디지털 에디션 · 한국어

새로운 기술 크리에이터.

무료 미리보기

서론 · 전체 목차 · 2장

지은이

Hernán Capucci

독립 출판 · 2026

이 무료 미리보기에는 책의 서론과 전체 목차, 그리고 대표적인 한 장이 담겨 있어요. 전체 판으로 넘어가기 전에 이 책의 어조와 접근 방식, 그리고 설명하는 방식을 미리 느껴 볼 수 있어요.

이 미리보기에 담긴 내용

- 서론 전체 (서론 — 이 책이 존재하는 이유).
- 책의 전체 목차.
- 2장 전체: 인터넷은 어떻게 작동하나.
- 이어지는 내용 안내.

전체 목차

서론 — 이 책이 존재하는 이유

어느 날 AI에게 무언가를 만들어 달라고 요청해요. 어떤 기능을 위한 코드, 두 서비스를 연결하는 자동화, 구체적인 문제를 해결하는 코드 조각 같은 것을요. 돌려

받은 결과물은 잘 작동하는 것처럼 보여요. 그래서 그걸 사용해요. 그런데 그 과정 어딘가에서, 어쩌면 소리 내어 묻기는 망설여지는 질문 하나가 떠올라요. 이게 제대로 된 건지 내가 어떻게 알 수 있을까요?

그 질문은 사소하지 않아요. 그것은 스스로의 판단력으로 움직이는 것과, 아무도 오류를 발견하지 않기를 바라며 움직이는 것 사이의 차이예요.

이 책은 그 질문에 답할 수 있게 하려고 존재해요.

이해 없는 접근

몇 년 전만 해도 소프트웨어를 만들려면 특정한 기술 교육이 필요했어요. 프로그래밍 언어를 배우고, 자료 구조를 이해하고, 개발 환경을 설정하고, 이미 많은 걸 안다고 전제하는 문서를 읽어야 했지요. 높고 실제적인 장벽이었어요.

그 후 생성형 인공지능 도구가 등장했어요. 갑자기 누구나 자연어로 명령을 작성해서 작동하는 코드, 설정이 완료된 자동화, 외부 서비스에 연결된 시스템을 받을 수 있게 됐어요. 기술 장벽이 크게 낮아졌지요.

하지만 낮아진 건 기술 장벽이지, 개념의 장벽이 아니었어요.

어떤 도구가 대신 코드를 만들어 준다고 해서, 그것이 무엇을 만들었는지 이해한다는 뜻은 아니에요. 어떤 플랫폼이 클릭 한 번으로 애플리케이션을 배포해 준다고 해서, 배포가 무엇인지, 무엇이 잘못될 수 있는지, 문제가 생겼을 때 어떻게 되돌리는지 이해한다는 뜻은 아니에요. AI가 오류를 설명해 준다고 해서, 그것이 왜 일어났는지, 어떻게 다시 일어나지 않게 할지 안다는 뜻도 아니지요.

접근성은 좋아졌어요. 하지만 그 아래에 있는 시스템에 대한 이해는 많은 경우 같은 속도로 따라오지 못했어요.

만드는 것과 이해하는 것의 차이

오늘날에는 기술 교육 없이 디지털 제품을 만드는 사람이 그 어느 때보다 많아요. 앱, 자동화, 서비스 간 통합, 콘텐츠 플랫폼, 디지털화된 내부 프로세스 같은 것들이요.

이러한 대중화는 실재하고 가치 있는 일이에요. 문제는 무언가가 기대한 대로 작동하지 않을 때, 또는 만들어 놓은 시스템에 대해 결정을 내려야 할 때 나타나요.

AI를 사용해 온라인 상점을 만든 어떤 창업자는 왜「데이터베이스가 다운됐는지」알지 못해요. 고객 정보가 보호되고 있는지도 몰라요. 고용한 개발자가 넘겨준 작업물이 잘 만들어진 것인지 아닌지도 몰라요. 그걸 알아내려면 무엇을 물어봐야 하는지도 모르고요.

AI로 자기 업무 프로세스를 자동화한 어떤 전문가는 자신이 만든 것의 각 부분이 정확히 무슨 일을 하는지 이해하지 못해요. 어제 잘 작동했다는 이유로 결과를 받아들여요. 작동이 멈추면 어디부터 살펴봐야 할지 파악할 방법이 없어요.

외부 기술 팀과 함께 일하는 어떤 창업자는 **API**나 저장소, 시스템 아키텍처 이야기를 들으면 고개를 끄덕여요. 하지만 이해하지는 못해요. 그리고 무엇을 물어야 할지 모르니 묻지도 않아요.

이 모든 경우에서 문제는 지능이나 능력의 부족이 아니에요. 디지털 생태계에서 스스로의 판단력으로 움직이게 해 주는 어휘와 개념적 틀의 부족이에요.

출발점으로서의 AI

인공지능은 이 책을 시급하게 만든 요인이에요. 말하자면 알람 시계 같은 거예요.

하지만 이 책이 향하는 곳은 인공지능을 이야기하는 것이 아니에요. AI가 있든 없든, 모든 디지털 프로젝트 아래에 존재하는 시스템을 이야기하는 것이에요.

API는 AI 개념이 아니에요. 수십 년 전부터 존재했어요. 코드 저장소 역시 새로운 것이 아니에요. 개발 환경과 프로덕션 환경의 차이는 어떤 언어 모델이 발명한 것이 아니에요. 사용자 인증, 관계형 데이터베이스, 배포의 생애 주기 — 이 모든 것은 생성형 AI 이전에도 존재했고, 이후에도 계속 존재할 거예요.

AI로 달라진 것은, 이제 이런 것들이 예전에는 접하지 않던 사람들의 손에 닿게 됐다는 점이에요. 그리고 이해 없는 접근성은 새로운 의존성을 만들어 내요.

도구에 대한 의존성이 있어요. 도구가 바뀌거나, 오류가 나거나, 예상치 못한 결과를 내놓으면, 그것을 평가할 방법이 없어요. 이해하고 있는 개발자에 대한 의존성이 있어요. 그 사람이 바뀌거나, 요금을 올리거나, 사라지면, 그 사람 없이는 진행할 수 없어요. 더 많이 안다고 말하는 사람에 대한 의존성이 있어요. 자기만의 틀이 없으면, 진짜 지식과 빈 약속을 구별할 수 없어요.

AI는 문을 하나 열었어요. 이 책은 그 안에 무엇이 있는지 이해할 어휘를 줘요.

빠져 있는 어휘

처음부터 짚고 넘어갈 만한, 흔한 오해가 하나 있어요.

이 책이 다루는 문제는 프로그래밍을 할 줄 모른다는 것이 아니에요. 프로그래밍을 못 하는 것은 문제가 아니에요. 디지털 세계에서 대단히 유능한 아주 많은 사람들이 프로그래밍을 할 줄 모르고, 배울 필요도 없어요.

중요한 것은 코드를 쓸 줄 아는 것이 아니에요. 중요한 것은 코드가 무엇을 하는지 이해할 만큼의 판단력을 가지고 코드를 읽을 줄 아는 거예요. 중요한 것은 정보를 어떻게 저장할지 결정할 수 있도록 데이터베이스가 무엇인지 이해하는 거예요. 개발자가 넘겨준 것을 검토할 수 있도록 저장소가 무엇인지 아는 거예요. AI에게 엔드포인트를 정확하게 만들어 달라고 요청할 수 있도록 엔드포인트가 무엇인지 아는 거예요. 실수로 운영 중인 무언가를 망가뜨리지 않도록 프로덕션 환경이 무엇인지 아는 거예요.

그 어떤 것도 코드 한 줄 쓸 필요가 없어요.

이런 구체적인 상황을 떠올려 보세요. 개발자가「스테이징 환경에서 마이그레이션이 실패했어요」라고 말해요. 맥락이 없으면 그 말은 소음일 뿐이에요. 이 책의 어휘가 있으면,「마이그레이션」이 데이터베이스 구조의 변경이고,「스테이징」이 프로덕션 직전의 테스트 환경이며, 그 문제가 아마도 아직 사용자에게는 영향을 주지 않았지만 계속 진행하기 전에 해결해야 한다는 것을 이해하게 돼요. 이 차이는 프로그래밍을 요구하지 않아요. 개념을 이해하기를 요구해요.

정작 필요한 것은 기술 어휘예요. 올바른 질문을 하고, 답변을 해석하고, 제안을 평가하고, 경고 신호를 알아차리고, 판단력을 가지고 결정을 내리게 해 주는 어휘요. 그 어휘는 도구를 사용한다고 얻어지지 않아요. 그 뒤에 있는 개념을 이해할 때 얻어져요.

이것이 바로 이 책이 쌓아 가는 거예요. 개념 하나하나씩, 맨 처음부터, 보편적인 예시와 함께, 어떤 사전 지식도 전제하지 않고요.

이 책은 누구를 위한 책인가

이 책은 기술 교육 없이 디지털 결과물을 만들면서, 더 나은 판단력으로 그 일을 해내고 싶은 사람들을 위해 쓰였어요.

여기에는 AI의 도움으로 첫 디지털 제품을 만들고 있지만 자신이 만드는 것을 완전히 이해하지는 못하는 사람이 포함돼요. 외부 기술 팀과 일하면서 매번의 대화가 번역 작업이 되지 않고 그들과 이야기할 수 있기를 바라는 사람도요. 일상 업무에서 자동화 도구를 쓰면서 그것이 실제로 무슨 일을 하는지 이해하고 싶은 사람도요. 아이디어가 있고, 그것을 개발하는 데 시간과 돈을 들이기 전에 기술적으로 평가해 봐야 하는 사람도요.

업종이나 업무 분야는 중요하지 않아요. 나이나 기술 분야의 정규 교육 수준도 중요하지 않아요.

정작 중요한 것은, 일이나 프로젝트에서 컴퓨터를 능숙하게 사용한다는 점이에요. AI 도구를 적어도 하나쯤은 써 봤다는 점이에요. 그리고 자신이 몸담고 있는 디지털 생태계를 더 잘 이해하고 싶어 한다는 점이에요.

출발점은 코드 저장소를 한 번도 열어 본 적이 없는 상태일 수 있어요. 개발 환경을 한 번도 설정해 본 적이 없는 상태일 수도 있고요. 서버와 클라우드의 차이가 명확하지 않은 상태일 수도 있어요. 그 어떤 출발점도 장애물이 아니에요. 이 책은 바로 거기서부터 시작해요.

이 책에 없는 것

계속하기 전에, 이 책이 하지 않는 일을 분명히 해 두는 게 좋겠어요.

이 책은 프로그래밍을 가르치지 않아요. 어떤 프로그래밍 언어나 특정 도구의 설명서도 아니에요. 그걸 찾고 있다면, 그 목적에는 더 낫고 더 전문적인 자료들이 있어요.

이 책은 AI가 모든 것을 해낸다고 주장하지 않아요. AI는 그렇게 하지 못해요. 실제 능력과 실제 한계를 함께 가지고 있어요. 이 책은 그 두 가지를 정직하게 보여줘요.

이 책은 정해진 기간 안에 프로젝트가 완성된다고 약속하지 않아요. 기술적인 과정에는 그 나름의 복잡함과 그 나름의 시간이 있어요. 그것을 축소해서 말하는 건 거짓말이 될 거예요.

이 책은 특정 도구나 플랫폼, 서비스에 의존하지 않아요. 구체적인 도구를 언급할 때는 예시로 드는 거예요. 여기서 가르치는 개념들은 어떤 스택이나 환경을 쓰더라도 똑같이 적용돼요.

이 책은 특정한 성공 사례나 따라야 할 모범으로서의 실제 프로젝트를 이야기하지 않아요. 예시들은 의도적으로 일반적이고 보편적이에요.

이 책은 쉽지 않은 것을 두고「쉽다」고 말하지 않아요. 이 책의 어떤 개념들은 단순해요. 어떤 것들은 주의를 기울이고 다시 읽어 봐야 해요. 무언가에 실제로 복잡함이 있을 때, 이 책은 그것을 짚어 줘요.

이 책에 있는 것

이 책을 다 읽고 나면, 지금은 어쩌면 이름만 알고 있는 개념들을 구체적으로 이해하게 될 거예요.

애플리케이션의 내부가 어떻게 되어 있는지 이해하게 될 거예요. 화면에 보이는 것을 다루는 부분은 어디이고, 로직과 데이터를 처리하는 부분은 어디이며, 그 두 부분이 어떻게 소통하는지를요. 그렇다고 혼자서 애플리케이션을 만들 수 있게 되는 건 아니지만, 누군가 그것을 설명하거나 무언가 문제가 생겼을 때 무슨 이야기를 하는지는 이해하게 될 거예요.

코드 저장소를 열어서 그 구조를 읽을 수 있게 될 거예요. 어떤 폴더들이 있고, 프로젝트의 각 부분이 무슨 일을 하며, 변경 이력에 무엇이 적혀 있는지를요. 그 정보는 어떤 저장소에나 있고, 누군가 넘겨준 것을 이해하는 데 그것을 활용할 수 있다는 사실을 지금은 어쩌면 모르고 있을 거예요.

AI가 쓸모 있는 결과를 내놓도록 충분한 맥락을 담은 프롬프트를 쓸 수 있게 될 거예요. 모호한 지시와 정확한 지시의 차이는 재능이 아니에요. 도구가 제대로 일하려면 어떤 정보가 필요한지 아는 거예요.

AI나 개발자가 만들어 낸 것을 구체적인 질문으로 검토할 수 있게 될 거예요. 이 부분은 정확히 무슨 일을 하나요? 이 필드가 비어 있는 채로 들어오면 어떻게 되나요? 민감한 정보에 접근하는 부분이 있나요? 이런 질문들은 프로그래밍을 알아야 할 필요가 없어요. 관련된 개념들을 이해하기를 요구하지요.

이 책의 용어집을 작업 도구로 활용할 수 있게 될 거예요. 모르는 용어가 나오면 정의를 펼쳐 보고, 그것을 이해한 다음, 전보다 더 또렷한 상태로 대화나 문서로 돌아오는 식으로요.

그리고 어떤 답변이 — 사람의 답변이든 도구의 답변이든 — 지금의 지식으로는 평가할 수 없는 것일 때를 알아차릴 수 있게 될 거예요. 그것이 늘 정답을 안다는

뜻은 아니에요. 정작 해야 할 질문이 방금 던진 질문보다 더 낫다는 것을, 언제 그런지를 아는 것을 뜻해요.

이 책의 구성

이 책은 여섯 개의 부와 이 서론 격의 장, 실용적인 부록, 그리고 기술 용어집으로 구성돼 있어요.

1부는 기초를 다뤄요. 소프트웨어가 무엇인지, 인터넷이 어떻게 작동하는지, 클라이언트-서버 모델이 무엇인지, 데이터베이스가 무엇이고 코드가 무엇인지요. 디지털 생태계에서 가장 기본이 되는 개념들이에요. 뒤따르는 각 부는 이 개념들을 이미 안다고 전제해요.

2부는 애플리케이션의 아키텍처로 들어가요. 프론트엔드, 백엔드, API, 데이터베이스를 깊이 있게 다루고, 인증과 세션도 다뤄요. 현대의 모든 디지털 제품이 가진 내부 구조예요.

3부는 프로젝트를 현실 세계에서 돌아가게 만드는 도구와 프로세스를 다뤄요. **Git**을 이용한 버전 관리, 명령어 터미널, 개발 환경, 배포, 그리고 기본적인 보안이요.

4부는 AI와 똑똑하게 일하는 방법에 할애돼 있어요. AI가 무엇을 할 수 있는지, 어디서 체계적으로 실패하는지, 효과적인 프롬프트를 어떻게 구성하는지, 그것이 내놓는 결과를 어떻게 검토하는지, 그리고 과정에 대한 통제력을 잃지 않으면서 모델과 에이전트, 자동화를 어떻게 활용하는지요.

5부는 구체적인 무언가를 만드는 맥락에서 이 모든 것을 한데 모아요. 시작하기 전에 어떻게 준비하는지, 기술 팀과 어떻게 소통하는지, 아이디어에서 실제로 작동하는 제품으로 어떻게 나아가는지요.

6부는 실용적인 부록이에요. 빠르게 참고할 수 있는 체크리스트, 기본적인 터미널과 Git 명령어, 재사용할 수 있는 프롬프트 템플릿, 그리고 각 영역을 더 깊이 파고들 수 있는 자료들이요.

이 책의 맨 끝에는 200개가 넘는 용어를 쉬운 말로, 예시와 함께, 전문 용어 없이 설명한 기술 용어집이 있어요. 장식용 부록이 아니에요. 이 책의 필수적인 한 부분이에요.

부의 순서는 임의로 정한 것이 아니에요. 각 부는 앞선 부 위에 쌓여요. 시스템과 도구를 다루는 부들은 1부의 기초가 필요해요. AI와 제작을 다루는 부들은 아키텍처와 도구가 필요하고요. 순서대로 읽기로 한다면, 그게 바로 그 이유예요.

이 책을 활용하는 법

이 책을 읽는 방법은 두 가지가 있고, 둘 다 괜찮은 방법이에요.

첫 번째는 처음부터 끝까지 죽 이어서 읽는 거예요. 디지털 생태계와 사전 접점이 거의 또는 전혀 없는 분에게 권하는 방법이에요. 이렇게 읽으면 이해가 차곡차곡 쌓여요. 각 장이 다음 장을 위한 바탕을 마련해 주거든요.

두 번째는 참고서처럼 읽는 거예요. 누군가 API를 언급했는데 그게 뭔지 이해되지 않는다면, 8장을 펼쳐요. 첫 배포를 앞두고 있는데 무엇이 관련되는지 모른다면, 14장을 펼쳐요. 보안 회의를 앞두고 인증이 무엇인지 이해해야 한다면, 10장을 펼쳐요. 각 장은 독립적으로 읽을 수 있지만, 상호 참조가 무엇을 먼저 읽어 두면 좋은지 알려 줘요.

용어집도 같은 방식으로 작동해요. 어떤 기술 용어가 한 장에서 처음 등장할 때는 용어집 참조와 함께 굵은 글씨로 표시돼요. 이 책의 디지털 버전에서는 그 참조가 살아 있는 하이퍼링크예요. 클릭 한 번으로 정의로 바로 갔다가 다시 장으로 돌아올 수 있어요. 인쇄 버전에서는 그 참조가 페이지 번호를 알려 줘요.

예시에 대한 참고 하나. 이 책 곳곳에서 같은 유형의 상황들을 만나게 될 거예요. 온라인 상점, 예약 관리 앱, 강의 플랫폼 같은 것들이요. 어떤 업종에 대해서도 사전 지식이 필요하지 않아서 고른, 일반적인 상황들이에요. 개념들은 다른 어떤 종류의 프로젝트에도 똑같이 적용돼요.

도구에 대한 참고 하나. 어떤 도구를 예시로 언급할 때는, 말 그대로 예시일 뿐이에요. 이 책이 가르치는 개념들은 특정 플랫폼에 의존하지 않아요. 오늘 쓰는 도구가 내일 바뀌더라도, 개념들은 여전히 유효해요.

스스로의 판단력으로 만들기

디지털 도구를 이해하지 못한 채 쓰는 사람과, 자신이 무엇을 하는지 알면서 쓰는 사람 사이에는 차이가 있어요.

그 차이는 프로그래밍을 아는 데 있지 않아요. 올바른 질문을 할 어휘를 갖추는 데 있어요. 답변을 평가할 개념적 틀을 갖추는 데 있고요. 정확히 왜인지는 몰라도 무

언가 잘못됐을 때를 알아차리는 능력에 있어요. 무슨 일이 벌어지는지 이해하려고 남에게 전적으로 의존하지 않고 스스로 결정을 내리는 자율성에 있어요.

이것이 바로 이 책이 만들어 가는 인물상이에요. 새로운 기술 크리에이터요. AI를 사용하고, 기술 팀과 일하고, 디지털 결과물을 만드는 — 그리고 그것을 스스로의 판단력으로 해내는 사람이요.

이 사람은 프로그래머가 아니에요. 프로그래머가 되려고 하지도 않아요. 하지만 눈을 감은 채 움직이거나, 평가하지도 못하면서 결과를 받아들이거나, 누군가 모든 것을 설명해 주기만을 기다리지도 않아요.

이 사람은 자신이 일하는 시스템을 이해해요. 무엇을 물어야 하는지, 받은 것을 어떻게 평가하는지, 언제 무언가 잘못됐다고 인정해야 하는지를 알아요. 그 이해는 코드를 쓸 필요가 없어요. 자신이 사용하는 것이 어떻게 작동하는지 이해하기를 요구하지요.

이 장에서 배운 것

- AI는 기술적 실행에 대한 접근을 대중화했지만, 디지털 생태계에 대한 이해까지 대중화하지는 못했어요.
- 프로그래밍을 못 하는 것은 문제가 아니에요. 최소한의 기술 어휘가 없는 것이 문제예요.
- 이 책은 프로그래밍을 가르치지 않아요. 판단력을 가지고 만들고, 기술 팀과 이야기하고, AI가 내놓는 결과를 검토하게 해 주는 개념들을 가르쳐요.
- 이 책은 여섯 개의 부와 실용적인 부록, 그리고 200개가 넘는 용어를 담은 용어집으로 이루어져 있어요. 처음부터 죽 읽을 수도 있고, 참고서로 활용할 수도 있어요.
- 디지털 버전에서는 용어집의 용어들이 각 장에서 살아 있는 하이퍼링크예요.

다음 내용

1장은 맨 처음부터 시작해요. 소프트웨어가 무엇인지, 그리고 직접 소프트웨어를 짤 일이 결코 없더라도 왜 그것을 이해하는 게 중요한지를요. 1부의 첫 블록이자, 이후에 오는 모든 것의 바탕이에요.

2장 — 인터넷은 어떻게 작동하나

브라우저에 주소를 입력하고 Enter를 눌러요. 1초도 안 되어 페이지가 나타나요. 텍스트, 이미지, 최신 데이터까지요.

그 모든 게 어디서 온 걸까요? 어떻게 화면까지 도착한 걸까요? 왜 가끔은 도착하지 않을까요?

이 질문들에 답하는 데 프로그래밍 지식은 필요 없어요. 필요한 건 어떤 기기든 세계의 어떤 서버와도 연결해 주는 인프라를 이해하는 거예요. 그 이해가 바로 무언가가 왜 작동하는지, 그리고 왜 가끔은 작동하지 않는지를 진단하는 토대가 돼요.

요청이 거치는 경로

브라우저에 주소를 입력할 때, 여러분이 쓰고 있는 그것을 **URL**이라고 불러요. Uniform Resource Locator, 즉 통합 자원 식별자예요. 인터넷에서 특정 자원, 그러니까 페이지, 이미지, 파일, 데이터 하나를 가리키는 주소예요.

브라우저는 그 자원을 가진 서버를 찾아야 해요. 그런데 서버는 사람이 읽을 수 있는 이름으로 식별되지 않아요. 숫자로 식별돼요. 서버마다 **IP 주소**가 있어요. 네트워크 안에서 그 서버를 유일하게 지목하는 숫자 문자열인데, 우편 주소가 도시 안의 한 건물을 지목하는 것과 비슷해요.

문제는 여러분이 숫자를 입력하지 않는다는 거예요. `www.enciclopedia.org`나 `noticias.ejemplo.com`을 입력하죠. 브라우저가 그에 해당하는 서버를 찾으려면 그 이름을 IP 주소로 번역해야 해요. 그 과정을 **DNS** 해석이라고 불러요. Domain Name System, 즉 도메인 이름 시스템이에요.

DNS는 인터넷의 전화번호부처럼 작동해요. 브라우저가 DNS 서버에 이렇게 물어요. 「`noticias.ejemplo.com`의 IP는 뭐야?」 DNS 서버는 그 숫자로 답해요. 이제 브라우저는 실제 주소를 갖게 됐어요.

그 주소를 가지고 브라우저는 서버에 요청을 보내요. 그 요청은 네트워크를 통해, 케이블과 광섬유와 무선 연결을 거쳐, 요청을 기다리는 서버까지 이동해요. 서버는 그것을 받아서 처리할 것을 처리하고 응답을 돌려줘요. 그 응답은 다시 브라우저까지 되돌아와요. 브라우저는 그 내용을 해석해서 화면에 보여줘요.

이 모든 과정, 그러니까 DNS 조회, 가는 여정, 처리, 돌아오는 응답이 정상적인 조건에서는 몇 분의 1초 만에 일어나요.

그림 1. 주소를 입력하면 브라우저는 마법의 구름으로 가는 게 아니라, 번역과 요청과 응답의 사슬을 시작해요.

짚고 넘어갈 만한 최적화가 하나 있어요. 바로 캐시예요. 브라우저는 같은 사이트를 방문할 때마다 매번 DNS를 조회하지는 않아요. 응답, 즉 이름을 IP로 번역한 결과를 한동안 저장해 둬요. 운영체제도 마찬가지로 그렇게 해요. 그래서 반복 방문이 더 빨라져요. 하지만 이건 서버가 최근에 IP 주소를 바꿨다면 여러분의 기기가 여전히 옛 주소를 쓰고 있을 수도 있다는 뜻이기도 해요. 브라우저 캐시를 비우거나 만료될 때까지 기다리면 그 문제가 해결돼요.

주소를 입력하면 브라우저는 마법의 구름으로 가는 게 아니라, 번역과 요청과 응답의 사슬을 시작해요.

도메인이란 무엇인가

도메인은 인터넷에서 어떤 사이트나 서비스를 식별하는, 사람이 읽을 수 있는 이름이에요. `enciclopedia.org`, `noticias.ejemplo.com`, `miapp.io` 같은 게 도메인이에요.

도메인은 저절로 존재하지 않아요. 누군가가 등록하는 거예요. 도메인 등록을 전문으로 하는 회사들이 있는데, 이들을 등록 대행업체라고 불러요. 도메인은 보통 1년 단위로 일정 기간 동안 임대해요. 갱신하지 않으면 도메인은 만료되고, 다른 사람이 등록할 수 있도록 다시 풀려요.

도메인의 맨 끝부분, 즉 `.com`, `.org`, `.io`, `.ar` 같은 부분을 최상위 도메인 또는 TLD(Top Level Domain)라고 불러요. 조직의 유형이나 출신 국가를 나타내요. 이 구분은 역사적인 가치와 포지셔닝의 가치가 있지만, 기술적으로는 시스템의 작동을 바꾸지 않아요.

헷갈릴 수 있는 부분은 도메인과 서버 사이의 관계예요. 도메인은 서버가 아니에요. 서버를 가리키는 이름표예요. 하나의 서버에 여러 도메인이 연결될 수 있어요. 하나의 도메인이 시간대나 접속하는 지역에 따라 서로 다른 서버로 안내할 수도 있어요. DNS는 이름과 주소 사이의 그 관계를 최신 상태로 유지하는 시스템이에요.

도메인이 제대로 설정되지 않았을 때, 또는 만료됐을 때 DNS는 주소를 해석할 수 없어요. 브라우저는 어느 서버로 가야 할지 몰라요. 페이지를 담고 있는 서버가 완벽하게 작동하고 있어도 페이지는 열리지 않아요.

서버란 무엇이고, 호스팅이란 무엇인가

서버는 요청을 받아서 응답을 돌려주도록 설계된 컴퓨터예요. 근본적으로는 여느 컴퓨터와 다르지 않아요. 프로세서, 메모리, 저장 장치, 네트워크 연결을 갖추고 있어요.

실용적인 차이는 서버가 항상 켜져 있고, 높은 가용성으로 인터넷에 연결돼 있으며, 여러 요청을 동시에 처리하도록 설정돼 있다는 점이에요. 화면도 키보드도 없어요. 누구의 책상 위에도 있지 않아요. 서버는 데이터 센터, 즉 많은 서버를 지속적이고 안전하게 계속 돌리도록 설계된 시설에서 살아가요.

호스팅은 그런 서버의 공간과 자원을 여러분에게 임대해 주는 서비스예요. 인터넷에 무언가를, 그러니까 페이지나 애플리케이션이나 파일을「게시」한다는 건, 그 콘텐츠를 호스팅 업체의 서버에 올려 두고, 그곳을 가리키도록 도메인을 설정하고, 요청하는 사람에게 서버가 그것을 전달하도록 두는 일이에요.

호스팅에는 여러 형태가 있어요. 가장 기본적인 형태에서는 다른 사이트들과 서버를 공유해요. 더 싸지만 자원이 제한적이에요. 더 고급 형태에서는 전용 가상 서버나 물리 서버를 갖게 돼요. 더 비싸지만 통제력과 용량이 더 커요. 클라우드 모델에서는 자원이 수요에 따라 동적으로 할당돼요.

디지털한 무언가를 만드는 사람에게 호스팅이 무엇인지 이해한다는 건, 여러분이 만든 시스템이 어딘가 물리적인 장소에 살고 있다는 것, 그 장소에는 용량의 한계가 있다는 것, 그리고 그 장소가 고장 나면, 또는 도메인이 그곳을 제대로 가리키지 못하면, 시스템이 아무리 완벽하게 만들어졌어도 접근할 수 없게 된다는 것을 이해하는 거예요.

유용한 구분이 하나 있어요. 웹 서버는 HTTP 요청을 받아서 콘텐츠를 돌려주는 구성 요소예요. 애플리케이션 서버는 시스템의 로직을 실행하는 것으로, 3장에서 더 자세히 다룰 거예요. 단순한 시스템에서는 두 기능이 같은 기계에서 돌아가는 경우가 많아요. 더 큰 시스템에서는 분리돼 있어요. 지금 중요한 건「서버」가 하나의 단일한 덩어리가 아니라는 점이에요. 요청에 응답하기 위해 함께 일하는 여러 조각의 모음이에요.

HTTP와 HTTPS: 웹의 언어

브라우저가 요청을 보내고 서버가 응답을 돌려줄 때, 둘은 같은 언어로 말해야 해요. 그 언어를 프로토콜이라고 불러요. 프로토콜은 메시지를 어떻게 형식화하고 전송하고 해석할지를 정의하는 규칙의 모음이에요.

웹의 프로토콜은 **HTTP**예요. HyperText Transfer Protocol이죠. 요청과 응답의 구조를 정의해요. 어떤 정보가 먼저 오는지, 콘텐츠의 유형을 어떻게 표시하는지, 작업의 결과를 어떻게 전달하는지 같은 걸요.

기본적인 HTTP 요청에는 다음이 들어가요. 메서드(무엇을 하고 싶은지, 즉 정보를 가져오기, 데이터를 보내기, 무언가를 삭제하기), 자원의 주소, 그리고 누가 묻고 있는지와 어떤 유형의 응답을 받아들이는지에 대한 추가 정보를 담은 헤더예요. HTTP 응답에는 다음이 들어가요. 상태 코드(작동했는지 아닌지), 콘텐츠에 대한 정보를 담은 헤더, 그리고 본문, 즉 콘텐츠 그 자체예요.

HTTPS는 HTTP의 보안 버전이에요.「S」는「Secure」의 S예요. 기술적인 차이는 HTTPS에서는 모든 통신이 암호화된 채로 이동한다는 점이에요. 브라우저가 서버에 보내는 데이터와 서버가 돌려주는 데이터를, 중간에서 연결을 가로채는 누군가가 읽을 수 없어요.

그 암호화를 가능하게 하는 건 **SSL/TLS** 인증서예요. 서버가 자기가 말하는 그 서버가 맞다는 걸 증명하려고 브라우저에 제시하는 디지털 파일인데, 통신의 암호화를 활성화해요. 최신 브라우저는 연결이 HTTPS를 쓸 때 주소 표시줄에 자물쇠를 보여줘요.

만드는 사람에게 왜 중요할까요. 사용자 데이터, 그러니까 비밀번호, 개인정보, 결제 정보를 다루는 시스템이라면 예외 없이 HTTPS를 써야 해요. HTTPS가 없으면 그 데이터는 평문으로 네트워크를 이동하고 가로채일 수 있어요. HTTP를 쓰면 브라우저가「안전하지 않은 사이트」라는 경고를 띄울 수 있는데, 이건 사용자를 떠나게 만들고 시스템에 대한 신뢰를 떨어뜨려요.

> 핵심 개념: 프로토콜 두 당사자가 어떻게 소통할지를 정의하는 규칙의 모음이에요. *HTTP*는 브라우저와 서버가 웹에서 정보를 주고받으려고 쓰는 프로토콜이에요. *HTTPS*는 그것의 암호화된 버전이에요.

서버가 응답하는 것

서버가 요청을 받을 때마다 숫자 하나를 포함한 응답을 돌려줘요. 바로 상태 코드예요. 그 숫자가 요청에 무슨 일이 일어났는지를 나타내요.

코드는 범주별로 묶여요. 2로 시작하는 것들은 성공을 뜻해요. 3으로 시작하는 것들은 리다이렉션이에요. 4로 시작하는 것들은 클라이언트 쪽 오류를 나타내요. 요청에 뭔가 잘못됐다는 거죠. 5로 시작하는 것들은 서버 쪽 오류를 나타내요. 서버가 요청은 받았지만 제대로 처리하지 못했다는 거예요.

시스템을 다루는 사람에게 가장 중요한 세 가지는 이거예요.

200 — OK. 요청이 제대로 처리됐고 서버가 기대한 콘텐츠를 돌려줬어요. 정상적인 결과예요. 페이지가 보인다면 서버가 200으로 응답한 거예요.

404 — Not Found. 서버가 요청은 받았지만, 여러분이 요청한 자원이 그 서버에 존재하지 않아요. 여러분이 쓴 주소가 이용할 수 있는 어떤 파일, 페이지, 데이터에도 해당하지 않아요. 서버의 오류가 아니에요. 여러분이 찾던 게 거기 없는 거예요.

500 — Internal Server Error. 서버가 요청을 받아 처리하려 했는데, 자기 자신의 작동에서 뭔가가 실패했어요. 문제는 여러분이 요청한 것에 있지 않아요. 서버가 그 요청을 어떻게 다뤘는지에 있어요. 코드나 서버 설정의 오류예요.

코드	이름	무슨 일이 일어났나	일상적인 예시
200	OK	서버가 여러분이 요청한 것을 찾아서 전달했어요.	페이지를 열면 문제없이 전부 불러와져요.
301 / 302	리다이렉션	자원이 다른 주소에 있어요. 브라우저가 자동으로 거기로 데려가요.	옛 URL이 사이트의 새 도메인으로 보내 줘요.

코드	이름	무슨 일이 일어났나	일상적인 예시
400	Bad Request	요청이 잘못된 형식으로 도착해서 서버가 해석할 수 없었어요.	필수 항목이 빠진 채로 양식을 제출해요.
401	Unauthorized	유효한 인증 정보를 제시하지 않았어요. 서버는 여러분이 누구인지 몰라요.	로그인하지 않은 채 비공개 콘텐츠를 보려고 해요.
403	Forbidden	인증 정보는 유효하지만, 그것에 대한 권한이 없어요.	계정에는 들어왔지만, 그 영역이 여러분의 역할에 해당하지 않아요.
404	Not Found	여러분이 요청한 게 그 서버에 존재하지 않아요.	URL을 잘못 썼거나 이제는 존재하지 않아요.
500	Internal Server Error	서버가 요청은 받았지만 처리에 실패했어요.	결제를 끝내려는데「예상치 못한 오류」가 나타나요.
503	Service Unavailable	서버가 지금 응답할 수 없어요. 과부하 상태이거나 점검 중이에요.	대규모 세일이나 출시 도중에 사이트가 다운돼요.

모든 코드를 외울 필요는 없어요. 중요한 건 패턴을 알아보는 거예요. 2xx는 대체로 성공, 3xx는 리다이렉션, 4xx는 요청 쪽의 문제, 5xx는 서버 쪽의 문제를 나타내요.

왜 앱이「다운됐나」

「페이지가 안 열려요」,「시스템이 다운됐어요」,「작동을 안 해요」. 무언가가 이용할 수 없게 될 때는 대개 구체적인 기술적 원인이 있어요. 어디서 고장이 날 수 있는지 그 지점들을 이해하면, 문제를 해결해 줄 사람을 부르기 전에 문제 범위를 좁힐 수 있어요.

가장 흔한 원인들을, 어디서 일어나는지에 따라 정리하면 이래요.

서버가 다운됐어요. 서버가 응답을 멈췄어요. 호스팅 서비스에 문제가 생겼거나, 업데이트 때문에 서버가 재시작됐거나, 시스템이 자원(메모리, CPU)을 다 써서 작동을 멈춘 걸 수 있어요. 이 경우 브라우저는 어떤 종류의 응답도 받지 못해요.

도메인이 해석되지 않아요. DNS가 도메인 이름을 IP 주소로 번역하지 못해요. 도메인이 만료됐거나, DNS 설정이 잘못 바뀌었거나, DNS 서버에 문제가 있어서일 수 있어요. 결과는 앞의 경우와 비슷해요. 브라우저는 어디로 가야 할지 몰라요.

인증서가 만료됐어요. 서버의 HTTPS 인증서가 만료되면, 브라우저는 연결을 차단하고 보안 경고를 띄워요. 페이지는 기술적으로는 존재하고 서버도 작동하지만, 브라우저가 연결을 거부하며 사용자를 보호하는 거예요. 눈에 보이는 결과는 페이지가 아니라 경고 화면이에요.

500 오류. 서버가 응답은 하지만 내부 오류와 함께 응답해요. 페이지는 존재하고 서버도 켜져 있지만, 요청을 처리하다가 코드가 문제를 만난 거예요. 사용자는 기대한 콘텐츠 대신 오류 메시지를 보게 돼요.

네트워크 문제. 사용자의 기기와 서버 사이의 연결이 중간 어딘가에서 끊겼어요. 사용자의 인터넷 연결일 수도, 네트워크 사업자일 수도, 양쪽 사이 인프라의 어떤 지점일 수도 있어요.

이것들을 겉으로 구분해 보면 어디서부터 시작할지 아는 데 도움이 돼요.

- 페이지가 아무것도 안 열리고 경고도 없다면 → 서버가 다운됐거나 DNS가 해석되지 않는 것일 수 있어요.
- 브라우저가 명시적인 보안 경고를 보여준다면 → 인증서 만료거나 암호화되지 않은 HTTP예요.
- 페이지가 열리는데 콘텐츠에 오류 메시지가 있다면 → 아마 500 오류, 코드의 문제예요.

- 다른 사람은 되는데 여러분만 접근이 안 된다면 → 네트워크 문제거나 로컬 캐시 문제일 수 있어요.

이걸 안다고 해서 직접 해결할 수 있다는 뜻은 아니에요. 해결할 수 있는 사람에게 정확하게 전달할 수 있다는 뜻이에요.

> 웹이 열리지 않을 때 유용한 질문
>
> 시스템을 이용할 수 없을 때, 해결할 수 있는 사람을 부르기 전에 다음 질문들이 문제 범위를 좁히는 데 도움이 돼요.
>
> - 이 문제가 모두에게 생기나요, 아니면 저에게만 생기나요?
> - 도메인이 해석되나요? 이름이 어떤 주소를 가리키고 있나요?
> - 서버가 응답하나요? 오류라도, 뭔가 도착하긴 하나요?
> - 어떤 오류 코드가 있나요 — *404, 500* — 아니면 아예 아무 응답도 없나요?
> - 프로덕션이 실패하고 있나요, 아니면 테스트 환경만 그런가요?
> - 오류의 로그가 있나요?
> - 최근에 어떤 변경이 있었나요 — 배포, 설정 변경, 도메인 갱신 같은 거요?

자주 일어나고 헷갈려서 짚어 둘 만한 경우가 하나 있어요. 시스템이 어떤 사람들에게는 작동하고 다른 사람들에게는 작동하지 않는 경우예요. 이건 서버가 다운됐다는 뜻인 경우는 거의 없어요. 대개는 DNS 전파의 문제를 나타내요. 도메인 설정을 변경하면 그 변경이 세계의 모든 DNS 서버에 도달하기까지 몇 분에서 몇 시간이 걸려요. 그동안 서로 다른 사용자가 서로 다른 DNS 서버에 조회하고 서로 다른 응답을 받아요. 어떤 사람은 새 사이트를 보고, 어떤 사람은 여전히 옛 사이트를 보거나 아무것도 못 봐요. 이건 예상된 동작이지 급하게 해결해야 할 오류가 아니에요.

회의에서 듣게 될 말

기술 팀이나 협력 업체와의 회의에서는, 모두가 무슨 이야기를 하는지 안다고 전제하는 표현들을 써요. 이 주제에서 가장 흔한 표현들, 그것이 실제로 뜻하는 것, 그리고 여러분이 무엇을 물어볼 수 있는지를 소개할게요.

「**DNS** 문제 같아 보여요.」도메인 이름이 제대로 해석되지 않고 있어요. 서버의 IP 주소로 번역되지 않는 거예요. 잘못된 설정일 수도 있고, 아직 완전히 전파되지 않은 최근 변경일 수도 있어요.

유용한 질문: 「모든 사용자에게 생기나요, 아니면 일부에게만 생기나요?」모두에게 생긴다면 아마 설정 오류예요. 일부에게만 생긴다면 진행 중인 전파일 가능성이 가장 커요. 몇 시간이 걸리고 저절로 해결되는 과정이에요.

가정하지 말 것: 서버가 다운됐다고 가정하지 마세요. DNS와 서버는 서로 다른 조각이에요. DNS가 실패하는 동안에도 서버는 완벽하게 작동하고 있을 수 있어요.

「서버가 **500**으로 응답해요.」서버가 요청은 받았지만 처리하다가 오류를 만났어요. 문제는 코드나 시스템 설정에 있지, 사용자의 요청이나 네트워크에 있지 않아요.

유용한 질문: 「오류의 로그가 있나요?」로그는 시스템이 실패하기 전에 무엇을 했는지에 대한 기록이에요. 500은 거의 항상 원인을 알려주는 흔적을 남겨요. 로그가 없으면 진단이 훨씬 더 어려워져요.

가정하지 말 것: 인터넷이나 사용자 기기의 문제라고 가정하지 마세요. 500은 서버가 실제로 응답했다는 뜻이에요. 다만 내부 오류와 함께 응답한 거예요.

「인증서가 만료됐어요.」서버의 HTTPS 인증서가 만료됐어요. 브라우저는 연결이 더는 안전하다고 보장될 수 없다고 판단하고 기본적으로 접근을 차단해요. 시스템은 안에서 완벽하게 작동하고 있을 수 있지만, 어떤 사용자도 들어올 수 없어요.

유용한 질문: 「갱신에 얼마나 걸리나요?」대부분의 경우 빠른 과정이에요. 알아야 할 건 예상 해결 시간과, 이미 영향받은 사용자가 있어서 안내해야 하는지예요.

가정하지 말 것: 코드나 인프라의 문제라고 가정하지 마세요. 이건 문서가 만료되는 것과 비슷한 행정적 만료예요. 시스템의 품질에 대해서는 아무것도 말해 주지 않아요.

「프로덕션이 다운됐어요.」실제 사용자가 쓰는 시스템을 이용할 수 없어요. 테스트 환경도 개발 환경도 아니에요. 실제 트래픽과 실제 데이터가 있는 환경이에요. 이 표현은 긴급함을 나타내요.

유용한 질문: 「이미 해결 작업을 하고 있나요? 예상 시간은 얼마나 되나요?」이 두 질문이 사용자에게 무언가를 안내해야 할지 아니면 조용히 기다려야 할지 판단하는 데 필요한 맥락을 줘요.

가정하지 말 것: 문제가 모두에게 똑같이 영향을 준다고 가정하지 마세요. 다운이 부분적일 때도 있어요. 특정 지역, 특정 기기, 특정 기능에만 영향을 줄 수 있어요.

이 장에서 배운 것

- URL은 인터넷에서 자원을 식별하는 주소예요. 그 자원에 도달하려고 브라우저는 DNS를 조회하고, DNS는 도메인 이름을 IP 주소로 번역해요.
- 서버는 요청을 받아서 응답을 돌려주는, 항상 켜져 있는 컴퓨터예요. 호스팅은 그 서버를 임대해 주는 서비스예요.
- HTTP는 브라우저와 서버가 어떻게 소통할지를 정의하는 프로토콜이에요. HTTPS는 그것의 암호화된 버전으로, 민감한 데이터를 다루는 어떤 시스템에서든 필수예요.
- 상태 코드는 무슨 일이 일어났는지 나타내려고 서버가 쓰는 언어예요. 200은 성공, 404는 자원을 찾을 수 없음, 500은 서버 내부 오류예요.
- 「앱이 다운됐다」에는 구체적인 기술적 원인이 있어요. 서버 다운, DNS 미해석, 인증서 만료, 코드 오류 같은 거요. 이것들을 알면 문제 범위를 좁힐 수 있어요.

다음 내용

3장은 여러분이 방금 이해한 그 대화의 두 주역, 즉 클라이언트와 서버로 들어가요. 정보가 그 둘 사이를 어떻게 이동하는지는 이제 알아요. 다음으로 이해할 건 그 주고받음에서 각자가 무엇을 하는지, 그리고 왜 그 구분이 모든 애플리케이션의 근본 구조인지예요.

이 미리보기는 여기까지예요.

전체 책은 이어서 전체 지도를 펼쳐요.
클라이언트, 서버, 데이터베이스, 코드,
API, 보안, 인공지능, 그리고 배포까지.

전체 판에는
22개 장, 실용 부록,
기술 용어집, 그리고 PDF + EPUB 버전이 담겨 있어요.