DİJİTAL SÜRÜM · TÜRKÇE

Yeni *kreatör* teknik.

ÜCRETSİZ ÖRNEK

Giriş · Tam içindekiler · 2. Bölüm

YAZAR

Hernán Capucci

Bağımsız Sürüm · 2026

Bu ücretsiz örnek kitabın girişini, tam içindekiler listesini ve temsili bir bölümünü içerir. Böylece tam sürüme geçmeden önce kitabın tonunu, yaklaşımını ve öğretme biçimini değerlendirebilirsin.

Bu örnek ne içeriyor

- Tam giriş (Bölüm 0).
- Kitabın eksiksiz içindekiler listesi.
- 2. Bölümün tamamı: *İnternet nasıl çalışır*.
- Devamlılıkla ilgili notlar.

Tam içindekiler

Bölüm 0 — Bu kitap neden var

Bir gün yapay zekâdan sana bir şey üretmesini istersin — bir özellik için kod, iki servisi birbirine bağlayan bir otomasyon, somut bir sorunu çözen bir parça. Sana verdiği şey çalışıyor gibi

görünür. Onu kullanırsın. Ve bu sürecin bir noktasında, belki yüksek sesle sormaya cesaret edemediğin bir soru belirir: bunun doğru olup olmadığını nasıl bilirim?

Bu soru önemsiz değil. Kendi muhakemenle hareket etmek ile kimsenin hatayı bulmamasını umarak hareket etmek arasındaki farktır.

Bu kitap, ona yanıt verebilmen için var.

Anlamadan erişim

Birkaç yıl önce **software** inşa etmek belirli bir teknik eğitim gerektiriyordu. Bir programlama dili öğrenmek, veri yapılarını anlamak, geliştirme ortamları kurmak, zaten çok şey bildiğini varsayan belgeleri okumak. Yüksek ve gerçek bir engeldi.

Sonra **üretken yapay zekâ** araçları geldi. Aniden, herkes doğal dilde bir komut yazıp çalışan kod, yapılandırılmış bir otomasyon, dış servislere bağlı bir sistem alabilir oldu. Teknik engel önemli ölçüde düştü.

Ama düşen teknik engeldi, kavramsal engel değil.

Bir aracın senin yerine kod üretmesi, ne ürettiğini anladığın anlamına gelmez. Bir platformun uygulamanı tek tıkla dağıtması, dağıtımın ne olduğunu, neyin bozulabileceğini veya bir şey ters gittiğinde nasıl geri döneceğini anladığın anlamına gelmez. Yapay zekânın sana bir hatayı açıklaması, neden olduğunu ya da tekrar olmasını nasıl önleyeceğini bildiğin anlamına gelmez.

Erişim iyileşti. Altta yatan **sistemin** anlaşılması ise birçok durumda aynı ritmi tutturamadı.

Yaratmak anlamakla aynı şey değil

Bugün, teknik eğitimi olmadan dijital ürünler inşa eden insan sayısı her zamankinden fazla. Uygulamalar, otomasyonlar, servisler arası entegrasyonlar, içerik platformları, dijitalleştirilmiş iç süreçler.

Bu demokratikleşme gerçek ve değerli. Sorun, bir şey beklenildiği gibi çalışmadığında ya da inşa edilen sistem hakkında karar vermek gerektiğinde ortaya çıkar.

Çevrimiçi mağazasını inşa etmek için yapay zekâ kullanan bir girişimci, "veritabanının neden çöktüğünü" bilmez. Müşterilerinin bilgilerinin korunup korunmadığını bilmez. Tuttuğu geliştiricinin teslim ettiği işin iyi yapılıp yapılmadığını bilmez. Bunu öğrenmek için ona hangi soruları soracağını bilmez.

Süreçlerini yapay zekâ ile otomatikleştiren bir profesyonel, kurduğu şeyin her parçasının tam olarak ne yaptığını anlamaz. Sonucu, dün çalıştığı için kabul eder. Çalışmayı bıraktığında, nereye bakacağını anlamak için elinde araç yoktur.

Dış bir teknik ekiple çalışan bir kurucu, kendisine **API'lerden**, **depolardan** ya da sistem mimarisinden bahsedildiğinde başını sallar. Ama anlamaz. Ve sormaz, çünkü ne soracağını bilmez.

Tüm bu durumlarda sorun, zekâ ya da yetenek eksikliği değil. Dijital ekosistemde kendi muhakemenle hareket etmeni sağlayan kelime dağarcığının ve kavramsal çerçevenin eksikliğidir.

Başlangıç noktası olarak yapay zekâ

Bu kitabı acil kılan şey yapay zekâ oldu. O, çalar saattir.

Ama kitabın asıl amacı yapay zekâdan bahsetmek değil. Herhangi bir dijital projenin altında bulunan sistemlerden bahsetmek — yapay zekâ olsun ya da olmasın.

API bir yapay zekâ kavramı değil. Onlarca yıldır var. Bir kod **deposu** da yeni değil. Bir geliştirme ortamı ile bir üretim ortamı arasındaki farkı hiçbir dil modeli icat etmedi. Kullanıcı kimlik doğrulaması, ilişkisel veritabanları, bir **deploy** yaşam döngüsü — bütün bunlar üretken yapay zekâdan önce de vardı ve sonra da var olmaya devam edecek.

Yapay zekâ ile değişen şey, bu şeylerin artık daha önce onlara dokunmayan insanların erişiminde olması. Ve anlamadan gelen bu erişilebilirlik yeni bağımlılıklar yaratıyor.

Araca bağımlılık: değişirse, bozulursa, beklenmedik sonuçlar verirse, onları değerlendirecek bir yolun yok. Gerçekten anlayan geliştiriciye bağımlılık: değişirse, ücretlerini artırırsa, ortadan kaybolursa, onsuz devam edemezsin. Daha çok bildiğini söyleyene bağımlılık: kendine ait bir çerçeven olmadan, gerçek bilgiyi boş vaatlerden ayıramazsın.

Yapay zekâ bir kapı açtı. Bu kitap, içeride ne olduğunu anlamak için kelime dağarcığını veriyor.

Eksik olan kelime dağarcığı

Baştan açıklığa kavuşturmaya değer, sık görülen bir kafa karışıklığı var.

Bu kitabın ele aldığı sorun, programlamayı bilmemen değil. Programlamayı bilmemek sorun değil. Dijital dünyada çok etkili olan pek çok insan programlamayı bilmiyor ve öğrenmesine de gerek yok.

Önemli olan kod yazmayı bilmek değil. Önemli olan, ne yaptığını anlayacak kadar muhakemeyle kodu okumayı bilmek. Önemli olan, bilgiyi nasıl saklayacağına karar verebilmek için bir **veritabanının** ne olduğunu anlamak. Bir geliştiricinin sana teslim ettiğini gözden geçirebilmek için deponun ne olduğunu bilmek. Yapay zekâdan bir tane hassasiyetle inşa etmesini isteyebilmek için bir **uç noktanın** ne olduğunu bilmek. Yanlışlıkla canlıda bir şeyi bozmamak için üretim ortamının ne olduğunu bilmek.

Bunların hiçbiri tek bir satır kod yazmayı gerektirmez.

Şu somut duruma bak: geliştirici sana "staging ortamında migration başarısız oldu" diyor. Bağlam olmadan bu cümle gürültüdür. Bu kitabın kelime dağarcığıyla, "migration"ın veritabanı yapısında bir değişiklik olduğunu, "staging"in üretim öncesi test ortamı olduğunu ve sorunun muhtemelen kullanıcıları henüz etkilemediğini — ama devam etmeden önce çözülmesi gerektiğini anlarsın. Bu fark programlamayı gerektirmez. Kavramları anlamayı gerektirir.

Gerektirdiği şey ise teknik kelime dağarcığı. Doğru soruları sormayı, yanıtları yorumlamayı, önerileri değerlendirmeyi, uyarı işaretlerini fark etmeyi ve muhakemeyle karar vermeyi sağlayan kelime dağarcığı. Bu kelime dağarcığı araçları kullanarak edinilmez. Arkalarındaki kavramları anlayarak edinilir.

Bu kitabın inşa ettiği şey budur. Kavram kavram, sıfırdan, evrensel örneklerle ve hiçbir ön bilgi varsaymadan.

Bu kitap kimin için

Bu kitap, teknik eğitimi olmadan dijital şeyler inşa eden ve bunu daha fazla muhakemeyle yapmak isteyen insanlar için yazıldı.

Buna şunlar dahil: yapay zekânın yardımıyla ilk dijital ürününü kuran ve inşa ettiği şeyi tam olarak anlamayan kişi. Dış bir teknik ekiple çalışan ve her konuşmanın bir çeviri çabası olmadan onlarla konuşabilmek isteyen kişi. Günlük işinde otomasyon araçları kullanan ve gerçekte ne yaptığını anlamak isteyen kişi. Bir fikri olan ve onu geliştirmeye zaman ve para harcamadan önce teknik olarak değerlendirmesi gereken kişi.

Sektör ya da çalışma alanı önemli değil. Yaş ya da teknolojide aldığın formel eğitim düzeyi de önemli değil.

Önemli olan, işinde ya da projende bilgisayarı akıcı bir şekilde kullanman. En azından bir yapay zekâ aracını denemiş olman. Ve içinde hareket ettiğin **dijital ekosistemi** daha iyi anlamak istemen.

Başlangıç noktan, hiç bir kod deposu açmamış olman olabilir. Hiç bir geliştirme ortamı kurmamış olman. Bir sunucu ile bir bulut arasındaki farkın ne olduğundan emin olmaman. Bu başlangıç noktalarının hiçbiri bir engel değil. Bu kitap tam oradan başlıyor.

Bu kitapta bulamayacakların

Devam etmeden önce, bu kitabın yapmadığı şeyler konusunda açık olmakta fayda var.

Programlamayı öğretmez. Hiçbir programlama dilinin ya da belirli bir aracın el kitabı değil. Aradığın buysa, bunun için daha iyi ve daha uzmanlaşmış kaynaklar var.

Yapay zekânın her şeyi yaptığını iddia etmez. Yapmıyor. Gerçek yetenekleri ve gerçek sınırları var. Bu kitap her iki şeyi de dürüstçe gösterir.

Belirli bir sürede projenin hazır olacağını vaat etmez. Teknik süreçlerin kendine özgü bir karmaşıklığı ve kendi zamanı vardır. Bunları küçümsemek yalan söylemek olurdu.

Hiçbir belirli araca, platforma ya da servise bağlı değildir. Somut bir araçtan bahsedildiğinde, bu örnek amaçlıdır. Öğretilen kavramlar, kullandığın herhangi bir stack ya da ortamda aynı şekilde geçerlidir.

Belirli başarı hikâyelerinden ya da takip edilecek model olarak gerçek projelerden bahsetmez. Örnekler tasarım gereği geneldir ve evrenseldir.

Kolay olmadığında “kolay” demez. Bu kitaptaki bazı kavramlar basittir. Diğerleri dikkat ve yeniden okumayı gerektirir. Bir şeyin gerçek bir karmaşıklığı olduğunda, kitap bunu belirtir.

Bu kitapta bulacakların

Bu kitabı bitirdiğinde, bugün belki sadece adını bildiğin kavramları somut bir şekilde anlamış olacaksın.

Bir uygulamanın içeriden ne olduğunu anlayacaksın: hangi parçanın ekranda gördüğünü yönettiğini, hangi parçanın mantığı ve verileri işlediğini ve bu iki parçanın nasıl iletişim kurduğunu. Bu, onu tek başına inşa etmeyi bilmeni sağlamaz ama sana anlatıldığında ya da bir şey bozulduğunda neden bahsettiklerini anlamanı sağlar.

Bir kod deposunu açıp yapısını okuyabileceksin: hangi klasörlerin olduğunu, projenin her parçasının ne yaptığını, değişiklik geçmişinin ne söylediğini. Bu bilgi herhangi bir depoda mevcut ve bugün belki birinin sana teslim ettiğini anlamak için onu kullanabileceğini bilmiyorsun.

Yapay zekânın işe yarar bir şey üretmesi için yeterli bağlamla bir **istem** yazabileceksin. Belirsiz bir komut ile hassas bir komut arasındaki fark yetenek değil: aracın iyi çalışmak için hangi bilgiye ihtiyaç duyduğunu bilmektir.

Yapay zekânın ya da bir geliştiricinin ürettiğini somut sorularla gözden geçirebileceksin: bu parça tam olarak ne yapıyor? Bu alan boş gelirse ne olur? Hassas bilgilere erişen herhangi bir bölüm var mı? Bu sorular programlamayı bilmeyi gerektirmez. Söz konusu kavramları anlamayı gerektirir.

Bu kitabın sözlüğünü bir çalışma aracı olarak kullanabileceksin: bilinmeyen bir terim çıktığında bir tanımı açmak, onu anlamak ve konuşmaya ya da belgeye öncekinden daha fazla netlikle dönmek.

Ve bir yanıtın — bir kişiden ya da bir araçtan — mevcut bilginle değerlendirilemeyeceğini ne zaman fark edeceğini bileceksin. Bu her zaman doğru yanıtı bilmek anlamına gelmez. Sorman gereken sorunun, sorduğun sorudan daha iyi olduğunu ne zaman bileceğin anlamına gelir.

Nasıl düzenlendi

Kitap, bu giriş bölümüne ek olarak altı kısımdan, pratik eklerden ve teknik bir sözlükten oluşuyor.

Kısım I temelleri kapsar: software nedir, internet nasıl çalışır, istemci-sunucu modeli nedir, veritabanı nedir ve kod nedir. Bunlar dijital ekosistemin en temel kavramlarıdır. Sonraki her kısım bunları bilinen kabul eder.

Kısım II bir uygulamanın mimarisine girer: **frontend**, **backend**, API'ler, derinlemesine veritabanları, kimlik doğrulama ve oturumlar. Herhangi bir modern dijital ürünün iç yapısıdır.

Kısım III bir projeyi gerçek dünyada işler hâle getiren araçları ve süreçleri kapsar: **Git** ile sürüm kontrolü, komut terminali, geliştirme ortamları, deploy ve temel güvenlik.

Kısım IV yapay zekâ ile akıllıca çalışmaya ayrılmıştır: ne yapabilir, sistematik olarak nerede başarısız olur, etkili bir istem nasıl oluşturulur, ürettiği şey nasıl gözden geçirilir ve modeller, **ajanlar** ve otomasyonlar süreç üzerindeki kontrolü kaybetmeden nasıl kullanılır.

Kısım V her şeyi somut bir şey inşa etme bağlamında bir araya getirir: başlamadan önce nasıl hazırlanılır, teknik ekiplerle nasıl iletişim kurulur, bir fikirden çalışan bir ürüne nasıl geçilir.

Kısım VI pratik eklerdir: hızlı başvuru kontrol listeleri, temel terminal ve Git komutları, yeniden kullanılabilir istem şablonları ve her alanda derinleşmek için kaynaklar.

Kitabın sonunda, 200'den fazla terimin açık bir dille, örneklerle ve jargon olmadan açıklandığı bir **teknik sözlük** var. Dekoratif bir ek değil. Kitabın ayrılmaz bir parçasıdır.

Kısımların sırası rastgele değil. Her biri bir öncekinin üzerine inşa edilir. Sistemler ve araçlar kısımları, Kısım I'in temellerine ihtiyaç duyar. Yapay zekâ ve inşa kısımları, mimariye ve araçlara ihtiyaç duyar. Sırayla okumayı seçersen, sebebi budur.

Nasıl kullanılır

Bu kitabı okumanın iki yolu var ve ikisi de geçerli.

Birincisi baştan sona, kesintisiz okumak. Dijital ekosistemle önceden az ya da hiç teması olmayan kişiler için önerilen yol budur. Bu güzergâh, anlamayı birikimli olarak inşa eder: her bölüm bir sonrakinin zeminini hazırlar.

İkincisi bir başvuru kitabı gibi. Biri sana bir API'den bahsetti ve ne olduğunu anlamadın: Bölüm 8'i açarsın. İlk deploy'unu yapacaksın ve neyi içerdiğini bilmiyorsun: Bölüm 14'ü açarsın. Güvenlik üzerine bir toplantıdan önce **kimlik doğrulamanın** ne olduğunu anlaman gerekiyor: Bölüm 10'u açarsın. Çapraz referanslar önceden neyin okunmasının iyi olacağını belirtse de, her bölüm bağımsız olarak okunabilir.

Sözlük de aynı şekilde çalışır. Bir teknik terim bir bölümde ilk kez geçtiğinde, sözlüğe bir referansla **kalın** yazılır. Kitabın dijital sürümünde bu referans aktif bir köprüdür: doğrudan tanıma gidip tek bir tıkla bölüme dönebilirsin. Basılı sürümde ise referans sayfa numarasını gösterir.

Örnekler hakkında bir not: kitap boyunca aynı tür bağlamlarla karşılaşacaksın — bir çevrimiçi mağaza, bir randevu yönetimi uygulaması, bir kurs platformu. Bunlar, hiçbir sektör hakkında ön bilgi gerektirmedikleri için seçilmiş genel durumlardır. Kavramlar, başka herhangi bir proje türünde de tıpatıp aynı şekilde geçerlidir.

Araçlar hakkında bir not: bir araç örnek olarak anıldığında, o sadece bir örnektir. Bu kitabın öğrettiği kavramlar hiçbir belirli platforma bağlı değildir. Bugün kullandığın araç yarın değişirse, kavramlar geçerli olmaya devam eder.

Kendi muhakemenle yaratmak

Dijital araçları anlamadan kullanan biri ile onları ne yaptığını bilerek kullanan biri arasında bir fark vardır.

Bu fark programlamayı bilmekte değil. Doğru soruları sormak için kelime dağarcığına sahip olmakta. Yanıtları değerlendirmek için kavramsal çerçeveye. Tam olarak nedenini bilmesen bile bir şeyin ne zaman yanlış olduğunu fark etme yetisine. Ne olup bittiğini anlamak için tamamen başkalarına bağımlı olmadan karar verme özerkliğine.

Bu kitabın inşa ettiği profil budur: yeni teknik kreatör. Yapay zekâ kullanan, teknik ekiplerle çalışan, dijital şeyler inşa eden — ve bunu kendi muhakemesiyle yapan biri.

Bir programcı değil. Olmaya da çalışmıyor. Ama aynı zamanda körü körüne hareket etmiyor, sonuçları değerlendiremeden kabul etmiyor ve her şeyi başkasının açıklamasına bağımlı değil.

İçinde çalıştığı sistemi anlar. Ne soracağını, aldığı şeyi nasıl değerlendireceğini ve bir şeyin ne zaman yolunda olmadığını ne zaman fark edeceğini bilir. Bu anlayış kod yazmayı gerektirmez. Kullandığın şeyin nasıl çalıştığını anlamayı gerektirir.

Bu bölümde öğrendiklerin

- Yapay zekâ teknik uygulamaya erişimi demokratikleştirdi, ama dijital ekosistemin anlaşılmasını değil.

- Programlamayı bilmemek sorun değil. Asıl sorun, asgari teknik kelime dağarcığının eksikliğidir.
- Bu kitap programlamayı öğretmez. Muhakemeyle yaratmayı, teknik ekiplerle konuşmayı ve yapay zekânın ürettiğini gözden geçirmeyi sağlayan kavramları öğretir.
- Kitap altı kısımdan, pratik eklerden ve 200'den fazla terimlik bir sözlükten oluşur. Baştan sona okunabilir ya da başvuru olarak kullanılabilir.
- Dijital sürümde, sözlük terimleri her bölümde aktif köprülerdir.

Sırada ne var

Bölüm 1 en baştan başlıyor: software nedir ve hiç yazmayacak olsan bile onu anlamak neden önemlidir. Kısım I'in ilk bloğu ve sonrasında gelen her şeyin temelidir.

Bölüm 2 — İnternet nasıl çalışır

Tarayıcıya bir adres yazıp Enter'a basıyorsun. Bir saniyeden kısa sürede bir sayfa beliriyor: metin, görseller, güncel veriler.

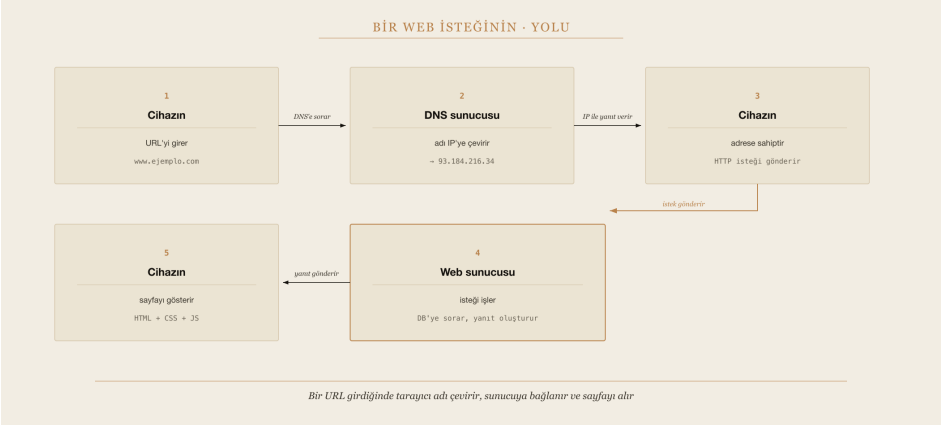
Bütün bunlar nereden geldi? Ekranına nasıl ulaştı? Neden bazen ulaşmıyor?

Bu sorulara yanıt vermek için programlamayı bilmek gerekmez. Herhangi bir cihazı dünyadaki herhangi bir sunucuya bağlayan altyapıyı anlamak gerekir. Bu anlayış, bir şeyin neden çalıştığını — ve neden bazen çalışmadığını — teşhis edebilmenin temelidir.

Bir isteğin güzergâhı

Tarayıcıya bir adres yazdığında, yazdığın şeye **URL** denir — Uniform Resource Locator, yani tekdüze kaynak konumlayıcısı. İnternetteki belirli bir kaynağı tanımlayan adrestir: bir sayfa, bir görsel, bir dosya, bir veri.

Tarayıcının o kaynağa sahip olan sunucuyu bulması gerekir. Ama sunucular okunabilir adlarıyla tanımlanmaz: sayılarla tanımlanır. Her sunucunun bir **IP adresi** vardır — bir sayı dizisi, onu ağda benzersiz biçimde konumlar; tıpkı bir posta adresinin bir binayı bir şehirde konumlaması gibi.



Şekil 1. Bir adres yazdığında, tarayıcı sihirli bir buluta gitmez: bir çeviriler, istekler ve yanıtlar zinciri başlatır.

Sorun şu ki sen sayı yazmıyorsun. `www.ansiklopedi.org` ya da `haberler.ornek.com` yazıyorsun. Tarayıcının ilgili sunucuyu bulabilmesi için bu adı bir IP adresine çevirmesi gerekir. Bu sürece **DNS** çözümlemesi denir — Domain Name System, yani alan adı sistemi.

DNS, internetin telefon rehberi gibi çalışır. Tarayıcı bir DNS sunucusuna danışır ve sorar: "`haberler.ornek.com` hangi IP'ye sahip?" DNS sunucusu sayıyla yanıt verir. Tarayıcı artık gerçek adrese sahiptir.

Bu adresle tarayıcı sunucuya bir istek gönderir. Bu istek ağ üzerinden — kablolarla, fiber optiklerle ve kablosuz bağlantılarla — onu bekleyen sunucuya kadar gider. Sunucu onu alır, işlemesi gerekeni işler ve bir yanıt döndürür. Bu yanıt tarayıcıya geri yolculuk eder. Tarayıcı içeriği yorumlar ve ekranda gösterir.

Bütün bu süreç — DNS sorgusu, gidiş yolculuğu, işleme, dönüş yanıtı — normal koşullarda saniyenin kesirlerinde gerçekleşir.

Anmaya değer bir iyileştirme var: **önbellek**. Tarayıcı, aynı siteyi her ziyaret ettiğinde DNS'e danışmaz. Yanıtı — adın IP'ye çevirisini — bir süreliğine saklar. İşletim sistemi de aynısını yapar. Bu, tekrarlanan ziyaretleri daha hızlı kılar. Ama aynı zamanda, sunucu yakın zamanda IP adresini değiştirdiyse, cihazının hâlâ eski adresi kullanıyor olabileceği anlamına gelir. Tarayıcının önbelleğini boşaltmak ya da süresinin dolmasını beklemek bu sorunu çözer.

Bir adres yazdığında, tarayıcı sihirli bir buluta gitmez: bir çeviriler, istekler ve yanıtlar zinciri başlatır.

Alan adı nedir

Alan adı, internetteki bir siteyi ya da servisi tanımlayan okunabilir addır. `ansiklopedi.org`, `haberler.ornek.com`, `uygulamam.io` birer alan adıdır.

Alan adları kendi başlarına var olmaz: biri onları kaydeder. Alan adı kaydetmekte uzmanlaşmış şirketler vardır — bunlara kayıt kuruluşu denir — ve alan adı, genellikle bir yıl gibi belirli süreler için kiralanır. Yenilenmezse, alan adının süresi dolar ve başka birinin kaydetmesi için kullanılabilir hale gelir.

Alan adının son kısmı — `.com`, `.org`, `.io`, `.ar` — üst düzey alan adı ya da TLD (Top Level Domain) olarak adlandırılır. Kuruluşun türünü ya da menşe ülkesini belirtir. Bu ayrımın tarihsel ve konumlandırma değeri vardır, ama teknik olarak sistemin işleyişini değiştirmez.

Kafa karıştırabilecek kısım, alan adı ile sunucu arasındaki ilişkidir. Bir alan adı, sunucu değildir: sunucuya işaret eden etikettir. Aynı sunucuya birden çok alan adı işaret edebilir. Aynı alan adı, günün saatine ya da erişilen dünya bölgesine göre farklı sunuculara yönlendirebilir. DNS, adlar ile adresler arasındaki bu ilişkiyi güncel tutan sistemdir.

Bir alan adı doğru yapılandırılmadığında — ya da süresi dolduğunda — DNS adresi çözemez. Tarayıcı hangi sunucuya gideceğini bilmez. Onu barındıran sunucu kusursuz çalışıyor olsa bile sayfa yüklenmez.

Sunucu nedir — ve barındırma nedir

Bir **sunucu**, istekleri almak ve yanıtları döndürmek için tasarlanmış bir bilgisayardır. Temelde herhangi bir bilgisayardan farklı değildir: işlemcisi, belleği, depolaması ve ağ bağlantısı vardır.

Pratikteki fark, bir sunucunun sürekli açık olması, internete yüksek erişilebilirlikle bağlı olması ve aynı anda birden çok isteği karşılamak üzere yapılandırılmış olmasıdır. Ekranı ya da klavyesi yoktur. Kimsenin masasında durmaz. Bir veri merkezinde yaşar — birçok sunucuyu sürekli ve güvenli biçimde çalışır tutmak için tasarlanmış bir tesiste.

Barındırma, o sunuculardaki alanı ve kaynakları sana kiralayan servistir. İnternette bir şeyi "yayımladığında" — bir sayfayı, bir uygulamayı, bir dosyayı — yaptığın şey, o içeriği bir barındırma şirketinin sunucusuna yerleştirmek, ona işaret etmesi için bir alan adı yapılandırmak ve sunucunun onu isteyene teslim etmesine izin vermektir.

Farklı barındırma biçimleri vardır. En temel biçimde, sunucuyu başka sitelerle paylaşırsın — daha ucuzdur ama sınırlı kaynaklarla. Daha gelişmiş biçimlerde, sana özel sanal ya da fiziksel bir sunucun olur — daha pahalıdır ama daha fazla denetim ve kapasiteyle. Bulut modelinde, kaynaklar talebe göre dinamik olarak atanır.

Dijital bir şey inşa eden biri için barındırmanın ne olduğunu anlamak, oluşturduğun sistemin fiziksel bir yerde yaşadığını, o yerin kapasite sınırları olduğunu ve o yer başarısız olursa — ya da alan adı ona doğru işaret etmezse — sistem kusursuz inşa edilmiş olsa bile erişilemez olduğunu anlamak demektir.

İşe yarar bir ayrım: web sunucusu, HTTP isteklerini alıp içerik döndüren bileşendir. Uygulama sunucusu ise sistemin mantığını yürüten sunucudur — 3. Bölüm'de daha ayrıntılı görülecek. Birçok basit sistemde, bu iki işlev aynı makinede çalışır. Daha büyük sistemlerde ayrıdırlar. Şimdilik önemli olan şu: "sunucu" tek ve yekpare bir şey değildir: istekleri yanıtlamak için birlikte çalışan parçaların bütünüdür.

HTTP ve HTTPS: web'in dili

Tarayıcı bir istek gönderdiğinde ve sunucu bir yanıt döndürdüğünde, ikisinin de aynı dili konuşması gerekir. Bu dile **protokol** denir. Bir protokol, mesajların nasıl biçimlendirileceğini, iletileceğini ve yorumlanacağını tanımlayan kurallar bütünüdür.

Web'in protokolü **HTTP**'dir — HyperText Transfer Protocol. İsteklerin ve yanıtların yapısını tanımlar: hangi bilgi önce gelir, içerik türü nasıl belirtilir, işlemin sonucu nasıl bildirilir.

Temel bir HTTP isteği şunları içerir: yöntem (ne yapmak istediğin — bilgi almak, veri göndermek, bir şey silmek), kaynağın adresi ve kimin sorduğuna ve ne tür bir yanıt kabul ettiğine dair ek bilgi içeren başlıklar. Bir HTTP yanıtı şunları içerir: durum kodu (işe yarayıp yaramadığı), içerik hakkında bilgi içeren başlıklar ve gövde — içeriğin kendisi.

HTTPS, HTTP'nin güvenli sürümüdür. "S" harfi "Secure"dan gelir. Teknik fark, HTTPS'de tüm iletişimin şifreli olarak yolculuk etmesidir: tarayıcının sunucuya gönderdiği ve sunucunun döndürdüğü veriler, yolda bağlantıyı kesen biri tarafından okunamaz.

Bu şifrelemeyi bir **SSL/TLS sertifikası** mümkün kılar — sunucunun, olduğunu söylediği kişi olduğunu kanıtlamak için tarayıcıya sunduğu ve iletişimin şifrelenmesini etkinleştiren dijital bir dosya. Modern tarayıcılar, bağlantı HTTPS kullandığında adres çubuğunda bir kilit gösterir.

İnşa eden biri için neden önemli olduğu: kullanıcı verilerini — parolaları, kişisel bilgileri, ödemeleri — işleyen herhangi bir sistem, istisnasız HTTPS kullanmalıdır. HTTPS olmadan, bu veriler ağ üzerinde düz metin olarak yolculuk eder ve yakalanabilir. HTTP ile tarayıcı, kullanıcıları uzaklaştıran ve sisteme duyulan güveni azaltan bir "güvenli olmayan site" uyarısı gösterebilir.

> ***Temel kavram: protokol*** *İki tarafın nasıl iletişim kuracağını tanımlayan kurallar bütünü. HTTP, tarayıcıların ve sunucuların web'de bilgi alışverişi için kullandığı protokoldür. HTTPS, onun şifreli sürümüdür.*

Sunucunun yanıtladığı şey

Bir sunucu her istek aldığında, bir sayı içeren bir yanıt döndürür: **durum kodu**. Bu sayı, istekle ne olduğunu belirtir.

Kodlar kategorilere göre gruplanır. 2 ile başlayanlar başarı anlamına gelir. 3 ile başlayanlar yönlendirmedir. 4 ile başlayanlar istemci tarafında bir hata belirtir — istekte bir şey yanlıştır. 5 ile başlayanlar sunucu tarafında bir hata belirtir — sunucu isteği aldı ama onu doğru işleyemedi.

Sistemlerle çalışan biri için en önemli üçü:

200 — OK. İstek doğru işlendi ve sunucu beklenen içeriği döndürdü. Normal sonuç budur. Bir sayfa görüyorsan, sunucu 200 yanıtı verdi.

404 — Not Found. Sunucu isteği aldı, ama istediğin kaynak o sunucuda yok. Kullandığın adres, mevcut hiçbir dosyaya, sayfaya ya da veriye karşılık gelmiyor. Bu bir sunucu hatası değildir: aradığın şey orada değildir.

500 — Internal Server Error. Sunucu isteği aldı, onu işlemeye çalıştı ve kendi işleyişinde bir şey başarısız oldu. Sorun istediğin şeyde değil: sunucunun isteği nasıl ele aldığındadır. Kodun ya da sunucu yapılandırmasının bir hatasıdır.

Kod	Ad	Ne oldu	Günlük örnek
200	OK	Sunucu istediğin şeyi buldu ve teslim etti.	Bir sayfa açıyorsun ve sorunsuz yükleniyor.

Kod	Ad	Ne oldu	Günlük örnek
301 / 302	Yönlendirme	Kaynak başka bir adreste. Tarayıcı seni oraya otomatik götürür.	Eski bir URL seni sitenin yeni alan adına gönderir.
400	Bad Request	İstek bozuk biçimde geldi; sunucu onu yorumlayamadı.	Zorunlu verileri eksik bir form gönderiyorsun.
401	Unauthorized	Geçerli kimlik bilgileri sunmadın. Sunucu kim olduğunu bilmiyor.	Oturum açmadan özel içeriği görmeye çalışıyorsun.
403	Forbidden	Kimlik bilgilerin geçerli, ama bunun için iznin yok.	Hesabına girdin, ama o bölüm rolüne uygun değil.
404	Not Found	İstediğin şey o sunucuda yok.	Bir URL yanlış yazıldı ya da artık yok.
500	Internal Server Error	Sunucu isteğini aldı ama işlerken başarısız oldu.	Bir satın almayı tamamlamaya çalışıyorsun ve “beklenmeyen hata” beliriyor.
503	Service Unavailable	Sunucu şu an karşılayamıyor; aşırı yüklü ya da bakımda.	Yoğun bir satış ya da lansman sırasında bir site çöküyor.

Tüm kodları ezberlemek gerekmez. Önemli olan örüntüyü tanımaktır: 2xx genellikle başarıyı, 3xx yönlendirmeyi, 4xx istek tarafında bir sorunu ve 5xx sunucu tarafında bir sorunu belirtir.

Uygulama neden "çöktü"

"Sayfa yüklenmiyor", "sistem çökmüş", "çalışmıyor" — bir şey kullanılamaz hale geldiğinde, genellikle belirli bir teknik neden vardır. Olası başarısızlık noktalarını anlamak, onu çözecek birini aramadan önce sorunu daraltmayı sağlar.

En sık görülen nedenler, oluştukları yere göre sıralanmış:

Sunucu çökmüş. Sunucu yanıt vermeyi bıraktı. Barındırma servisinde bir sorun çıktığı için, sunucu bir güncelleme nedeniyle yeniden başladığı için ya da sistem kaynaksız (bellek, CPU) kaldığı ve çalışmayı bıraktığı için olabilir. Bu durumda tarayıcı hiçbir tür yanıt almaz.

Alan adı çözülmüyor. DNS, alan adının adını bir IP adresine çeviremiyor. Alan adının süresi dolduğu için, DNS yapılandırması hatalı değiştiği için ya da DNS sunucusunda bir sorun olduğu için olabilir. Sonuç bir öncekine benzer: tarayıcı nereye gideceğini bilmez.

Sertifikanın süresi dolmuş. Sunucunun HTTPS sertifikasının süresi dolduysa, tarayıcı bağlantıyı engeller ve bir güvenlik uyarısı gösterir. Sayfa teknik olarak vardır ve sunucu çalışır, ama tarayıcı bağlantıyı reddederek kullanıcıyı korur. Görünür sonuç: bir uyarı ekranı, sayfa değil.

500 hatası. Sunucu yanıt verir ama bir iç hatayla. Sayfa vardır, sunucu açıktır, ama kod isteği işlerken bir sorunla karşılaştı. Kullanıcı, beklenen içerik yerine bir hata mesajı görür.

Ağ sorunu. Kullanıcının cihazı ile sunucu arasındaki bağlantı, aradaki bir noktada kesintiye uğramıştır. Kullanıcının internet bağlantısı, bir ağ sağlayıcısı ya da iki taraf arasındaki altyapının bir noktası olabilir.

Bunları yüzeysel olarak ayırt etmek, nereden başlanacağını bilmeye yardımcı olur:

- Sayfa hiçbir şey yüklemiyorsa ve uyarı yoksa → muhtemelen sunucu çökmüş ya da DNS çözülmüyor.
- Tarayıcı açık bir güvenlik uyarısı gösteriyorsa → sertifikanın süresi dolmuş ya da şifresiz HTTP.
- Sayfa, içerikte bir hata mesajıyla yükleniyorsa → muhtemelen 500 hatası, kod sorunu.
- Yalnızca sen erişemiyorsan ama başkaları erişebiliyorsa → muhtemelen bir ağ ya da yerel önbellek sorunu.

Bunu bilmek, onu doğrudan çözebilmek anlamına gelmez. Onu çözebilecek kişiye kesin biçimde iletebilmek anlamına gelir.

> ***Bir web sitesi yüklenmediğinde işe yarar sorular***
>
> *Sistem kullanılamaz olduğunda, bu sorular onu çözebilecek birini aramadan önce sorunu daraltmaya yardımcı olur:*
>
> - *Sorun herkesin başına mı geliyor yoksa yalnızca benim mi?*
> - *Alan adı çözülüyor mu? Ad bir adrese işaret ediyor mu?*
> - *Sunucu yanıt veriyor mu? Bir hata bile olsa bir şey ulaşıyor mu?*
> - *Hangi hata kodu var — 404, 500 — yoksa hiçbir tür yanıt yok mu?*
> - *Üretim mi yoksa yalnızca bir test ortamı mı hata veriyor?*
> - *Hatanın günlükleri var mı?*
> - *Yakın zamanda bir değişiklik oldu mu — dağıtım, yapılandırma değişikliği, alan adı yenileme?*

Sık görüldüğü ve kafa karıştırdığı için anmaya değer bir durum var: sistem bazı kişiler için çalışır, bazıları için çalışmaz. Bu neredeyse hiçbir zaman sunucunun çöktüğü anlamına gelmez. Genellikle bir DNS yayılma sorununu gösterir — alan adı yapılandırmasında bir değişiklik yapıldığında, o değişikliğin dünyadaki tüm DNS sunucularına ulaşması dakikalar ile saatler arasında sürer. Bu süre boyunca, farklı kullanıcılar farklı DNS sunucularına danışır ve farklı yanıtlar alır. Bazıları yeni siteyi görür; bazıları eskisini görmeye devam eder ya da hiçbir şey görmez. Bu beklenen bir davranıştır, acilen çözülmesi gereken bir hata değil.

Bir toplantıda bunu nasıl duyacaksın

Bir teknik ekiple ya da bir sağlayıcıyla yapılan bir toplantıda, herkesin neyden bahsedildiğini anladığını varsayan ifadeler kullanılır. Bu konudaki en sık görülenler, pratikte ne anlama geldikleri ve ne sorabileceğin şunlar.

“DNS sorunu gibi görünüyor.” Alan adının adı doğru çözülmüyor — sunucunun IP adresine çevrilemiyor. Hatalı bir yapılandırma ya da henüz tam yayılmamış yakın bir değişiklik olabilir.

İşe yarar soru: "Tüm kullanıcıların başına mı geliyor yoksa yalnızca bazılarının mı?" Herkeste oluyorsa, muhtemelen bir yapılandırma hatasıdır. Yalnızca bazılarındaysa, büyük olasılıkla devam eden bir yayılmadır — saatler süren ve kendiliğinden çözülen bir süreç.

Varsaymamak gerekir: sunucunun çöktüğünü. DNS ve sunucu farklı parçalardır. DNS hata verirken sunucu kusursuz çalışıyor olabilir.

"Sunucu 500 yanıtı veriyor." Sunucu isteği aldı ama onu işlerken bir hatayla karşılaştı. Sorun kodda ya da sistemin yapılandırmasındadır, kullanıcının isteğinde ya da ağda değil.

İşe yarar soru: "Hatanın günlükleri var mı?" Günlükler, sistemin başarısız olmadan önce ne yaptığının kaydıdır. Bir 500 neredeyse her zaman nedeni belirleyen bir iz bırakır. Günlükler olmadan teşhis çok daha zorlaşır.

Varsaymamak gerekir: bunun bir internet ya da kullanıcının cihazı sorunu olduğunu. Bir 500, sunucunun gerçekten yanıt verdiği anlamına gelir — yalnızca bir iç hatayla.

"Sertifikanın süresi doldu." Sunucunun HTTPS sertifikasının süresi doldu. Tarayıcı, bağlantının artık güvenli olarak güvence altına alınamayacağını tespit eder ve erişimi varsayılan olarak engeller. Sistem içeriden kusursuz çalışıyor olabilir, ama hiçbir kullanıcı giremez.

İşe yarar soru: "Yenileme ne kadar sürüyor?" Çoğu durumda hızlı bir süreçtir. Bilinmesi gereken şey, tahmini çözüm süresi ve haber verilmesi gereken, hâlihazırda etkilenmiş kullanıcılar olup olmadığıdır.

Varsaymamak gerekir: bir kod ya da altyapı sorunu olduğunu. Bu, bir belgenin süresinin dolmasına benzer idari bir süre dolmasıdır. Sistemin kalitesi hakkında hiçbir şey belirtmez.

"Üretimde çökmüş." Gerçek kullanıcıların kullandığı sistem kullanılamaz durumda. Bu bir test ya da geliştirme ortamı değil: gerçek trafiğin ve gerçek verilerin olduğu sistemdir. Bu ifade aciliyet belirtir.

İşe yarar soru: "Çözmek için çalışılıyor mu? Tahmini süre nedir?" Bu iki soru, kullanıcılara bir şey bildirilmesi mi yoksa sessizce beklenmesi mi gerektiğini bilmek için gereken bağlamı verir.

Varsaymamak gerekir: sorunun herkesi aynı ölçüde etkilediğini. Bazen çöküş kısmidir — yalnızca bir bölgeyi, bir cihazı ya da belirli bir işlevi etkiler.

Bu bölümde öğrendiklerin

- Bir URL, internetteki bir kaynağı tanımlayan adrestir. Ona ulaşmak için tarayıcı, alan adının adını bir IP adresine çeviren DNS'e danışır.
- Bir sunucu, istekleri alan ve yanıtları döndüren, her zaman açık bir bilgisayardır. Barındırma, o sunucuyu kiralayan servistir.
- HTTP, tarayıcının ve sunucunun nasıl iletişim kuracağını tanımlayan protokoldür. HTTPS, onun şifreli sürümüdür; hassas veri işleyen herhangi bir sistem için zorunludur.
- Durum kodları, sunucunun ne olduğunu belirtmek için kullandığı dildir: 200 başarı, 404 kaynak bulunamadı, 500 sunucunun iç hatasıdır.
- "Uygulama çöktü"nün belirli teknik nedenleri vardır: sunucu çökmüş, DNS çözülmüyor, sertifikanın süresi dolmuş, kod hatası. Bunları bilmek sorunu daraltmayı sağlar.

Sırada ne var

3. Bölüm, az önce anladığın o konuşmanın iki aktörüne girer: istemci ve sunucu. Bilginin onların arasında nasıl yolculuk ettiğini artık biliyorsun. Sırada, bu alışverişte her birinin ne yaptığını — ve bu bölünmenin neden herhangi bir uygulamanın temel yapısı olduğunu — anlamak var.

Bu örnek burada bitiyor.

Kitabın tamamı eksiksiz haritayla devam eder:
istemci, sunucu, veritabanları, kod,
API'ler, güvenlik, yapay zekâ ve dağıtım.

Tam sürüm şunları içerir:
22 bölüm, pratik ekler,
teknik sözlük ve PDF + EPUB sürümleri.

elnuevocreadortecnico.com