache heißt **Protokoll**. Ein Protokoll ist ein Satz von Regeln, der festlegt, wie die Nachrichten formatiert, übertragen und interpretiert werden.

Das Protokoll des Webs ist **HTTP** — HyperText Transfer Protocol. Es legt die Struktur der Anfragen und der Antworten fest: welche Information zuerst kommt, wie die Art des Inhalts angegeben wird, wie das Ergebnis der Operation mitgeteilt wird.

Eine grundlegende HTTP-Anfrage umfasst: die Methode (was du tun willst — Information abrufen, Daten senden, etwas löschen), die Adresse der Ressource und Header, die zusätzliche Information darüber enthalten, wer fragt und welche Art von Antwort akzeptiert wird. Eine HTTP-Antwort umfasst: den Statuscode (ob es funktioniert hat oder nicht), Header mit Information über den Inhalt und den Body — den Inhalt selbst.

HTTPS ist die sichere Version von HTTP. Das „S" steht für „Secure". Der technische Unterschied ist, dass bei HTTPS die gesamte Kommunikation verschlüsselt reist: Die Daten, die der Browser an den Server sendet, und die, die der Server zurückgibt, können von jemandem, der die Verbindung unterwegs abfängt, nicht gelesen werden.

Diese Verschlüsselung ermöglicht ein **SSL/TLS-Zertifikat** — eine digitale Datei, die der Server dem Browser vorlegt, um zu beweisen, dass er der ist, der er zu sein behauptet, und die die Verschlüsselung der Kommunikation aktiviert. Moderne Browser zeigen ein Schloss in der Adressleiste, wenn die Verbindung HTTPS nutzt.

Warum das für jemanden wichtig ist, der baut: Jedes System, das Nutzerdaten verarbeitet — Passwörter, persönliche Information, Zahlungen —, muss ausnahmslos HTTPS verwenden. Ohne HTTPS reisen diese Daten im Klartext durchs Netz und können abgefangen werden. Bei HTTP kann der Browser eine Warnung „unsichere Seite" anzeigen, die Nutzer abschreckt und das Vertrauen ins System mindert.

> ***Schlüsselbegriff: Protokoll*** *Satz von Regeln, der festlegt, wie zwei Parteien miteinander kommunizieren. HTTP ist das Protokoll, das Browser und Server verwenden, um im Web Information auszutauschen. HTTPS ist seine verschlüsselte Version.*

Was der Server antwortet

Jedes Mal, wenn ein Server eine Anfrage empfängt, gibt er eine Antwort zurück, die eine Zahl enthält: den **Statuscode**. Diese Zahl zeigt an, was mit der Anfrage passiert ist.

Die Codes sind in Kategorien gruppiert. Die, die mit 2 beginnen, bedeuten Erfolg. Die, die mit 3 beginnen, sind Umleitungen. Die, die mit 4 beginnen, zeigen einen Fehler aufseiten des Clients an — etwas an der Anfrage stimmt nicht. Die, die mit 5 beginnen, zeigen einen Fehler aufseiten des Servers an — der Server hat die Anfrage empfangen, konnte sie aber nicht korrekt verarbeiten.

Die drei wichtigsten für jemanden, der mit Systemen arbeitet:

200 — OK. Die Anfrage wurde korrekt verarbeitet und der Server hat den erwarteten Inhalt zurückgegeben. Das ist das normale Ergebnis. Wenn du eine Seite siehst, hat der Server mit 200 geantwortet.

404 — Not Found. Der Server hat die Anfrage empfangen, aber die Ressource, die du angefragt hast, existiert auf diesem Server nicht. Die Adresse, die du verwendet hast, entspricht

keiner verfügbaren Datei, Seite oder keinem verfügbaren Datum. Es ist kein Fehler des Servers: Es ist nur, dass das, was du gesucht hast, nicht da ist.

500 — Internal Server Error. Der Server hat die Anfrage empfangen, versucht, sie zu verarbeiten, und etwas in seiner eigenen Funktionsweise ist fehlgeschlagen. Das Problem liegt nicht an dem, was du angefragt hast: Es liegt daran, wie der Server die Anfrage gehandhabt hat. Es ist ein Fehler des Codes oder der Konfiguration des Servers.

Code	Name	Was passiert ist	Alltägliches Beispiel
200	OK	Der Server hat gefunden und ausgeliefert, was du angefragt hast.	Du öffnest eine Seite und alles lädt problemlos.
301 / 302	Umleitung	Die Ressource ist an einer anderen Adresse. Der Browser bringt dich automatisch dorthin.	Eine alte URL schickt dich auf die neue Domain der Seite.
400	Bad Request	Die Anfrage kam fehlerhaft an; der Server konnte sie nicht interpretieren.	Du sendest ein Formular mit unvollständigen Pflichtdaten ab.
401	Unauthorized	Du hast keine gültigen Zugangsdaten vorgelegt. Der Server weiß nicht, wer du bist.	Du versuchst, private Inhalte zu sehen, ohne dich anzumelden.

Code	Name	Was passiert ist	Alltägliches Beispiel
403	Forbidden	Deine Zugangsdaten sind gültig, aber du hast keine Berechtigung dafür.	Du bist in deinem Konto, aber dieser Bereich entspricht nicht deiner Rolle.
404	Not Found	Was du angefragt hast, existiert auf diesem Server nicht.	Eine URL wurde falsch geschrieben oder existiert nicht mehr.
500	Internal Server Error	Der Server hat deine Anfrage empfangen, ist aber beim Verarbeiten fehlgeschlagen.	Du versuchst, einen Kauf abzuschließen, und es erscheint „unerwarteter Fehler".
503	Service Unavailable	Der Server kann jetzt nicht bedienen; er ist überlastet oder in Wartung.	Eine Seite bricht während eines großen Verkaufs oder Launchs zusammen.

Du musst nicht alle Codes auswendig lernen. Wichtig ist, das Muster zu erkennen: Die 2xx zeigen meist Erfolg an, die 3xx Umleitung, die 4xx ein Problem aufseiten der Anfrage und die 5xx ein Problem aufseiten des Servers.

Warum die App „abgestürzt" ist

„Die Seite lädt nicht", „das System ist abgestürzt", „funktioniert nicht" — wenn etwas nicht mehr verfügbar ist, gibt es meist eine spezifische technische Ursache. Die möglichen Fehlerpunkte zu verstehen, erlaubt es, das Problem einzugrenzen, bevor man jemanden ruft, der es löst.

Die häufigsten Ursachen, geordnet danach, wo sie auftreten:

Der Server ist abgestürzt. Der Server antwortet nicht mehr. Das kann daran liegen, dass der Hosting-Dienst ein Problem hatte, dass der Server wegen einer Aktualisierung neu gestartet wurde oder dass dem System die Ressourcen (Speicher, CPU) ausgegangen sind und es aufgehört hat zu funktionieren. In diesem Fall bekommt der Browser keinerlei Antwort.

Die Domain löst nicht auf. Das DNS kann den Namen der Domain nicht in eine IP-Adresse übersetzen. Das kann daran liegen, dass die Domain abgelaufen ist, dass die DNS-Konfiguration falsch geändert wurde oder dass es ein Problem mit dem DNS-Server gibt. Das Ergebnis ähnelt dem vorigen: Der Browser weiß nicht, wohin er gehen soll.

Das Zertifikat ist abgelaufen. Wenn das HTTPS-Zertifikat des Servers abgelaufen ist, blockiert der Browser die Verbindung und zeigt eine Sicherheitswarnung an. Die Seite existiert technisch und der Server funktioniert, aber der Browser schützt den Nutzer, indem er die Verbindung ablehnt. Das sichtbare Ergebnis: ein Warnbildschirm, nicht die Seite.

Fehler 500. Der Server antwortet, aber mit einem internen Fehler. Die Seite existiert, der Server ist eingeschaltet, aber der Code ist beim Verarbeiten der Anfrage auf ein Problem gestoßen. Der Nutzer sieht eine Fehlermeldung statt des erwarteten Inhalts.

Netzwerkproblem. Die Verbindung zwischen dem Gerät des Nutzers und dem Server ist an irgendeinem Zwischenpunkt unterbrochen. Das kann die Internetverbindung des Nutzers sein, ein Netzanbieter oder ein Punkt der Infrastruktur zwischen beiden Seiten.

Sie oberflächlich zu unterscheiden hilft zu wissen, wo man anfängt:

- Wenn die Seite gar nichts lädt und es keine Warnung gibt → möglicherweise abgestürzter Server oder DNS, das nicht auflöst.
- Wenn der Browser einen ausdrücklichen Sicherheitshinweis zeigt → abgelaufenes Zertifikat oder HTTP ohne Verschlüsselung.
- Wenn die Seite mit einer Fehlermeldung im Inhalt lädt → wahrscheinlich Fehler 500, ein Problem des Codes.
- Wenn nur du nicht zugreifen kannst, andere aber schon → möglicherweise ein Netzwerkproblem oder ein lokales Cache-Problem.

Das zu wissen, heißt nicht, es direkt lösen zu können. Es heißt, es präzise an denjenigen kommunizieren zu können, der es kann.

> ***Nützliche Fragen, wenn eine Website nicht lädt***
>
> *Wenn das System nicht verfügbar ist, helfen diese Fragen, das Problem einzugrenzen, bevor man denjenigen ruft, der es lösen kann:*
>
> - *Tritt das Problem bei allen auf oder nur bei mir?*
> - *Löst die Domain auf? Zeigt der Name auf irgendeine Adresse?*
> - *Antwortet der Server? Kommt überhaupt etwas an, und sei es ein Fehler?*
> - *Welcher Fehlercode liegt vor — 404, 500 — oder gibt es gar keine Antwort?*
> - *Schlägt die Produktion fehl oder nur eine Testumgebung?*
> - *Gibt es Logs des Fehlers?*
> - *Gab es eine kürzliche Änderung — ein Deployment, eine Konfigurationsänderung, eine Domain-Verlängerung?*

Es gibt einen Fall, der es wert ist, erwähnt zu werden, weil er häufig und verwirrend ist: Das System funktioniert für manche Menschen und für andere nicht. Das bedeutet fast nie, dass der Server abgestürzt ist. Meist deutet es auf ein Problem der DNS-Propagierung hin — wenn eine Änderung an der Domain-Konfiguration vorgenommen wurde, braucht diese Änderung zwischen Minuten und Stunden, um zu allen DNS-Servern der Welt zu gelangen. Während dieser Zeit fragen verschiedene Nutzer verschiedene DNS-Server ab und bekommen verschiedene Antworten. Manche sehen die neue Seite; andere sehen weiter die alte oder sehen nichts. Es ist ein erwartetes Verhalten, kein Fehler, der dringend zu lösen wäre.

Wie du es in einem Meeting hören wirst

In einem Meeting mit einem technischen Team oder mit einem Anbieter werden Sätze verwendet, die voraussetzen, dass alle verstehen, wovon die Rede ist. Das sind die häufigsten zu diesem Thema, was sie in der Praxis bedeuten und was du fragen kannst.

„Sieht nach einem DNS-Problem aus." Der Name der Domain löst nicht korrekt auf — er lässt sich nicht in die IP-Adresse des Servers übersetzen. Das kann eine fehlerhafte Konfiguration sein oder eine kürzliche Änderung, die noch nicht vollständig propagiert ist.

Nützliche Frage: „Tritt es bei allen Nutzern auf oder nur bei manchen?" Wenn bei allen, ist es wahrscheinlich ein Konfigurationsfehler. Wenn nur bei manchen, ist es höchstwahrscheinlich eine laufende Propagierung — ein Prozess, der Stunden dauert und sich von selbst löst.

Nicht voreilig annehmen: dass der Server abgestürzt ist. Das DNS und der Server sind verschiedene Teile. Der Server kann perfekt funktionieren, während das DNS fehlschlägt.

„Der Server antwortet mit 500." Der Server hat die Anfrage empfangen, ist aber beim Verarbeiten auf einen Fehler gestoßen. Das Problem liegt im Code oder in der Konfiguration des Systems, nicht in der Anfrage des Nutzers und nicht im Netz.

Nützliche Frage: „Gibt es Logs des Fehlers?" Die Logs sind die Aufzeichnung dessen, was das System getan hat, bevor es fehlschlug. Ein 500 hinterlässt fast immer eine Spur, die die Ursache identifiziert. Ohne Logs wird die Diagnose viel schwieriger.

Nicht voreilig annehmen: dass es ein Problem des Internets oder des Geräts des Nutzers ist. Ein 500 bedeutet, dass der Server tatsächlich geantwortet hat — nur eben mit einem internen Fehler.

„Das Zertifikat ist abgelaufen." Das HTTPS-Zertifikat des Servers ist abgelaufen. Der Browser erkennt, dass die Verbindung nicht mehr als sicher garantiert werden kann, und blockiert den Zugang standardmäßig. Das System kann von innen perfekt funktionieren, aber kein Nutzer kann hinein.

Nützliche Frage: „Wie lange dauert die Verlängerung?" In den meisten Fällen ist es ein schneller Prozess. Was wichtig zu wissen ist, ist die geschätzte Zeit bis zur Lösung und ob es bereits betroffene Nutzer gibt, die man informieren muss.

Nicht voreilig annehmen: dass es ein Problem des Codes oder der Infrastruktur ist. Es ist ein administrativer Ablauf, ähnlich wie das Ablaufen eines Dokuments. Er sagt nichts über die Qualität des Systems aus.

„Es ist in der Produktion abgestürzt." Das System, das die echten Nutzer verwenden, ist nicht verfügbar. Es ist keine Test- oder Entwicklungsumgebung: Es ist die, die den echten Traffic und die echten Daten hat. Dieser Satz signalisiert Dringlichkeit.

Nützliche Frage: „Wird schon an der Lösung gearbeitet? Wie ist die geschätzte Zeit?" Diese zwei Fragen geben den nötigen Kontext, um zu wissen, ob man den Nutzern etwas kommunizieren oder still abwarten muss.

Nicht voreilig annehmen: dass das Problem alle gleichermaßen betrifft. Manchmal ist der Ausfall teilweise — er betrifft nur eine Region, ein Gerät oder eine bestimmte Funktion.

Was du in diesem Kapitel gelernt hast

- Eine URL ist die Adresse, die eine Ressource im Internet identifiziert. Um zu ihr zu gelangen, fragt der Browser das DNS ab, das den Namen der Domain in eine IP-Adresse übersetzt.
- Ein Server ist ein immer eingeschalteter Computer, der Anfragen empfängt und Antworten zurückgibt. Das Hosting ist der Dienst, der diesen Server vermietet.
- HTTP ist das Protokoll, das festlegt, wie Browser und Server kommunizieren. HTTPS ist seine verschlüsselte Version, verpflichtend für jedes System, das sensible Daten verarbeitet.
- Die Statuscodes sind die Sprache, die der Server verwendet, um anzuzeigen, was passiert ist: 200 ist Erfolg, 404 ist Ressource nicht gefunden, 500 ist interner Serverfehler.
- „Die App ist abgestürzt" hat spezifische technische Ursachen: abgestürzter Server, DNS, das nicht auflöst, abgelaufenes Zertifikat, Fehler im Code. Sie zu kennen, erlaubt es, das Problem einzugrenzen.

Was als Nächstes kommt

Kapitel 3 geht auf die beiden Akteure dieser Unterhaltung ein, die du gerade verstanden hast: den Client und den Server. Du weißt schon, wie die Information zwischen ihnen reist. Was als Nächstes kommt, ist zu verstehen, was jeder von ihnen bei diesem Austausch tut — und warum diese Aufteilung die grundlegende Struktur jeder Anwendung ist.

Diese Leseprobe endet hier.

Das vollständige Buch geht weiter mit der kompletten Landkarte:

Client, Server, Datenbanken, Code,

APIs, Sicherheit, künstliche Intelligenz und Deployment.

Die vollständige Ausgabe umfasst

22 Kapitel, praktische Anhänge,

ein technisches Glossar und die Versionen PDF + EPUB.