WEB & YAZILIM GELİŞTİRME SERİSİ · MODÜL 1

Yazılım Dünyasına Giriş & Kariyer Rehberi

Yazılım nedir, web nasıl çalışır, hangi roller ve diller var, sana hangi yol uygun? Sıfırdan başlayanlar için adım adım yol haritası ve kariyer rehberi.

Sıfırdan başlayanlar için · Eğitim amaçlıdır

Bu Kitap Hakkında

Bu rehber, yazılım ve web geliştirme dünyasına ilk adımını atanlar için hazırlanmıştır. Bir kursa giden, üniversite bölümü seçen ya da kendi kendine öğrenmek isteyen herkese; yazılımın ne olduğunu, web'in nasıl çalıştığını, hangi rollerin ve dillerin bulunduğunu ve kişiye en uygun yolu seçmeyi öğretir.

Dört seviye ve yirmi altı bölüm boyunca, her bölümde 'hangi yol sana uygun?' rehber kartı ve gerçek bir alıştırma yer alır. Bu, on altı modüllük 'Web & Yazılım Geliştirme' serisinin ilk modülüdür; teknik beceriler (HTML, CSS, JavaScript, algoritma, backend dilleri, veritabanı, mobil ve daha fazlası) sonraki modüllerde gelir. Bu seri eğitim amaçlıdır.

Web & Yazılım Geliştirme Serisi · Modül 1

İçindekiler

SEVİYE 1

Bu Dünya Nedir?

Yazılım ve web dünyasına ilk adım: yazılım nedir, internet nasıl çalışır, bir web sitesinin parçaları, yazılımcı rolleri, diller, hangi yol size uygun ve nasıl öğrenilir.

BÖLÜM 01

Web ve Yazılım Nedir?

Yazılım, bir bilgisayara "ne yapacağını" söyleyen komutlar bütünüdür. Web siteleri, mobil uygulamalar, oyunlar, bankacılık sistemleri — hepsi yazılımdır. Bu rehber, sizi bu dünyada doğru yola koymak için var.

Karşımıza çıkan yazılım türleri

- **Web sitesi:** tarayıcıda açılan sayfalar (haber sitesi, blog).
- **Web uygulaması:** tarayıcıda çalışan, etkileşimli program (e-posta, çevrimiçi tablo).
- **Mobil uygulama:** telefona kurulan program (sosyal medya, harita).
- **Masaüstü yazılımı:** bilgisayara kurulan program (Office, oyunlar).

"Geliştirici" ne yapar?

- Bir ihtiyacı/sorunu anlar ve onu çözecek yazılımı **tasarlar**.
- Kod yazarak bu çözümü **hayata geçirir**.
- Test eder, hataları (bug) bulup **düzeltir**.
- Yazılımı yayınlar ve zamanla **geliştirir**.

Şema 1.1 — Karşımıza çıkan başlıca yazılım türleri.

İPUCU

Yazılımcı olmak "her şeyi ezberlemek" değildir; **problem çözmeyi** ve **doğru bilgiyi bulmayı** öğrenmektir. En deneyimli geliştiriciler bile her gün arama yapar, dokümantasyon okur. Bu bir eksiklik değil, mesleğin doğasıdır.

Hangi yol sana uygun? REHBER

Bu rehber boyunca her bölümde, öğrendiğiniz konunun sizi hangi role/yola yönlendirdiğini göstereceğiz. Şimdilik şu kadarını bilin:

- Görseller, sayfalar ve kullanıcı deneyimi ilgini çekiyorsa → **önyüz (frontend)**.
- Mantık, veri ve sistemin "arka tarafı" ilgini çekiyorsa → **arkayüz (backend)**.
- İkisini de istiyorsan → **fullstack**. Henüz seçmene gerek yok; ileride netleşecek.

Alıştırma 8 dk

Çevreni gözlemle:

1. Bugün kullandığın 5 yazılımı (uygulama/site) listele.
2. Her birini "web / mobil / masaüstü" diye sınıflandır.
3. Hangisini sen yapmak isterdin? Bir cümleyle yaz.

BÖLÜM 02

İnternet ve Web Nasıl Çalışır?

Bir web adresini yazıp Enter'a bastığında perde arkasında çok şey olur. Bu mekanizmayı anlamak, geliştiriciliğin temelidir; çünkü yazdığın her web yazılımı bu sistemin üstünde çalışır.

İstemci ve sunucu

- **İstemci (client):** senin tarayıcın/telefonun — bir şey ister.
- **Sunucu (server):** isteğe yanıt veren, her zaman açık bilgisayar.
- Web, sürekli bir **istek–yanıt** alışverişidir: "şu sayfayı ver" → "buyur".

Adresten sayfaya yolculuk

1. Tarayıcıya bir adres yazarsın (örn. `ornek.com`).
2. **DNS**, bu adı bir **IP numarasına** (sunucunun adresi) çevirir.
3. Tarayıcı o sunucuya **HTTP/HTTPS** ile istek gönderir.
4. Sunucu sayfayı (HTML/CSS/JS) yanıt olarak yollar; tarayıcı ekrana çizer.

> **İPUCU**
>
> **HTTPS**'teki "S" güvenliği (şifreleme) ifade eder. Adres çubuğunda kilit ve "https://" görmek, veri alışverişinin şifrelendiği anlamına gelir — özellikle giriş ve ödeme sayfalarında bu şarttır.

Şema 2.1 — Bir adresten sayfaya: istek–yanıt yolculuğu.

Hangi yol sana uygun?

REHBER

Bu konu hangi yollarda öne çıkar?

- Tüm web geliştiriciler bunu bilmek zorundadır (frontend + backend).
- Sunucu, DNS ve ağ tarafı seni heyecanlandırıyorsa → ileride **DevOps/sistem** ilgini çekebilir.
- Endişelenme: şimdilik "büyük resmi" anlaman yeterli; detaylar Domain & Hosting modülünde.

Alıştırma

8 dk

Sistemi keşfet:

1. Bir web sitesini açıp adres çubuğundaki kilit/https işaretini bul.
2. İstemci ve sunucunun ne olduğunu kendi cümlelerinle yaz.
3. Bir "istek–yanıt" örneğini günlük hayattan benzet (örn. lokantada sipariş).

BÖLÜM 03

Bir Web Sitesinin Sahne Arkası

Bir web sitesi tek parça değildir; birlikte çalışan katmanlardan oluşur. Bu katmanları tanımak, "neyi öğrenmem gerekiyor?" sorusunun haritasını verir.

Üç ana katman

- **Önyüz (Frontend):** kullanıcının gördüğü her şey — `HTML` (yapı), `CSS` (görünüm), `JavaScript` (etkileşim).
- **Arkayüz (Backend):** sunucuda çalışan mantık — kullanıcı girişi, hesaplama, kurallar (PHP/Python/C# gibi dillerle).
- **Veritabanı:** bilgilerin saklandığı yer — kullanıcılar, ürünler, siparişler.

Bir örnekle: giriş yapmak

- **Önyüz** giriş formunu gösterir, bilgileri alır.
- **Arkayüz** şifreyi kontrol eder, doğru mu diye bakar.
- **Veritabanı** kullanıcının kayıtlı bilgisini sağlar.
- Sonuç önyüze döner: "Hoş geldin" veya "Hatalı şifre".

En basit web sayfası (HTML) — ileride detaylıca öğreneceksin

```
<!doctype html>
<html lang="tr">
  <body>
    <h1>Merhaba Dünya</h1>
    <p>İlk web sayfam.</p>
  </body>
</html>
```

Şema 3.1 — Bir web sitesinin üç katmanı birlikte çalışır.

İPUCU

Bu üç katmanı bir lokantaya benzet: **önyüz** salon ve menü (müşterinin gördüğü), **arkayüz** mutfak (işin yapıldığı), **veritabanı** kiler (malzemelerin saklandığı). Hepsi birlikte çalışır.

Hangi yol sana uygun?

REHBER

Hangi katman seni daha çok çekiyor?

- Salonu/menüyü tasarlamak hoşuna gidiyorsa → **frontend**.
- Mutfaktaki mantığı kurmak hoşuna gidiyorsa → **backend**.
- İkisini birden istiyorsan → **fullstack** (çoğu kişi buradan geçer).

Alıştırma

10 dk

Katmanları ayırt et:

1. Sevdiğin bir web uygulamasını düşün (örn. bir alışveriş sitesi).
2. Hangi kısımlar "önyüz", hangileri "arkayüz/veritabanı" olabilir, listele.
3. Sana en ilginç gelen katmanı işaretle.

BÖLÜM 04

Yazılımcı Rolleri

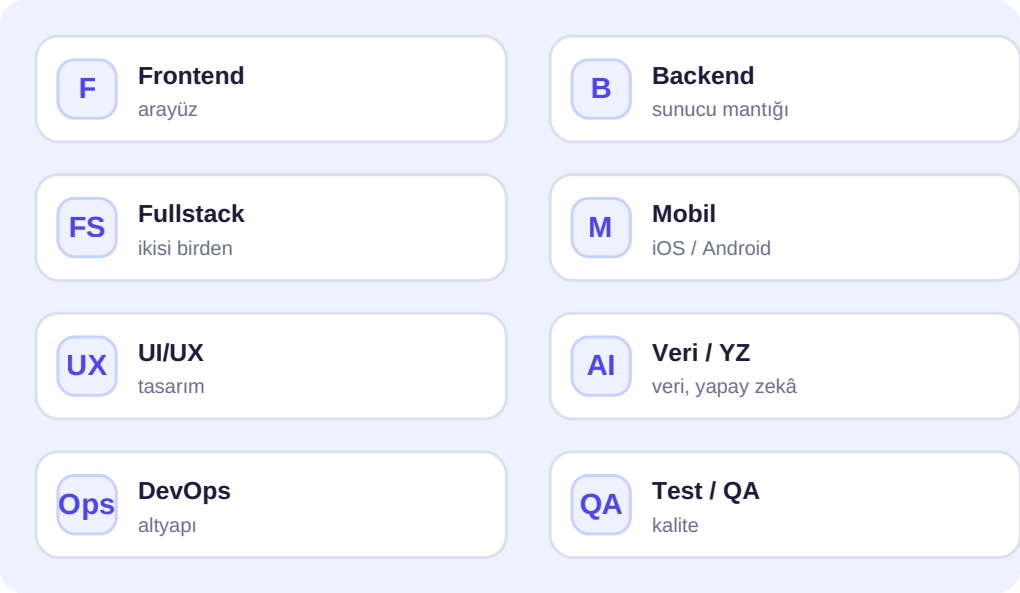
"Yazılımcı" tek bir meslek değil; bir şemsiyedir. Altında birbirinden farklı, hepsi değerli roller vardır. Bunları tanımak, kendine bir hedef seçmene yardım eder.

En yaygın roller

- **Frontend Geliştirici:** kullanıcının gördüğü arayüzü kurar (HTML/CSS/JS).
- **Backend Geliştirici:** sunucu mantığını, veriyi, kuralları yazar.
- **Fullstack Geliştirici:** hem önyüz hem arkayüzle çalışır.
- **Mobil Geliştirici:** iOS/Android uygulamaları yapar.

Diğer önemli roller

- **UI/UX Tasarımcı:** arayüzün görünümünü ve kullanım deneyimini tasarlar (kod ağırlığı az).
- **Veri / Yapay Zekâ:** veriyle, analizle ve yapay zekâ modelleriyle çalışır.
- **DevOps / Sistem:** sunucu, dağıtım ve altyapıyı yönetir.
- **Test / QA:** yazılımın doğru çalıştığını güvence altına alır.
- **Oyun Geliştirici:** oyun motorlarıyla oyun yapar.



Şema 4.1 — Yazılımcı rolleri: hepsi değerli, farklı yollar.

İPUCU

Hiçbir rolü "en iyisi" diye seçme; **sana en keyif vereni** seç. Sıkılmadan saatlerce uğraşabileceğin alan, uzun vadede en başarılı olacağın alandır. Üstelik ilk seçimin kalıcı değil; geçişler çok normaldir.

Hangi yol sana uygun?

REHBER

Kısa bir eşleştirme:

- Görsel, renk, kullanıcı deneyimi → **Frontend** veya **UI/UX**.
- Mantık, veri, sistemler → **Backend** veya **Veri**.
- Telefon uygulamaları → **Mobil**. Altyapı/sunucu → **DevOps**.
- Hepsine meraklıysan → **Fullstack** başlamak için harikadır.

Alıştırma

10 dk

Rolleri keşfet:

1. Yukarıdaki rollerden sana en çekici gelen 2 tanesini seç.
2. Her biri için "neden ilgimi çekti?" diye bir cümle yaz.
3. O rollerde çalışan birinin bir günü nasıl geçer, tahmin et.

BÖLÜM 05

Programlama Dilleri Manzarası

Onlarca programlama dili var ama hepsini öğrenmen gerekmez. Önemli olan, hangi dilin ne işe yaradığını bilmek ve doğru olanı doğru sırada öğrenmektir.

Önce bir kavram: HTML ve CSS "dil" mi?

- **HTML ve CSS** birer **işaretleme/biçimleme** dilidir; sayfanın yapısını ve görünümünü tarif eder ama "programlama" (mantık, karar) yapmaz.
- **JavaScript** gerçek bir programlama dilidir; karar verir, hesaplar, etkileşim kurar.
- Bu yüzden web yolu genelde HTML → CSS → JavaScript sırasıyla başlar.

Popüler diller ne işe yarar?

- **JavaScript:** web önyüzü (ve Node.js ile arkayüz). Web'in olmazsa olmazı.
- **Python:** öğrenmesi kolay; veri, yapay zekâ, otomasyon, web arkayüzü.
- **PHP:** klasik web arkayüzü; çok sayıda sitenin temeli.
- **C#:** Microsoft ekosistemi; web, masaüstü, oyun (Unity).
- **Diğerleri:** Java, C++, Swift, Kotlin, Go… her birinin kendi alanı var.

Aynı iş, farklı dillerde: "Merhaba" yazdırmak

```
// JavaScript
console.log("Merhaba");

# Python
print("Merhaba")

// C#
Console.WriteLine("Merhaba");
```

İPUCU

Acemiyken "hangi dil en iyi?" tartışmasına takılma. **İlk dilini öğrenmek en zor olanıdır**; ikincisi çok daha kolay gelir, çünkü mantık ortaktır. Bir dilde ustalaşmak, hepsinin kapısını aralar.

Hangi yol sana uygun? REHBER

Hangi dille başlamalı?

- Web önyüzü hedefin varsa → **JavaScript** (HTML/CSS'ten sonra).
- Genel başlangıç, veri/yapay zekâ ilgisi → **Python** (en başlangıç-dostu).
- Microsoft/masaüstü/oyun → **C#**. Klasik web arkayüzü → **PHP**.
- Bu seride dördünü de (JS/Python/PHP/C#) ayrı modüllerde ele alacağız.

Alıştırma 8 dk

Dilleri tanı:

1. Yukarıdaki dillerden ilgini çeken birini seç.
2. O dilin "ne için kullanıldığını" bir cümleyle yaz.
3. Hedefine (web/mobil/veri) göre hangisiyle başlayacağını not et.

BÖLÜM 06

Hangi Yol Sana Uygun?

Doğru başlangıç, "popüler olan" değil, "sana uygun olan"dır. Kendini biraz tanıyarak, hangi alanda mutlu ve üretken olacağını şimdiden sezebilirsin.

Kendine sor

- Görsel sonuçlar mı, yoksa görünmeyen mantık mı beni tatmin eder?
- Hızlı görünür sonuç mu isterim (bir sayfa yapıp görmek), yoksa derin problem çözmeyi mi?
- Tek başıma derinleşmeyi mi, çok yönlü olmayı mı severim?
- Hedefim ne: iş bulmak, kendi ürünümü yapmak, freelance, akademik?

Kaba bir pusula

- **Anında görsel sonuç + tasarım** → Frontend (HTML/CSS/JS).
- **Mantık, veri, sistem** → Backend (Python/PHP/C#) + Veritabanı.
- **Telefon uygulaması hayali** → Mobil.
- **Kararsızım, her şeyi merak ediyorum** → Fullstack ile başla, zamanla daralt.

Şema 6.1 — Kaba bir pusula: ilgine göre yol.

İPUCU

İlk seçimin bir **başlangıç noktası**, ömür boyu sözleşme değil. Pek çok geliştirici frontend'de başlayıp backend'e geçer ya da tam tersi. Önemli olan başlamak; yol yürürken netleşir.

Hangi yol sana uygun? REHBER

Hâlâ kararsızsan, en güvenli başlangıç:

- **HTML → CSS → JavaScript** ile başla. Çünkü: hızlı görsel sonuç verir, motive eder ve web'in temelidir.
- Bu üçü, hangi yola gidersen git işine yarar.
- Birkaç hafta sonra neyi sevdiğini çok daha net bileceksin.

Alıştırma 10 dk

Kendi pusulanı çıkar:

1. Yukarıdaki "kendine sor" sorularını yanıtla.
2. Hangi yola eğilimli olduğunu bir cümleyle yaz.
3. İlk 1 ay ne öğreneceğine dair küçük bir hedef belirle.

BÖLÜM 07

Nasıl Öğrenilir: Üniversite, Kurs, Kendi Kendine

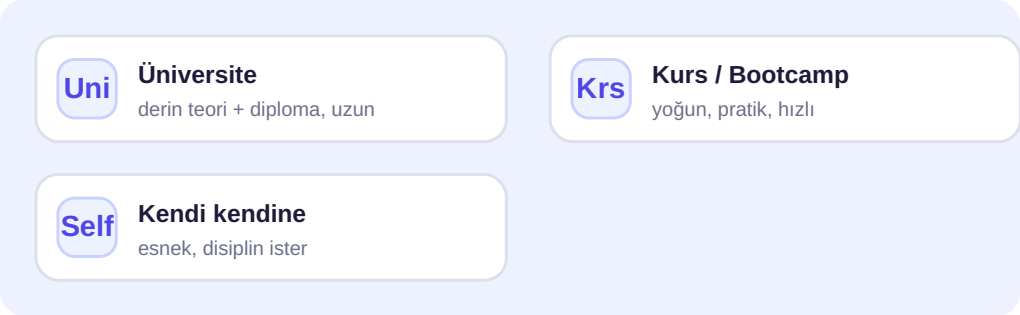
Yazılımcı olmanın tek bir yolu yok. Üniversite, kurs/bootcamp ve kendi kendine öğrenme; her birinin güçlü ve zayıf yanları var. En iyisi çoğu zaman bunların bir karışımıdır.

Üç ana yol

- **Üniversite (Bilgisayar/Yazılım Müh.):** derin teori, diploma, ağ; ama uzun ve bazen güncel pratikten kopuk.
- **Kurs / Bootcamp:** yoğun, pratik, hızlı iş odaklı; ama temel teori daha az olabilir.
- **Kendi kendine:** esnek, ücretsiz/ucuz kaynak bol; ama disiplin ve yol haritası gerektirir.

Hepsinin ortak gerçeği

- Hangi yolu seçersen seç, **kendin proje yapmadan** öğrenemezsin.
- Diploma/sertifika kapı açar; ama seni asıl **portfolyon ve becerin** işe aldırır.
- Öğrenme hiç bitmez; bu meslek sürekli güncellenir.



Şema 7.1 — Üç öğrenme yolu; en güçlüsü çoğu zaman karışımdır.

İPUCU

"Önce her şeyi öğreneyim, sonra proje yaparım" tuzağına düşme. **Öğrendiğin günden itibaren küçük projeler yap.** Bilgi, ancak uygulayınca kalıcı olur; izlediğin onuncu eğitim videosu, yaptığın ilk küçük projeden daha az şey öğretir.

Hangi yol sana uygun? REHBER

Hangi yol kime uygun?

- **Genç/öğrenciysen ve teori + diploma istiyorsan** → üniversite (yanında proje yap).
- **Hızlı iş değişimi/pratik istiyorsan** → kurs/bootcamp + portfolyo.
- **Disiplinliysen ve esneklik istiyorsan** → kendi kendine + net yol haritası (bu seri gibi).
- En güçlüsü: **karışım** — örn. üniversite + kendi projelerin.

Alıştırma 8 dk

Yolunu seç:

1. Üç yolu kendi durumuna göre artı-eksi yaz.
2. Sana en uygun (veya karışım) yolu belirle.
3. O yolda ilk atacağın somut adımı yaz.

BÖLÜM 08

Öğrenme Zihniyeti

Yazılım öğrenmenin önündeki en büyük engel zekâ değil, zihniyettir. Doğru alışkanlıklar ve gerçekçi beklentiler, yolda kalmanı sağlar.

Bilmen gereken gerçekler

- **Hata yapmak işin parçasıdır.** Kod ilk seferde çalışmaz; hatayı bulup düzeltmek (debug) mesleğin kendisidir.
- **Takılmak normaldir.** En iyi geliştiriciler de saatlerce takılır; fark, pes etmemekte ve doğru soruyu sormakta.
- **Karşılaştırma tuzağı:** başkasının ilerlemesiyle kendininkini kıyaslama; herkesin temposu farklı.

Sağlıklı alışkanlıklar

- **Küçük ve düzenli:** her gün 30-60 dk, haftada bir kez 8 saatten iyidir.
- **Proje odaklı:** öğrendiğini hemen küçük bir projede kullan.
- **"Tutorial cehennemi"nden kaç:** sürekli video izleyip hiç kod yazmama tuzağına düşme.
- **Topluluğa katıl:** takıldığında sormak, yalnız uğraşmaktan hızlıdır.

Şema 8.1 — Sağlıklı öğrenme döngüsü.

İPUCU

Bir hatayla karşılaşınca panik yapma; hata mesajını **dikkatle oku**. Çoğu zaman çözümün ipucu oradadır. Hata mesajını arama motoruna/yapay zekâya yazmak, çözüme giden en hızlı yollardan biridir (Yapay Zekâ modülünde derinleşeceğiz).

Hangi yol sana uygun?

REHBER

Bu zihniyet her yol için geçerli:

- Frontend, backend, mobil — fark etmez; hata yapıp düzelterek öğrenirsin.
- Sabır ve süreklilik, yetenekten daha belirleyicidir.
- Bu seri sana yol haritası verir; yürümek ve pratik etmek sana kalır.

Alıştırma

8 dk

Zihniyetini kur:

1. Haftalık gerçekçi bir çalışma planı yaz (gün/süre).
2. "Takıldığımda ne yapacağım?" diye 3 adım belirle (oku, ara, sor).
3. İlk küçük proje fikrini bir cümleyle yaz.

SEVİYE 2

Yolu Planlamak

Hedefe giden rotalar: frontend, backend, fullstack ve mobil yol haritaları; olmazsa olmaz araçlar; bir geliştiricinin günü; portfolyo ve ilk projeler.

BÖLÜM 09

Frontend Yol Haritası

Önyüz (frontend), kullanıcının gördüğü her şeydir. Hızlı görsel sonuç verdiği için çoğu kişinin sevdiği bir başlangıçtır. İşte mantıklı bir sıra.

Sıra önemli

- **1) HTML** — sayfanın iskeleti: başlıklar, paragraflar, formlar.
- **2) CSS** — görünüm: renk, düzen, responsive (mobil uyum).
- **3) JavaScript** — etkileşim: tıklama, form kontrolü, veri çekme.
- **4) Bir framework** — React/Vue gibi (büyük projeleri kolaylaştırır).

Yanında öğrenilenler

- **Git/GitHub** ile kodunu sürümle ve sergile.
- **Tarayıcı geliştirici araçları** ile hata ayıkla.
- **Responsive tasarım** ile her ekranda iyi görünüm.
- Temel **erişilebilirlik** ve **performans**.

Şema 9.1 — Frontend yol haritası: sıra önemli.

İPUCU

Framework'e (React gibi) atlamak için acele etme. Önce **saf HTML/CSS/JavaScript** ile sağlam bir temel kur; framework'ler bu temelin üstüne oturur. Temeli zayıf olanın framework'te işi zorlaşır.

Hangi yol sana uygun?

REHBER

Frontend sana göre mi?

- Görseli, düzeni, kullanıcının deneyimini önemsiyorsan → evet.
- Yaptığını anında ekranda görmek seni motive ediyorsa → evet.
- Bu seride sırasıyla HTML, CSS ve JavaScript modülleri seni adım adım taşıyacak.

Alıştırma

8 dk

Frontend planı:

1. HTML→CSS→JS sırasını neden bu şekilde olduğunu yaz.
2. İlk yapmak istediğin basit arayüzü (örn. kişisel sayfa) belirle.
3. Bunun için hangi 3 konuyu öğrenmen gerektiğini listele.

BÖLÜM 10

Backend Yol Haritası

Arkayüz (backend), perde arkasındaki mantıktır: kullanıcı girişi, kurallar, hesaplamalar ve veriyle çalışma. Görünmez ama bir uygulamanın "beyni"dir.

Backend'in yapı taşları

- **Bir dil:** PHP, Python veya C# (bu seride üçü de ayrı modül).
- **Veritabanı:** veriyi saklama ve sorgulama (SQL — ayrı modül).
- **API:** önyüz ile arkayüzün konuşma yolu.
- **Sunucu/hosting:** kodun yayında çalıştığı yer.

Mantıklı sıra

1. Programlama mantığını oturt (**Algoritma** modülü).
2. Bir backend dili seç ve temellerini öğren.
3. **Veritabanı (SQL)** ile veriyi saklamayı öğren.
4. **API** yaz; önyüzle konuştur.
5. **Hosting/sunucu** ile yayına al.

Backend mantığı sezgisi (sözde kod)

```
# Kullanıcı giriş yapmak istiyor
eger sifre == veritabanindaki_sifre:
    yanit = "Hoş geldin"
degilse:
    yanit = "Hatalı şifre"
```

Şema 10.1 — Backend yol haritası.

İPUCU

Backend'de üç dili birden öğrenmeye çalışma. **Birini seç, derinleş**; mantığı kavradığında diğerlerine geçmek kolaydır. Bu seri PHP, Python ve C#'ı ayrı ayrı sunar ama önce birini bitirmeni öneririz.

Hangi yol sana uygun? REHBER

Backend sana göre mi?

- Görünmeyen mantığı kurmak, problem çözmek hoşuna gidiyorsa → evet.
- Veri, kural ve sistemler ilgini çekiyorsa → evet.
- Hangi dil? Python başlangıç-dostu; PHP klasik web; C# Microsoft dünyası.

Alıştırma

10 dk

Backend planı:

1. Backend'in dört yapı taşını kendi cümlelerinle yaz.
2. Bir backend dili seç ve nedenini belirt.
3. O dille yapmak istediğin küçük bir işlevi tarif et.

BÖLÜM 11

Fullstack ve Mobil Yolları

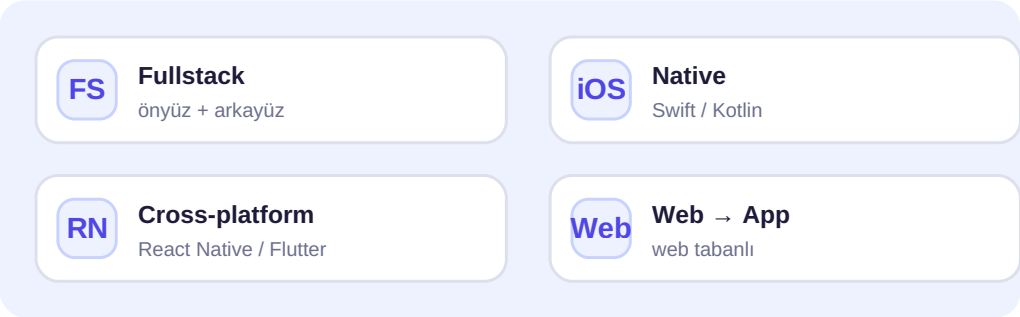
Frontend ve backend'i birleştiren fullstack, baştan sona uygulama yapabilmektir. Mobil ise telefon uygulamaları dünyasıdır. İkisi de güçlü kariyer yollarıdır.

Fullstack

- Hem önyüz hem arkayüzle çalışır; bir fikri tek başına hayata geçirebilir.
- Genelde frontend veya backend'de başlayıp diğerini ekleyerek gelişir.
- Küçük ekipler ve kendi ürününü yapmak isteyenler için idealdir.

Mobil

- **Native:** iOS için Swift, Android için Kotlin — en yüksek performans.
- **Cross-platform:** React Native veya Flutter ile tek kodla iki platform.
- Web bilgisi (özellikle JavaScript) mobil cross-platform'a geçişi kolaylaştırır.



Şema 11.1 — Fullstack ve mobil yolları.

İPUCU

"Fullstack" her şeyi aynı anda öğrenmek değildir; **bir uçtan başlayıp zamanla diğerini eklemektir**. Önce frontend (veya backend) işini iyi yap, sonra köprüyü kur. Aynı anda her şeyi öğrenmeye çalışmak yorar ve yavaşlatır.

Hangi yol sana uygun? REHBER

Hangisi sana göre?

- Bir fikri baştan sona tek başına yapmak istiyorsan → **fullstack**.
- Telefon uygulaması yapmak hayalinse → **mobil**.
- Mobil için web/JS biliyorsan cross-platform (React Native/Flutter) hızlı giriş sağlar.

Alıştırma 8 dk

Yolu değerlendir:

1. Fullstack ve mobil yollarını kendi hedefinle eşleştir.
2. Mobilde native mi cross-platform mı sana uygun, yaz.
3. Bir uygulama fikrini bir cümleyle tarif et.

BÖLÜM 12

Olmazsa Olmaz Araçlar

Her geliştiricinin bir alet çantası vardır. Bu araçları erken tanımak, öğrenme yolculuğunu çok daha verimli kılar. (Sonraki modülde derinleşeceğiz.)

Temel araçlar

- **Kod editörü / IDE:** kod yazdığın program (en yaygını **VS Code**).
- **Terminal / komut satırı:** komutlarla bilgisayarı yönetme.
- **Git & GitHub:** kodunu sürümleme, yedekleme ve sergileme.
- **Tarayıcı geliştirici araçları:** sayfayı inceleme ve hata ayıklama.

Neden önemli?

- **VS Code** yazım hatalarını yakalar, işini hızlandırır.
- **Git** sayesinde "her şeyi bozdum" anlarından geri dönersin.
- **GitHub** hem yedeğin hem de işverene gösterdiğin **portfolyon**dur.

Şema 12.1 — Her yolda işine yarayan temel araçlar.

İPUCU

En baştan **Git ve GitHub**'ı öğren; çoğu yeni başlayan bunu erteler ama erken öğrenen büyük avantaj kazanır. Her küçük projeni GitHub'a koymak, hem güvenli yedek hem de zamanla büyüyen bir portfolyo demektir.

Hangi yol sana uygun? REHBER

Bu araçlar her yol için ortak:

- Frontend, backend, mobil — hepsinde VS Code, terminal ve Git kullanılır.
- Yani bu araçları öğrenmek "boşa" değildir; her yolda işine yarar.
- Serinin "Geliştiricinin Araçları" modülü bunları ayrıntılı öğretecek.

Alıştırma

10 dk

Araçlarla tanış:

1. VS Code'u (veya bir kod editörünü) bilgisayarına kur.
2. Bir GitHub hesabı aç.
3. Terminali açıp ne işe yaradığını araştır.

BÖLÜM 13

Bir Geliştiricinin Günü

Yazılımcılık sadece "kod yazmak" değildir. Gerçek iş, takım çalışması, planlama ve iletişimle örülüdür. Bu gerçeği bilmek, beklentini doğru kurmanı sağlar.

Tipik bir gün

- Kısa bir **takım toplantısı** (kim ne yapıyor — "daily").
- Görevler üstünde **kod yazma** ve **hata ayıklama**.
- **Kod inceleme** (code review): birbirinin kodunu kontrol etme.
- Soru sorma, dokümantasyon okuma, araştırma.

Çevik (Agile) çalışma

- İş, küçük parçalara bölünür ve **kısa döngülerde** (sprint) yapılır.
- Sürekli geri bildirim ve iyileştirme vardır.
- İletişim ve takım uyumu, tek başına dahi olmaktan daha değerlidir.

Şema 13.1 — Bir geliştiricinin tipik günü.

İPUCU

Yazılımcılığın yarısı **iletişimdir**: net soru sormak, kodunu açıklamak, takımla anlaşmak. Sadece teknik değil, bu "yumuşak becerileri" de geliştirmek seni çok ileri taşır. İyi iletişim kuran orta düzey bir geliştirici, iletişimi zayıf bir uzmandan çoğu zaman daha değerlidir.

Hangi yol sana uygun? REHBER

Bu sana ne anlatır?

- İçe dönük olman sorun değil; ama temel iletişim her rolde gerekir.
- Takım çalışmasını seviyorsan kurumsal yol, bağımsızlığı seviyorsan freelance/ kendi ürün uygun olabilir.
- Her durumda kod inceleme ve geri bildirim öğrenmenin parçasıdır.

Alıştırma 8 dk

İş hayatını canlandır:

1. Bir geliştiricinin gününü kendi cümlelerinle özetle.
2. Hangi kısım (kod, takım, inceleme) sana çekici geliyor, yaz.
3. Geliştirmek istediğin bir "yumuşak beceri" belirle.

BÖLÜM 14

Portfolyo ve İlk Projeler

Bu meslekte seni asıl işe aldıran diploma değil, yapabildiklerinin kanıtıdır. Portfolyo, "ben bunu yapabiliyorum" demenin en güçlü yoludur.

Neden proje?

- Bilgi, ancak **uygulayınca** kalıcı olur.
- Bir proje, onlarca sertifikadan daha çok şey anlatır.
- GitHub'daki projelerin, canlı bir **özgeçmiştir**.

İlk proje fikirleri

- Kişisel tanıtım sayfası (HTML/CSS).
- Yapılacaklar listesi (to-do) uygulaması (JavaScript).
- Basit hesap makinesi veya hava durumu sayfası (API kullanımı).
- Sevdiğin bir konuda küçük bir bilgi sitesi.

Şema 14.1 — Projelerden portfolyoya.

İPUCU

Projelerini "mükemmel olunca paylaşırım" diye bekletme. **Küçük, bitmiş ve yayınlanmış** bir proje, yarım kalmış kusursuz bir projeden çok daha değerlidir. Yaptıkça hem öğrenir hem portfolyonu büyütürsün.

Hangi yol sana uygun? REHBER

İlk projen ne olmalı?

- Frontend yolundaysan → bir kişisel sayfa veya to-do uygulaması.
- Backend yolundaysan → küçük bir veri saklayan basit bir uygulama.
- Hangi yol olursa olsun: **küçük başla, bitir, GitHub'a koy.**

Alıştırma 10 dk

İlk projeni planla:

1. Yapabileceğin küçük bir proje fikri seç.
2. Onu 3 küçük adıma böl.
3. İlk adımı bu hafta yapmayı planla ve GitHub'a koymayı hedefle.

SEVİYE 3

Sektör ve Gerçekler

İşin gerçek yüzü: iş imkânları, maaş ve kariyer basamakları, üniversite bölümü seçimi, CV-GitHub-mülakat, yapay zekâ çağında yazılımcılık ve etik-lisanslar.

BÖLÜM 15

Sektörde İş İmkânları

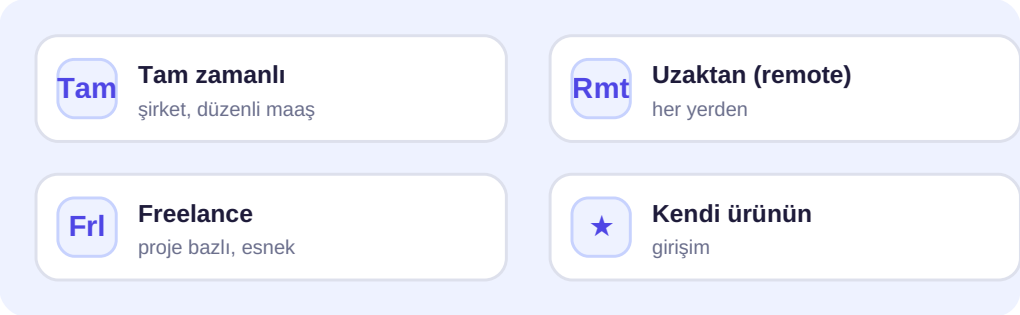
Yazılım, dünya genelinde talebi yüksek bir alandır. Çalışma biçimleri de çeşitlidir: ofis, uzaktan, freelance, hatta yurt dışına uzaktan çalışmak.

Çalışma biçimleri

- **Tam zamanlı (şirket):** düzenli maaş, takım, kariyer basamağı.
- **Uzaktan (remote):** evden veya her yerden; yurt içi/dışı şirketlere.
- **Freelance:** proje bazlı, bağımsız; esnek ama düzensiz gelir.
- **Girişim/kendi ürünün:** kendi yazılımını yapıp satmak.

Talep nerede?

- Web (frontend/backend/fullstack) her zaman yüksek talep görür.
- Mobil, veri/yapay zekâ ve bulut/DevOps hızla büyüyen alanlardır.
- İngilizce, özellikle uzaktan/yurt dışı işlerde büyük avantajdır.



Şema 15.1 — Çalışma biçimleri.

İPUCU

İngilizceyi ihmal etme. Dokümantasyonun çoğu, en güncel kaynaklar ve en iyi uzaktan iş fırsatları İngilizcedir. Teknik İngilizceyi geliştirmek, çoğu zaman fazladan bir programlama dili öğrenmekten daha çok kapı açar.

Hangi yol sana uygun? REHBER

Hangi çalışma biçimi sana uygun?

- Düzen ve takım istiyorsan → tam zamanlı (uzaktan da olabilir).
- Esneklik ve bağımsızlık istiyorsan → freelance (önce deneyim kazan).
- Kendi ürünün hayalinse → fullstack becerisi + girişim.

Alıştırma

8 dk

İş manzaranı çiz:

1. Dört çalışma biçimini kendi tercihinle sırala.
2. Hedeflediğin alanı (web/mobil/veri) belirle.
3. İngilizce seviyeni değerlendirip bir gelişim adımı yaz.

BÖLÜM 16

Maaş, Seviyeler ve Kariyer Basamakları

Yazılımda kariyer basamaklıdır: zamanla daha çok sorumluluk, daha çok değer. Bu basamakları bilmek, hedef koymanı ve ilerlemeni planlamanı sağlar.

Tipik seviyeler

- **Junior (Başlangıç):** temel görevler, mentor desteğiyle öğrenme.
- **Mid (Orta):** bağımsız çalışır, çoğu işi kendi çözer.
- **Senior (Kıdemli):** karmaşık işler, mimari kararlar, başkalarına rehberlik.
- **Lead / Mimar / Yönetici:** takım ve sistem yönetimi.

Ne belirler?

- Maaş; seviye, alan, şehir/ülke, şirket ve uzaktan olup olmamasına göre çok değişir.
- Junior'dan senior'a geçiş yıllarla değil, **beceri ve etkiyle** ölçülür.
- Sürekli öğrenme ve görünür işler (portfolyo, katkı) ilerlemeyi hızlandırır.

Şema 16.1 — Kariyer basamakları; yıllarla değil beceriyle ölçülür.

İPUCU

Maaş rakamlarına takılıp paniğe kapılma; bunlar ülkeye, şehre ve döneme göre çok değişir ve hızla güncellenir. Erken dönemde odağını **beceri kazanmaya** ver; değer ürettikçe maaş zaten gelir. Güncel rakamlar için sektör kaynaklarını araştır.

Hangi yol sana uygun? REHBER

Yolculuğun başında ne yapmalı?

- Hedefin "junior pozisyona hazır olmak" olsun; portfolyo + temel beceriler.
- İlk işte odak: öğrenmek ve katkı vermek; seviye sonra gelir.
- Her yolda (frontend/backend/mobil) bu basamaklar benzer işler.

Alıştırma 8 dk

Hedef koy:

1. Junior olmaya hazır olmak için gereken 3 beceriyi yaz.
2. Kendine gerçekçi bir "ilk iş" hedefi belirle.
3. O hedefe giden bir sonraki adımı not et.

BÖLÜM 17

Üniversite Bölümü Seçimi

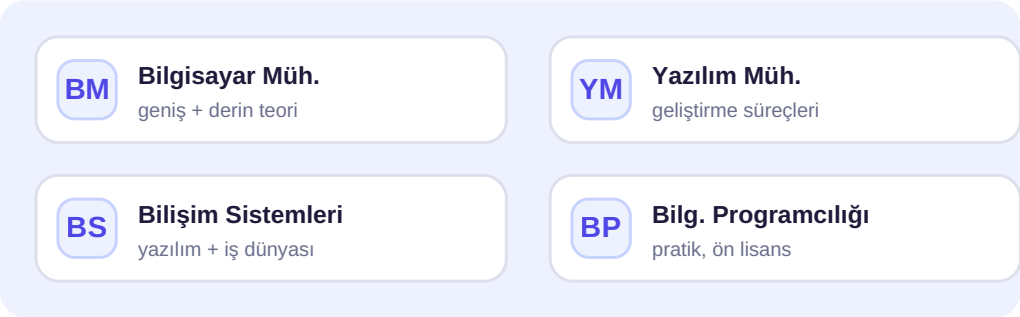
Üniversiteye yöneliyorsan, "hangi bölüm?" önemli bir sorudur. Birkaç bölüm bu alana çıkar; farklarını bilmek doğru seçimi kolaylaştırır.

İlgili bölümler

- **Bilgisayar Mühendisliği:** donanım + yazılım + teori; geniş ve derin.
- **Yazılım Mühendisliği:** yazılım geliştirme süreçlerine odaklı.
- **Bilişim Sistemleri / Yönetim Bilişim Sistemleri:** yazılım + iş/işletme.
- **Bilgisayar Programcılığı (ön lisans):** daha pratik, hızlı iş odaklı.

Bilmen gerekenler

- Bölüm sana **teori, temel ve diploma** verir; pratiği büyük ölçüde **kendi projelerinle** kazanırsın.
- Bölüm adı tek başına yetmez; **kendi çaban** belirleyicidir.
- Üniversiteye gitmeden de bu meslek yapılabilir; ama bazı yollarda diploma kapı açar.



Şema 17.1 — İlgili üniversite bölümleri.

İPUCU

Hangi bölüme girersen gir, **birinci sınıftan itibaren kendi projelerini yap** ve GitHub'a koy. Sadece dersleri geçen değil, derslerin yanında üreten öğrenci mezun olduğunda çok öndedir. Diploma temel atar; portfolyo iş bulur.

Hangi yol sana uygun? REHBER

Hangi bölüm kime?

- Derin teori ve geniş alan istiyorsan → Bilgisayar/Yazılım Mühendisliği.
- Yazılım + iş dünyası ilgini çekiyorsa → Bilişim/Yönetim Bilişim Sistemleri.
- Hızlı, pratik ve iş odaklıysan → Bilgisayar Programcılığı (ön lisans) veya kurs.

Alıştırma 10 dk

Bölüm araştırması:

1. İlgili bölümlerin ders içeriklerini (örnek bir üniversiteden) incele.
2. Sana en uygun bölümü ve nedenini yaz.
3. Bölümden bağımsız, kendin yapacağın ilk projeyi belirle.

BÖLÜM 18

CV, GitHub ve İş Görüşmesi Temelleri

İş bulmak da bir beceridir. İyi bir CV, dolu bir GitHub ve mülakat hazırlığı; becerini fırsata çevirmenin yoludur.

CV ve GitHub

- **CV:** sade, net; becerilerin, projelerin ve (varsa) deneyimin.
- **GitHub:** düzenli projeler, açıklayıcı README dosyaları.
- **Portfolyo sitesi:** projelerini canlı gösteren kişisel sayfa (artı puan).

Teknik mülakat nedir?

- Genelde: temel bilgi soruları + küçük bir **kodlama problemi** + projelerin üzerine sohbet.
- Amaç "her şeyi bilmen" değil, **nasıl düşündüğünü** görmek.
- Bilmediğinde dürüstçe "bilmiyorum ama şöyle araştırırım" demek değerlidir.

Şema 18.1 — Beceriden işe: başvuru yolculuğu.

İPUCU

GitHub'daki her projeye, ne olduğunu ve nasıl çalıştırılacağını anlatan kısa bir **README** ekle. İşverenler kodundan önce README'ni okur; iyi bir açıklama, sıradan bir projeyi bile profesyonel gösterir.

Hangi yol sana uygun? REHBER

Nereden başlamalı?

- Önce 2-3 bitmiş projeyi GitHub'a README'leriyle koy.
- Sonra sade bir CV ve mümkünse bir portfolyo sayfası hazırla.
- Mülakat için temel kavramları ve küçük kod problemlerini tekrar et.

Alıştırma 10 dk

Başvuruya hazırlan:

1. Bir projene açıklayıcı bir README yaz.
2. CV'ne koyacağın 5 beceri ve 2 projeyi listele.
3. Bir teknik mülakatta sorulabilecek 3 temel soruyu tahmin et.

BÖLÜM 19

Yapay Zekâ Çağında Yazılımcılık

Yapay zekâ kod yazmaya yardım ediyor; bu, "yazılımcılık bitti mi?" sorusunu doğuruyor. Kısa cevap: hayır — ama meslek değişiyor ve yeni beceriler önem kazanıyor.

Yapay zekâ neyi değiştiriyor?

- Kod yazmayı **hızlandırıyor**: taslak, örnek, hata açıklaması.
- Öğrenmeyi kolaylaştırıyor: bir kavramı anında açıklatabiliyorsun.
- Ama **ne istediğini bilmeyi**, kararları ve doğrulamayı hâlâ insan yapıyor.

Önem kazanan beceriler

- **Problemi doğru tanımlamak** ve sistemi tasarlamak.
- Yapay zekânın ürettiğini **okuyup doğrulamak** (her zaman doğru değildir).
- Temelleri anlamak: temeli olmayan, yapay zekânın hatasını fark edemez.
- İyi **istem (prompt) yazmak** ve araçları akıllıca kullanmak.

Şema 19.1 — Yapay zekâ + geliştirici iş birliği.

İPUCU

Yapay zekâyı öğrenme aşamasında **dikkatli** kullan: sana kodu hazır verip düşünmeni engelleyebilir. Önce kendin dene, takıldığında açıklama iste. Yapay zekâ temelleri olan için bir hızlandırıcı, temeli olmayan için bir koltuk değneğidir.

Hangi yol sana uygun? REHBER

Bu sana ne anlatır?

- Temelleri öğrenmek artık daha da önemli; yapay zekâ onları ikame etmiyor.
- Yapay zekâyı bir takım arkadaşı gibi gör: hızlandırır ama sorumluluk sende.
- Bu seride ayrı bir "Kodlamada Yapay Zekâ" modülü bunu derinleştirecek.

Alıştırma 8 dk

Dengeyi kur:

1. Yapay zekânın sana yardımcı olabileceği 2 durumu yaz.
2. Yapay zekâya güvenmeden önce neyi doğrulaman gerektiğini belirle.
3. Öğrenirken yapay zekâyı "ne zaman" kullanacağına dair bir kural koy.

BÖLÜM 20

Etik, Telif ve Lisanslar

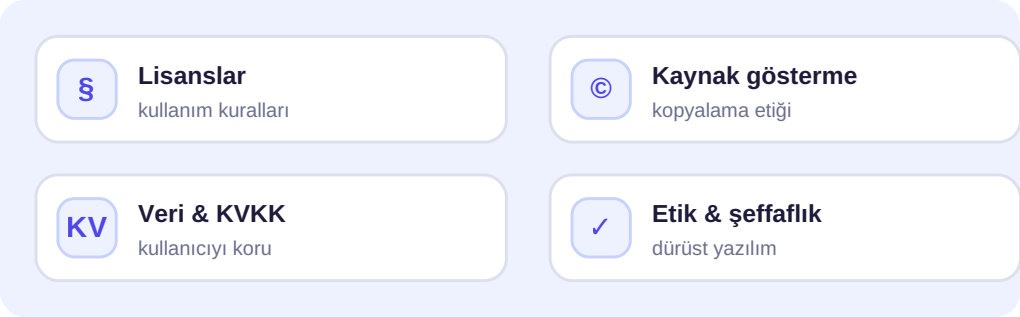
Kod yazmak teknik olduğu kadar sorumluluk da ister. Başkasının kodunu kullanma kuralları, kullanıcı verisinin korunması ve etik davranış; profesyonelliğin parçasıdır.

Açık kaynak ve lisanslar

- İnternetteki kodun çoğu bir **lisansla** gelir; onu nasıl kullanabileceğini lisans belirler.
- Bazı lisanslar serbest kullanım verir, bazıları şartlar koyar (kaynak gösterme, aynı lisansla paylaşma).
- Başkasının kodunu körü körüne kopyalamak hem etik hem hukuki sorun olabilir.

Veri ve etik

- Kullanıcı verisini korumak yasal bir zorunluluktur (**KVKK** ve benzeri).
- Yalnızca gereken veriyi topla, güvenli sakla, izinsiz paylaşma.
- Etik davranış: zarar vermeyen, dürüst ve şeffaf yazılım.



Şema 20.1 — Etik, telif ve veri sorumluluğu.

İPUCU

Bir projede hazır kod/kütüphane kullanırken **lisansına bak**. Çoğu açık kaynak lisans kullanıma izin verir ama kaynak göstermeni isteyebilir. Bunu baştan öğrenmek, ileride hukuki sorunlardan korur.

Hangi yol sana uygun? REHBER

Bu her yolda geçerli:

- Frontend, backend, mobil — hepsinde lisans ve veri sorumluluğu vardır.
- Kullanıcı verisiyle çalışıyorsan KVKK bilinci şarttır.
- Etik ve şeffaflık, uzun vadede itibarını ve kariyerini korur.

Alıştırma 8 dk

Sorumluluğu kavra:

1. Bir açık kaynak lisansını (örn. MIT) araştırıp ne izin verdiğini yaz.
2. Bir uygulamanın hangi kullanıcı verisini toplayabileceğini düşün.
3. O veriyi korumak için 2 kural belirle.

SEVİYE 4

Kendi Yol Haritanı Kur

Planı eyleme dök: kişisel yol haritası, etkili öğrenme sistemi, topluluk ve ağ kurma, güncel kalma, sürdürülebilir ilerleme ve 12 aylık eylem planı.

BÖLÜM 21

Kişisel Yol Haritası Oluşturma

Genel tavsiyeler güzeldir ama asıl işe yarayan, sana özel bir plandır. Hedefini netleştirip onu küçük, ulaşılabilir adımlara bölmek, yolda kalmanı sağlar.

Hedeften plana

1. **Net bir hedef** yaz: "6 ayda basit web siteleri yapabilen biri olmak".
2. Hedefi **aşamalara** böl: önce HTML/CSS, sonra JS, sonra bir proje.
3. Her aşamaya **gerçekçi süre** ve **küçük proje** ekle.
4. İlerlemeni görünür tut (takvim, kontrol listesi).

İyi hedefin özellikleri

- **Belirli:** "yazılım öğrenmek" değil, "to-do uygulaması yapabilmek".
- **Ölçülebilir:** bittiğini nasıl anlayacaksın?
- **Gerçekçi:** günde 1 saatle 1 ayda neyi bitirebilirsin?

Şema 21.1 — Hedeften kişisel yol haritasına.

İPUCU

Büyük hedef bunaltır; **küçük, bitirilebilir adımlar** motive eder. "Yazılımcı olacağım" yerine "bu hafta bir HTML sayfası yapacağım" de. Küçük zaferler birikince büyük hedefe ulaşırsın.

Hangi yol sana uygun?

REHBER

Yol haritanı seçtiğin yola göre kur:

- Frontend hedefi → HTML → CSS → JS → küçük projeler → framework.
- Backend hedefi → algoritma → bir dil → SQL → API → hosting.
- Bu serinin modülleri, planının doğal aşamalarıdır.

Alıştırma

12 dk

Kendi haritanı çiz:

1. Net, belirli bir 6 aylık hedef yaz.
2. Onu 3-4 aşamaya böl.
3. Her aşamaya bir küçük proje ve süre ekle.

BÖLÜM 22

Etkili Öğrenme Sistemi

Nasıl öğrendiğin, ne kadar öğrendiğin kadar önemlidir. İyi bir öğrenme sistemi; doğru kaynak, aktif pratik ve düzenli tekrarla kurulur.

Öğrenme döngüsü

- **Öğren:** bir konuyu kısa ve odaklı çalış.
- **Uygula:** hemen küçük bir kod/proje ile dene.
- **Takıl & çöz:** hatayı araştır, sor, çöz.
- **Tekrar et:** öğrendiğini birkaç gün sonra yeniden kullan.

Kaynak ve not

- Çok kaynağı aynı anda kovalama; **bir iyi kaynağı bitir**.
- Kendi **notlarını** tut (bir not uygulaması veya GitHub).
- Öğrendiğini birine (veya kendine) **anlatmak**, kalıcılığı artırır.

Şema 22.1 — Etkili öğrenme sistemi.

İPUCU

Pasif izlemek (video seyretmek) öğrenmek değildir; asıl öğrenme **kendin kod yazınca** olur. Her izlediğin/okuduğun konudan sonra "şimdi bunu kendim yapayım" de. Bir saat aktif pratik, üç saat pasif izlemekten çok daha öğreticidir.

Hangi yol sana uygun? REHBER

Sistemini yoluna uyarla:

- Hangi yolda olursan ol döngü aynı: öğren → uygula → çöz → tekrar.
- Frontend'de "uygula" anında ekranda görünür; backend'de küçük çıktılarla test et.
- Notların ve projelerin zamanla en değerli kaynağın olur.

Alıştırma 10 dk

Sistemini kur:

1. Kendine bir öğrenme döngüsü (öğren-uygula-çöz-tekrar) yaz.
2. Bir ana kaynak ve bir not yöntemi seç.
3. Bu hafta öğrendiğin bir şeyi nasıl tekrar edeceğini planla.

BÖLÜM 23

Topluluk ve Ağ Kurma

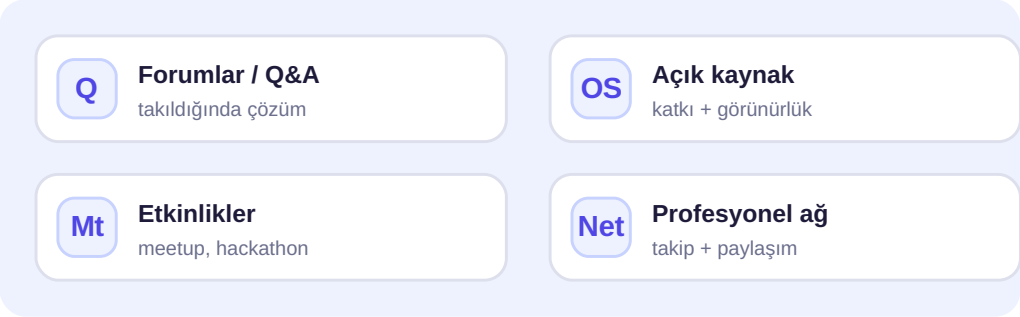
Yalnız öğrenmek zordur ve yavaştır. Bir topluluğun parçası olmak; sorularına yanıt, motivasyon ve fırsatlar getirir. Yazılım dünyası paylaşım üzerine kuruludur.

Nerede topluluk var?

- **Soru-cevap siteleri** ve forumlar: takıldığında çözüm bulursun.
- **Açık kaynak projeler:** küçük katkılarla hem öğrenir hem tanınırsın.
- **Yerel/çevrimiçi etkinlikler:** meetup, hackathon, atölyeler.
- **Sosyal/profesyonel ağlar:** diğer geliştiricileri takip et, paylaş.

Ağ kurmanın gücü

- Çoğu iş fırsatı **tanıdık/ağ** üzerinden gelir.
- Bir mentor, ayları kısaltabilir.
- Öğrendiğini paylaşmak, hem başkasına yardım eder hem seni pekiştirir.



Şema 23.1 — Topluluk ve ağ: öğrenmeyi hızlandırır.

İPUCU

Soru sormaktan çekinme; ama önce kendin araştır ve sorunu **net** anlat (ne yaptın, ne bekledin, ne oldu, hata mesajı ne?). İyi sorulmuş bir soru, hem hızlı yanıt alır hem de seni topluluğa saygın biri yapar.

Hangi yol sana uygun? REHBER

Topluluk her yol için değerli:

- Hangi alanda olursan ol, o alanın bir topluluğu vardır.
- Açık kaynağa küçük katkılar, portfolyona ve ağına ikisine birden yarar.
- İçe dönük olsan bile çevrimiçi topluluklar düşük baskılı bir başlangıçtır.

Alıştırma 8 dk

Ağını başlat:

1. İlgi alanınla ilgili bir topluluk/forum bul ve katıl.
2. İyi bir soru nasıl sorulur, bir taslak yaz.
3. Takip edeceğin 3 geliştirici/kaynak belirle.

BÖLÜM 24

Sürekli Güncel Kalmak

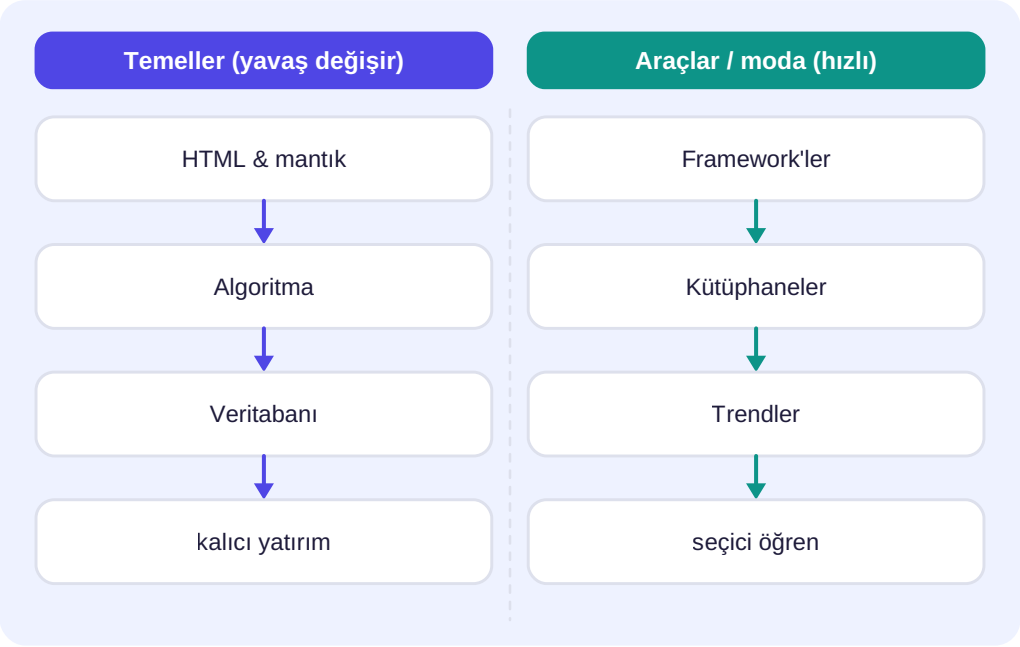
Teknoloji hızla değişir; bugünün popüler aracı yarın yenilenebilir. İyi haber: temelleri sağlam olan, yeniliği kolayca yakalar. Asıl beceri, "öğrenmeyi öğrenmek"tir.

Güncel kalma yolları

- Güvenilir kaynakları (blog, bülten, geliştirici hesapları) takip et.
- **Resmî dokümantasyon** okumayı öğren; en doğru ve güncel kaynak odur.
- Yeni bir aracı duyunca paniğe kapılma; "gerçekten gerekli mi?" diye sor.

Temel vs moda

- **Temeller** (HTML, mantık, algoritma, veritabanı) yavaş değişir; bunlara yatırım kalıcıdır.
- **Araç/framework modaları** daha hızlı döner; temeli olan bunları kolay öğrenir.
- Her yeni şeyi kovalamak yerine, **derinleşeceğin birini seç**.



Şema 24.1 — Temellere yatırım kalıcı; araç modaları hızlı döner.

İPUCU

Her yeni framework çıktığında öğrenmeye koşma; bu, "moda kovalama" yorgunluğuna yol açar. **Temellere** sağlam yatırım yap; temeli olan, yeni bir aracı birkaç günde çözer. Dokümantasyon okumayı öğrenmek, en kalıcı becerindir.

Hangi yol sana uygun?

REHBER

Güncel kalmak her yolda gerekir:

- Frontend araçları en hızlı değişen alandır; temeller (HTML/CSS/JS) kalıcıdır.
- Backend ve veritabanı daha yavaş değişir; bilgi daha uzun ömürlüdür.
- Hangi yol olursa olsun: temele yatırım + seçici güncelleme.

Alıştırma

8 dk

Güncel kalma planı:

1. Takip edeceğin 3 güvenilir kaynak seç.
2. Bir resmî dokümantasyon sayfasını açıp 10 dakika okumayı dene.
3. "Yeni araç gördüğümde ne soracağım?" diye bir kural yaz.

BÖLÜM 25

Tükenmişlikten Kaçınmak ve Sürdürülebilir İlerleme

Bu uzun bir yolculuk; sprint değil, maraton. Hızlı başlayıp tükenmek yerine, sürdürülebilir bir tempoyla ilerlemek seni gerçekten hedefe taşır.

Tükenmişliğin işaretleri

- Sürekli yorgunluk, motivasyon kaybı, "hiç ilerlemiyorum" hissi.
- Karşılaştırma tuzağı: başkalarının hızıyla kendini ezmek.
- Mola vermeden, sınırsız çalışıp verimini düşürmek.

Sürdürülebilir ilerleme

- **Düzenli ve ölçülü:** her gün az, ama sürekli.
- **Mola ve dinlenme** öğrenmenin parçasıdır; beyin molada pekiştirir.
- **Küçük zaferleri kutla:** ilerlemeni görünür kıl.
- **Kendine şefkat:** herkes takılır; bu başarısızlık değil, süreçtir.

Şema 25.1 — Sürdürülebilir, maraton temposu.

İPUCU

İlerlemeni başkalarıyla değil, **bir ay önceki kendinle** kıyasla. Herkesin başlangıç noktası ve temposu farklı. Düzenli, ölçülü ve kendine şefkatli bir tempo; parlayıp sönen bir hızdan çok daha uzağa götürür.

Hangi yol sana uygun?

REHBER

Sürdürülebilirlik her yol için şart:

- Hangi alanı seçersen seç, yol uzundur; tempo önemlidir.
- Tükenmek, en yetenekliyi bile durdurur; süreklilik kazanır.
- Sağlıklı bir tempo, uzun vadede en hızlı ilerlemedir.

Alıştırma

8 dk

Temponu ayarla:

1. Sürdürebileceğin haftalık bir çalışma temposu belirle.
2. Molalarını ve dinlenmeni plana yaz.
3. İlerlemeni nasıl görünür kılacağını (kayıt/kutlama) belirle.

BÖLÜM 26

Bitirme: 12 Aylık Eylem Planı

Tüm öğrendiklerini birleştirip kendine somut bir 12 aylık plan çıkarıyorsun. Bu plan, bu rehberin sana bıraktığı en değerli çıktı: nereye, nasıl gideceğinin haritası.

Örnek 12 aylık iskelet

- **Ay 1-2:** HTML & CSS — basit, responsive sayfalar + GitHub.
- **Ay 3-4:** JavaScript & Algoritma — etkileşimli küçük uygulamalar.
- **Ay 5-7:** Seçtiğin yol — bir backend dili + Veritabanı, ya da framework + mobil.
- **Ay 8-10:** Daha büyük bir proje (portfolyo parçası) + hosting ile yayına alma.
- **Ay 11-12:** Portfolyo cilalama, CV/GitHub, ilk başvurular/freelance.

Planı yaşatmak

1. Her ay bir **bitmiş çıktı** hedefle (sayfa, uygulama, proje).
2. İlerlemeni görünür tut; takıldığında topluluğa sor.
3. Yapay zekâyı öğrenmeyi köreltmeden, akıllıca kullan.
4. Tempoyu sürdürülebilir tut; her ay planı gözden geçir.

Şema 26.1 — Örnek 12 aylık eylem planı iskeleti.

İPUCU

Bu plan taşa kazınmış değil; **yürüdükçe güncellenecek** bir pusuladır. Önemli olan mükemmel plan değil, **başlamak ve devam etmek**. Bir yıl sonra geriye baktığında, bugün attığın küçük ilk adıma şaşıracaksın.

Hangi yol sana uygun? REHBER

Planını yoluna göre uyarla:

- Frontend hedefi → HTML/CSS/JS ağırlıklı + framework + projeler.
- Backend hedefi → algoritma + dil + SQL + API + hosting.
- Bu serinin modülleri, planının her ayına bir konu sağlar.

Alıştırma

15 dk

Kendi 12 aylık planını yaz:

1. Seçtiğin yolu (frontend/backend/fullstack/mobil) belirle.
2. 12 ayı 5-6 aşamaya böl ve her aşamaya bir çıktı yaz.
3. İlk ayın haftalık planını detaylandır.
4. Planı görünür bir yere (takvim/dosya) koy ve bu hafta başla.

EK

Terimler Sözlüğü ve Yol Özeti

Yazılım dünyasında en sık karşılaşacağın temel terimler. Bu sayfayı bir başvuru kaynağı olarak saklayabilirsin.

Frontend	`Kullanıcının gördüğü önyüz (HTML/CSS/JS)`	Backend	`Sunucu tarafı mantık ve veri`
Fullstack	`Hem önyüz hem arkayüz`	HTML	`Sayfa yapısı (işaretleme)`
CSS	`Sayfa görünümü (biçimleme)`	JavaScript	`Web'de etkileşim/mantık dili`
Veritabanı	`Bilgilerin saklandığı sistem`	API	`Yazılımların birbiriyle konuşma yolu`
Framework	`Hazır iskelet/araç seti`	Repo / commit	`GitHub'da proje / kayıt noktası`
Bug / debug	`Hata / hatayı bulup düzeltme`	IDE / editör	`Kod yazılan program (örn. VS Code)`

En güvenli başlangıç yolu ÖZET

Kararsızsan şu sırayla başla: **HTML → CSS → JavaScript**. Bu üçü hızlı görsel sonuç verir, motive eder ve hangi yola gidersen git (frontend, backend, fullstack, mobil) işine yarar. Birkaç hafta pratik sonra neyi sevdiğini netleştirir; sonra bu serideki ilgili modülle (Algoritma, bir backend dili, Veritabanı, Mobil…) derinleşirsin.