E-KİTAP

WEB GELİŞTİRME TERİMLERİ

HAYAL EDEREK ANLAMA

ENES TOY

İÇİNDEKİLER

1. Web’in Temelleri
 1.1. İnternetin Çalışma Prensibi
 1.2. IP ve Port Kavramları
 1.3. İki Cihaz Arasında Socket Bağlantısı Kurmak (Telnet)
 1.4. TCP/IP Nedir?
 1.5. HTTP, HTTPS & HTTP Metodları
 1.6. URL Yapısı
 1.7. Dns ve Alan Adı
 1.8. Sunucu ve İstemci Kavramları
 1.9. Tarayıcıların Çalışma Mantığı

2. Web Teknolojilerine Giriş
  2.1. HTML
  2.2. XML
  2.3. CSS
  2.4. JavaScript
  2.5. DOM & DOM Manipülasyonu
  2.6. AJAX & Veri Alışverişi

3. Frontend Geliştirme Teknikleri
 3.1. CSS Framework’leri ve Utility-First Yaklaşımı
 3.2. Frontend Frameworkleri
 3.3. MVVM & Virtual DOM
 3.4. Single Page Application (SPA)
 3.5. Router & State Management
 3.6. MVP, MVC ve MVVM Tasarım Desenleri

4. Backend Geliştirme Temelleri
 4.1. PHP
 4.2. Apache & Nginx
 4.3. Node.js
 4.4. CRUD Operasyonları
 4.5. API: RESTful & GraphQL
 4.6. cURL & Form
 4.7. WebSockets
 4.8. JWT; OAuth, OpenID, SAML
 4.9. MVC ve Backend Mimarileri
 4.10. Backend Frameworkleri ve Tasarım Prensipleri
 4.11. RESTful API ve JSON Yönetimi

İÇİNDEKİLER

5. Veritabanı Yönetimi
 5.1. Veritabanı ve SQL
 5.2. SQL ve NoSQL Farkları
 5.3. ORM
 5.4. Veritabanı Optimizasyonu
 5.5. Redis & Cache

6. Web Güvenliği ve Performans Optimizasyonu
 6.1. Session & Cookie Yönetimi
 6.2. CORS, CSRF, XSS
 6.3. HTTPS, SSL/TLS
 6.4. API Güvenliği
 6.5. RegExp ve Güvenlik
 6.6. Performans İyileştirme (Lazy Loading, Minification)

7. İşletim Sistemleri ve Sistem Yönetimi
 7.1. İşletim Sistemleri (Windows, Linux, macOS)
 7.2. UNIX & GNU/Linux
 7.3. Terminal Komutları
 7.4. Localhost & Sunucu Tarafı Kodlama
 7.5. SSH

8. Domain, Hosting ve DNS Yönetimi
 8.1. Domain Alım
 8.2. Domain Yönlendirme & NS
 8.3. Hosting Türleri (Paylaşımlı, VPS, Cloud)
 8.4. Sunucu Seçimi
 8.5. cPanel

9. DevOps ve Deployment Süreçleri
 9.1. Git & Versiyon Kontrolü
 9.2. CI/CD
 9.3. Infrastructure as Code (IaC)
 9.4. Docker & Containerization
 9.5. Kubernetes & Container Orchestration
 9.6. Load Balancing, CDN & Cloud
 9.7. Monitoring & Logging

İÇİNDEKİLER

10. Paralel ve Dağıtık Sistemler
 10.1. Thread & Process
 10.2. Senkron & Asenkron Programlama
 10.3. Multithreading, Concurrency, Parallelism
 10.4. Message Queue
 10.5. Veri Tutarlılığı (CAP Teoremi, Saga Pattern)

11. Yazılım Mimarileri ve Pratikleri
 11.1. Monolitik vs. Mikroservis
 11.2. Event-Driven Architecture (EDA)
 11.3. Hexagonal & Clean Architecture
 11.4. CQRS
 11.5. SOLID Prensipleri
 11.6. Dependency Injection

12. Yazılım Testleri ve Kalite Güvencesi
 12.1. Unit, Integration, End-to-End Testler
 12.2. Test Framework’leri (Jest, Mocha, JUnit, PyTest)
 12.3. Mocking & Dependency Injection

13. Gelişmiş Yazılım Pratikleri
 13.1. Test-Driven Development (TDD)
 13.2. Debugging
 13.3. Design Patterns (Singleton, Factory, Observer)
 13.4. Clean Code

14. Teşekkürler

15. Telif Hakları ve Görsel Kullanım Bildirimi

ÖNSÖZ

Sevgili Okuyucu,

Web dünyası, yalnızca görünmez ağlardan ve karmaşık teknolojilerden ibaret değildir; aynı zamanda **hayal gücümüzü** harekete geçiren, sürekli gelişen bir evrendir. **Web geliştirme terimlerini** soğuk teknik detaylardan uzaklaştırarak, **günlük yaşamdan örneklerle** ve **yaratıcı anlatımlarla keşfetmek**, öğrenme sürecini çok daha keyifli hale getirir.

Bu yolculukta, web'in temel yapı taşlarından başlayarak, internetin nasıl çalıştığını, web teknolojilerinin ve protokollerinin ardındaki prensipleri adım adım ele alacağız. Kitabın içeriği, web geliştirmeye girişten **frontend** ve **backend** teknolojilerine, **veritabanı yönetiminden DevOps süreçlerine** kadar geniş bir yelpazeyi kapsıyor. Her bölüm, kavramları somut örneklerle pekiştirerek, teknik bilgileri daha anlaşılır ve akılda kalıcı hale getirmeyi hedefliyor.

Örneğin, "**İnternetin Çalışma Prensibi**" bölümünde, bir web sitesine erişim sürecini keyifli bir yolculuğa benzeterek anlatıyoruz. **URL** ile başlayan bu yolculukta, **istemci** ve **sunucu** ilişkilerini, protokollerin nasıl devreye girdiğini ve verinin nasıl iletildiğini adım adım keşfedeceksiniz. Bu anlatım tarzı, karmaşık sistemleri daha **basit ve eğlenceli** hale getirerek öğrenme sürecinizi kolaylaştıracak.

İster web geliştirmeye yeni adım atan biri olun, ister deneyimli bir profesyonel, bu kitap her seviyeden okuyucuya hitap edecek şekilde hazırlandı. **Amacımız**, web teknolojilerini sadece teknik bir bilgi birikimi olarak değil, aynı zamanda **yaratıcı düşünceyi** besleyen bir rehber olarak sizlere sunmak.

Web'in temellerinden başlayarak **modern web geliştirme** pratiklerine uzanan bu **keşif yolculuğuna** hazır mısınız? Hayal gücünüzü kullanarak, karmaşık terimleri ve teknolojileri daha anlaşılır hale getireceğiniz bu serüvende sizlere keyifli okumalar dilerim!

Enes Toy

Web’in Temelleri

İnternetin Çalışma Prensibi

İnternet, dünya çapında milyarlarca cihazın birbirine bağlanmasını sağlayan bir ağdır. Bu cihazlar, birbirleriyle veri paylaşarak iletişim kurar. İnternetteki bilgiler, sunucularda saklanır ve tarayıcılar bu bilgileri kullanıcıya iletir. İnternetteki veriler, özel protokoller (**HTTP, TCP/IP** gibi) kullanılarak güvenli bir şekilde iletilir ve doğru cihazlara ulaşır.

✨Hayal Ederek Anlama

Düşün ki, internette bir şeyi aradığında aslında bir yolculuğa çıkıyorsun. Bu yolculuğun bir başlangıcı, bir ortası ve bir sonu var. İşte internetin çalışma prensibini bir yolculuğa benzetebiliriz:

1. **Yolculuk Başlıyor - Web Sitesi Adresi (URL)**
    - İnternete bağlandığında, tıpkı bir yola çıkmak gibi, bir hedef belirlersin. Bu hedef genellikle bir web sitesi adresi (URL) olur. Bu adres, gitmek istediğin yeri belirtir.
2. **Yolcu ve Yol Arkadaşı - İstemci ve Sunucu**
    - İstemci (senin bilgisayarın, telefonun ya da tarayıcın) yolculuğa başlar. Ama tek başına gitmez, bir yol arkadaşı gerekir. Bu yol arkadaşı, sunucudur. Sunucu, bilgileri depo eden bir yer gibidir.
    - Yani senin bilgisayarın, tarayıcı aracılığıyla doğru sunucuya "Merhaba, şuradaki siteyi alabilir miyim?" diye sorar. Sunucu, bilgileri saklayan ve talebine cevap veren yer gibidir.
3. **Yolculuk ve Veri İletimi - Protokoller**
    - Bu yolculuk, bir takım kurallar ve yollarla yapılır. Bu yollara protokoller denir. En yaygınları, HTTP ve TCP/IP'dir. Bu kurallar sayesinde bilgiler kaybolmaz, bozulmaz ve doğru bir şekilde varış noktasına ulaşır.
    - Yani tarayıcı, sunucudan bir şey istediğinde, protokoller sayesinde bu bilgi güvenli bir şekilde iletilir.
4. **Varış Noktasına Ulaşma - Web Sitesini Görüntülemek**
    - Sonunda, bilgisayarın sunucudan aldığı verileri işler ve sana web sitesini gösterir. Bu veriler metin, resimler ve videolar gibi çeşitli içerikler olabilir.
    - İnternetteki her şey, aslında bir veri paketi ve senin bilgisayarın, bu paketleri işleyip görüntülemeni sağlar.

Özetle:

İnternetin çalışma prensibini, bir bilgi yolculuğu olarak düşünebilirsin. Herhangi bir web sitesi adresini girdiğinde, bilgisayarın, doğru sunucuya gitmek için belirli protokoller üzerinden bilgiyi hızlı ve güvenli bir şekilde alır ve sonunda web sitesini ekranda gösterir. Bu karmaşık yolculuk, görünmeyen ama çok önemli bir sistemle çalışır ve her şeyin düzgün işlemesini sağlar.

Web’in Temelleri

IP Nedir?

IP (Internet Protocol) adresi, cihazların internet veya yerel ağlar üzerinden birbirleriyle iletişim kurmasını sağlayan benzersiz bir sayısal kimliktir. Bir nevi cihazların internet üzerindeki adresidir.

IP Adres Türleri:

1. **IPv4 (Internet Protocol Version 4)**
   - 32 bit uzunluğundadır (örn: 192.168.1.1).
   - 4 adet 0-255 arasında değişen sayıdan oluşur.
   - Toplamda yaklaşık 4.3 milyar adres kapasitesine sahiptir.
2. **IPv6 (Internet Protocol Version 6)**
   - 128 bit uzunluğundadır (örn: 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
   - Çok daha fazla adres kapasitesine sahiptir (trilyonlarca cihaz için yeterli).

IP Adresi Çeşitleri:

1. **Public (Genel) IP**
   - İnternete bağlı cihazların dış dünyadan erişebildiği IP’dir.
   - İnternet Servis Sağlayıcıları (ISS) tarafından atanır.
2. **Private (Özel) IP**
   - Yerel ağ içinde kullanılan, dışarıdan erişilemeyen IP’dir.
   - **Örnekler:** 192.168.1.1, 10.0.0.1
3. **Static (Statik) ve Dynamic (Dinamik) IP**
   - **Statik IP:** Değişmeyen sabit bir IP adresidir. Web sunucuları ve ağ yönetimi için idealdir.
   - **Dinamik IP:** İnternet Servis Sağlayıcıları tarafından belirli aralıklarla değiştirilen IP’dir.

💡 IP Adresi Ne İşe Yarar?

- Cihazların internete veya bir ağa bağlanmasını sağlar.
- Web sitelerine erişirken, veri paketlerinin doğru adrese yönlendirilmesine yardımcı olur.
- Ağ güvenliği, VPN, sunucu yönetimi gibi birçok alanda kullanılır.



Web’in Temelleri

✨Hayal Ederek Anlama

Bir gün yeni bir mahalleye taşındığını hayal et. Her evin kendine ait bir adresi var, değil mi? İşte IP adresi de internet üzerindeki cihazların adresidir.

IP Adresi Ne İşe Yarar?

Bilgisayarlar, telefonlar ve diğer cihazlar internet üzerinden birbirleriyle konuşmak için bir adrese ihtiyaç duyar. Tıpkı birine mektup göndermek için adresini bilmemiz gerektiği gibi, bilgisayarlar da birbirlerine veri göndermek için IP adreslerini kullanır.

İki Tür IP Adresi Vardır:

Ev Adresi (Özel IP):

Evde kullandığın Wi-Fi’ye bağlanan telefon, bilgisayar gibi cihazların bir iç adresi vardır. Bu adres, sadece evin içinde kullanılır. (Örneğin: 192.168.1.1)

Sokak Adresi (Genel IP):

İnternet dünyasında seni tanımlayan bir adres vardır. Bu adres, dışarıdaki herkesin görebileceği adrestir ve internet servis sağlayıcın (Türk Telekom, TurkNet gibi) bunu sana verir.

Sabit mi, Değişken mi?

- **Sabit Adres (Statik IP)**: Hep aynı kalan bir adres.
- **Değişken Adres (Dinamik IP):** Arada değişen bir adres.

Özetle

IP adresi, cihazların internette nerede olduğunu gösteren bir numaradır. Tıpkı ev adresi gibi, bilgisayarlar da birbirlerini bu numaralar sayesinde bulur ve iletişim kurar.

Ek Bilgi

Ortak İnternet Kullanımı (Modem Üzerinden Wi-Fi veya Kablolu Bağlantı) → Aynı IP
Eğer apartmandaki herkes aynı internet sağlayıcısından (aynı modem/router üzerinden) internete bağlanıyorsa, dış dünyaya çıkarken aynı genel (public) IP adresini kullanır.

💡 Örnek:

- Vatan Apartmanı (Modemin genel IP’si: 85.120.45.32)
- Daire 5 ve Daire 6’daki kişiler aynı Wi-Fi’ye bağlı → Aynı IP adresini kullanıyorlar.
- Ama cihazlar arasında fark yaratmak için her cihaza yerel (private) IP atanır:
  - Daire 5’teki telefonun yerel IP’si: 192.168.1.10
  - Daire 6’daki bilgisayarın yerel IP’si: 192.168.1.11

Yani dış dünyaya çıkarken aynı genel IP adresini kullanıyorsunuz, ama yerel ağ içinde cihazlarınızın **farklı özel IP adresleri** var. Eğer modemleriniz farklıysa IP adresleriniz de farklı olur!

Her modem, internet servis sağlayıcısından (ISS) kendine özel bir genel (public) IP adresi alır. Bu yüzden farklı modemler farklı IP adresleri kullanır. Yani Daire 5 ve Daire 6’da farklı modemler varsa, internete çıktığınızda IP adresleriniz de farklı olur.

Web’in Temelleri

Port Nedir?

Port, bir bilgisayardaki veya sunucudaki farklı uygulamaların ve servislerin iletişim kurmasını sağlayan sanal kapılardır. IP adresi cihazın adresini belirlerken, port numarası hangi hizmetin veya uygulamanın kullanılacağını gösterir.

Örneğin:

- **80 numaralı port:** Web sitelerini görüntülemek için (HTTP) kullanılır.
- **443 numaralı port:** Güvenli web trafiği (HTTPS) için kullanılır.
- **21 numaralı port:** Dosya aktarımı (FTP) için kullanılır.

✨Hayal Ederek Anlama

Bir apartmanda birçok daire olduğunu düşün. IP adresi bu apartmanın adresi gibidir. Ama binaya geldiğinde hangi daireye gideceğini bilmen gerekir, değil mi? İşte portlar da daire numarası gibi çalışır!

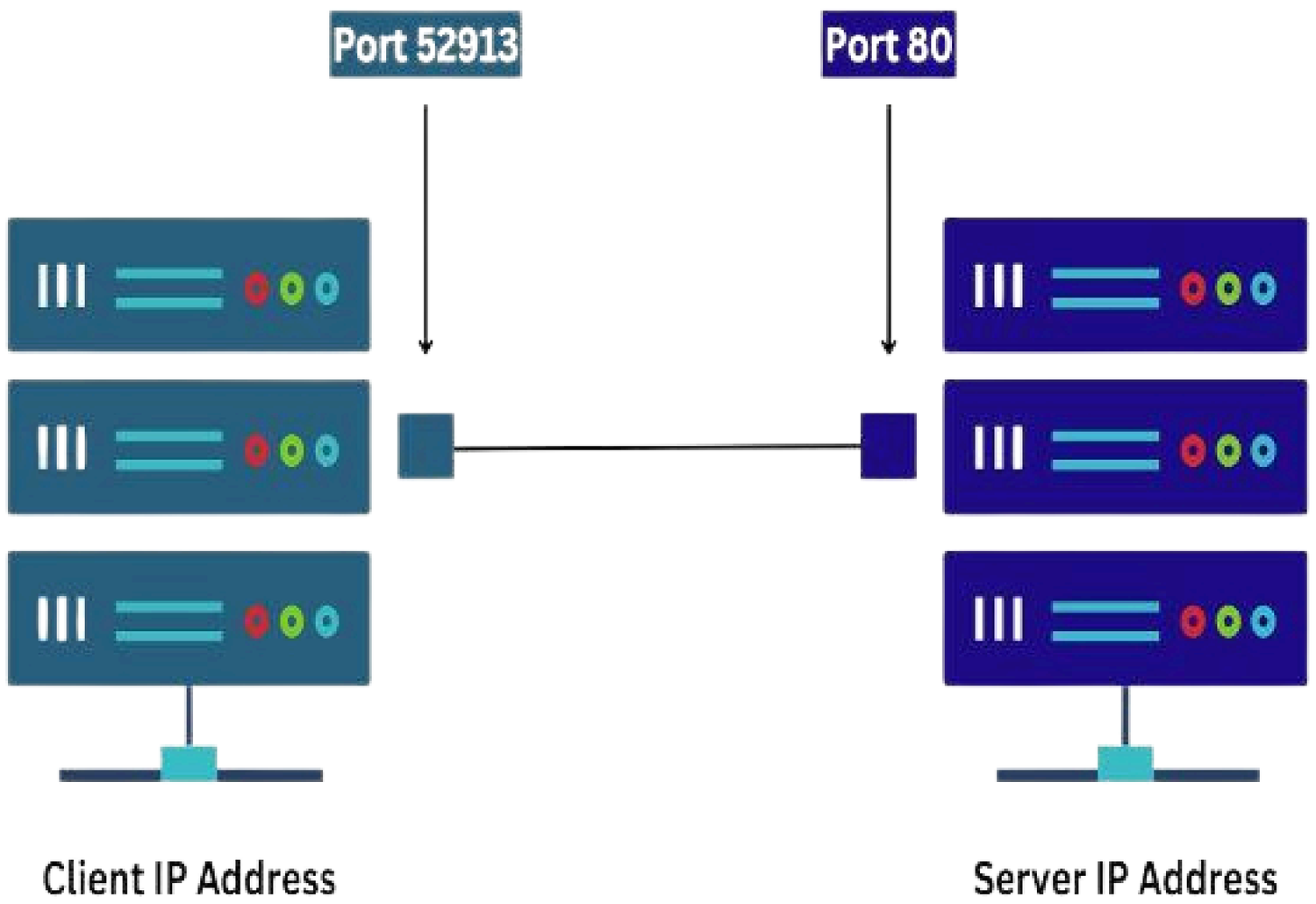
💡 Örnek:

Diyelim ki apartmana kargo geliyor.

- 80 numaralı dairede oturan Ali, mektupları (web sitelerini) alıyor.
- 21 numaralı dairede oturan Ayşe, dosya gönderip alıyor (FTP kullanıyor).
- 443 numaralı dairede oturan Mehmet, güvenli kargoları teslim alıyor (HTTPS).

Sonuç:

Bilgisayar ve internet dünyasında portlar, hangi verinin nereye gitmesi gerektiğini belirler. Bir IP adresi tek başına yeterli olmaz, port numarasıyla hangi hizmetin kullanılacağı anlaşılır.



Web’in Temelleri

İki Cihaz Arasında Socket Bağlantısı Kurmak (Telnet)

Socket, iki cihazın (bilgisayar, sunucu, vs.) birbirleriyle ağ üzerinden iletişim kurmasını sağlayan bir mekanizmadır. Telnet, bu iletişimi sağlamak için kullanılan eski ama hâlâ bazı yerlerde kullanılan bir protokoldür.

Nasıl Çalışır?

1. Bir cihaz sunucu (server) gibi davranır ve bir port üzerinden dinlemeye başlar.
2. Diğer cihaz istemci (client) olur ve bu porta bağlanarak veri gönderip alır.
3. Telnet kullanarak bir cihazın belirli bir portuna bağlanabilir ve komut gönderebilirsin.

💻 **Örnek Kullanım:** telnet 192.168.1.10 80
Bu komut, 192.168.1.10 IP adresindeki cihazın 80 numaralı portuna bağlanmaya çalışır.

✨Hayal Ederek Anlama

Bir tel konuşması yaptığını düşün! 📞
Socket = İki kişi arasındaki telefon hattı
Telnet = Telefon çevirme işlemi

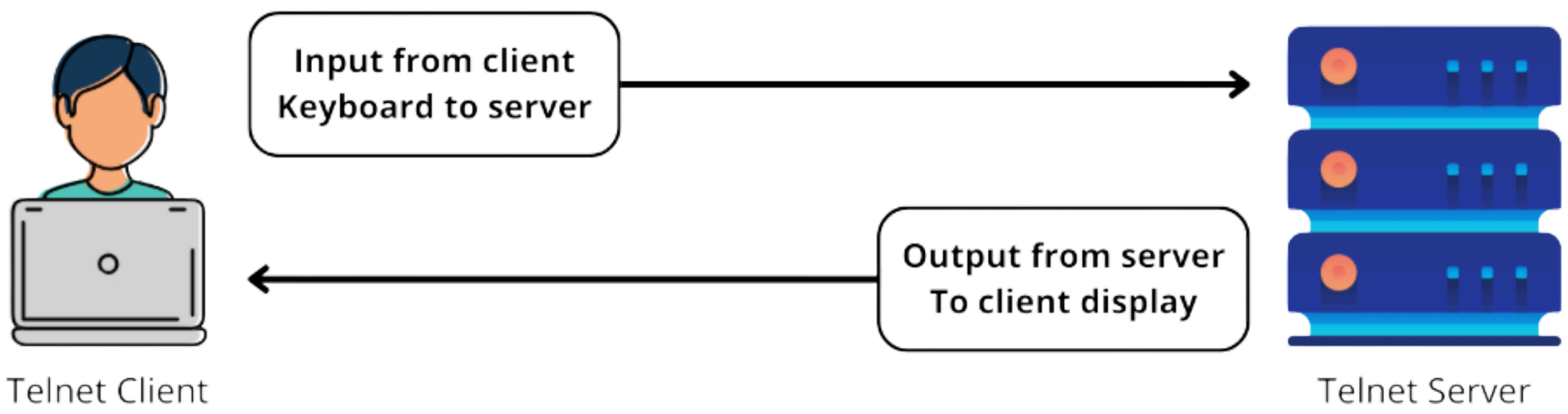
Nasıl Çalışır?

1. Ali (İstemci), telefonu eline alır ve Ayşe’yi (Sunucu) aramak ister.
2. Ayşe, telefonunu açmış ve bir çağrı bekliyordur.
3. Ali numarayı çevirir (IP adresi + Port numarası girer).
4. Ayşe açarsa, ikisi de konuşmaya başlar (veri alışverişi yapılır).

Eğer Ayşe telefonunu açmamışsa ya da numara yanlışsa, bağlantı kurulamaz.

Sonuç:

Socket, iki cihazın birbirine bağlanıp konuşmasını sağlar. Telnet ise bu bağlantıyı kurmak için kullanılan bir araçtır!



Web’in Temelleri

TCP/IP Nedir?

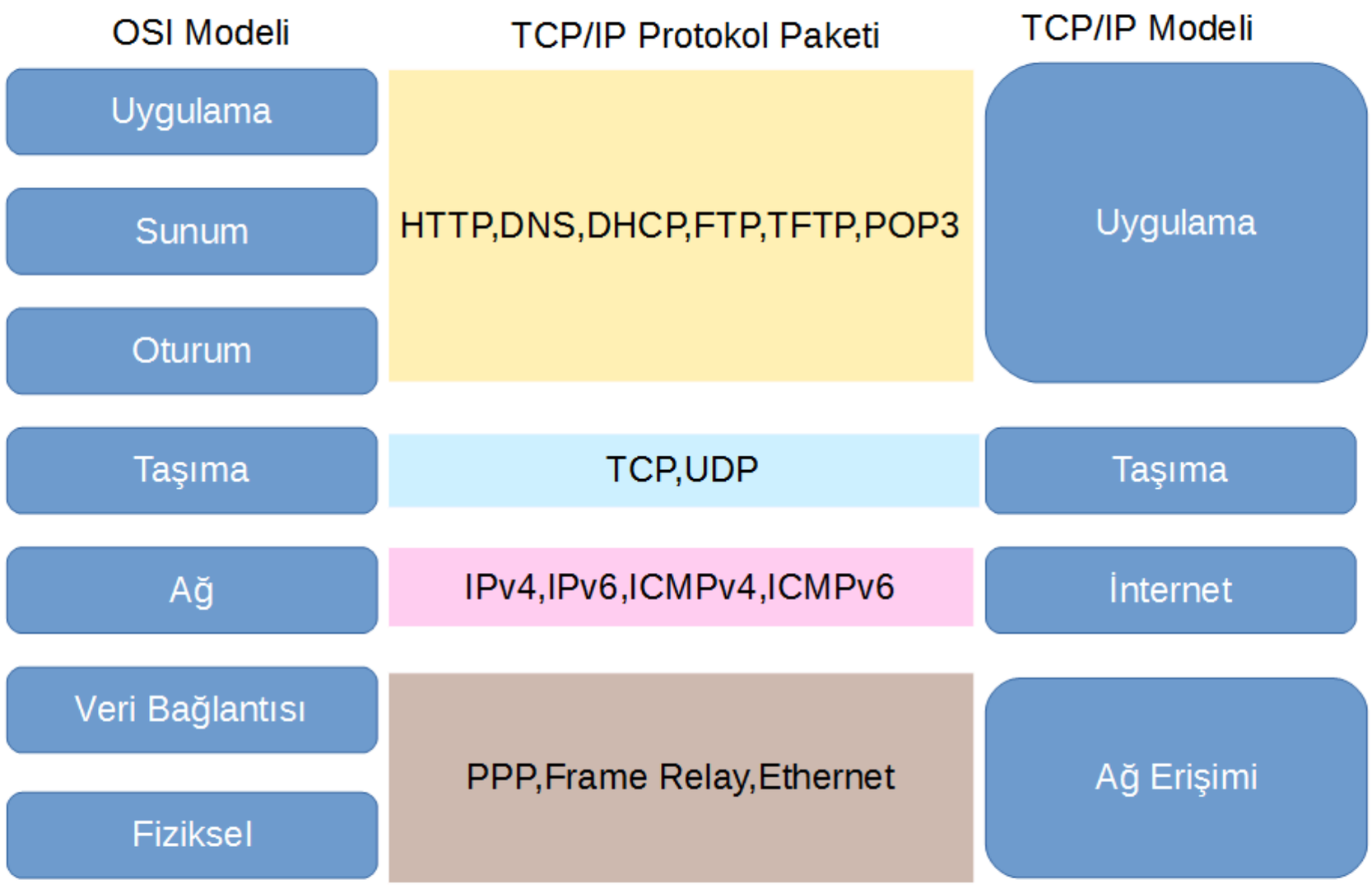
TCP/IP (Transmission Control Protocol / Internet Protocol), yani İletim Kontrol Protokolü / İnternet Protokolü, bilgisayarların ve diğer cihazların internet üzerinden birbiriyle iletişim kurmasını sağlayan temel protokoller bütünüdür. Bu protokol seti, verilerin doğru ve güvenli bir şekilde iletilmesini sağlar.

✨Hayal Ederek Anlama

Diyelim ki senin en iyi arkadaşın farklı bir şehirde yaşıyor ve ona bir hediye göndermek istiyorsun. Hadi bunu adım adım düşünelim:

1. **Hediyeyi paketlemek** 🎁
   - Hediyeni bir kutuya koyuyorsun ve içine bir not ekliyorsun.
   - Adresini düzgünce yazıyorsun ki yanlış yere gitmesin.
   - İşte burada IP (İnternet Protokolü) devreye giriyor. Çünkü IP, verinin (hediyenin) doğru adrese ulaşmasını sağlar.
2. **Kargoya vermek** 🚚
   - Kargo firması paketi alıyor ama bazen tek bir araçla götüremeyebilir. Paket önce bir şehre gider, oradan başka bir araca yüklenir ve en sonunda arkadaşına ulaşır.
   - Yani veriler de internet üzerinden parçalar halinde farklı yollardan giderek yerine ulaşır.
3. **Paketi eksiksiz almak** 📦
   - Diyelim ki kutunun içindeki bazı şeyler eksik geldi. Arkadaşın kargo şirketine bildirip eksik olan parçaları tekrar göndermelerini isteyebilir.
   - İşte burada TCP (İletim Kontrol Protokolü) devreye girer. TCP, eksik olan verileri kontrol eder ve eğer eksik varsa tekrar gönderilmesini sağlar.

Sonuç olarak, TCP/IP sistemi internette veri göndermeyi, tıpkı bir kargo firmasının paketleri güvenli ve doğru bir şekilde teslim etmesine benzer bir şekilde gerçekleştirir.



Web’in Temelleri

HTTP Nedir?

HTTP (HyperText Transfer Protocol), yani Hiper Metin Transfer Protokolü, internet üzerindeki web sitelerinin bilgisayarlar ve sunucular arasında bilgi alışverişi yapmasını sağlayan bir protokoldür. Tarayıcıya bir web sitesi adresi yazdığında, HTTP devreye girer ve o sayfanın içeriğini (metinler, resimler, videolar vb.) sana ulaştırır.

✨Hayal Ederek Anlama

Diyelim ki çok acıktın ve bir restorana gidip yemek sipariş vermek istiyorsun. HTTP, işte bu sipariş verme ve yemeğin sana getirilme süreci gibidir.

1. **Menüyü Açarsın 📜 → Tarayıcıda Web Sitesine Girmek**
   - Restorana girdin ve bir menü aldın. İşte bu, internet tarayıcını açıp bir web sitesine girmeye benzer.
2. **Sipariş Verirsin 🍔 → HTTP İsteği Göndermek**
   - Garsona "Bir hamburger istiyorum" dersin. Bu, tarayıcının bir web sitesine "Bana bu sayfayı göster" demesi gibidir.
3. **Mutfakta Hazırlanır 👨‍🍳 → Sunucunun Yanıt Vermesi**
   - Garson siparişi mutfağa götürür. Mutfak, internet üzerindeki sunuculara benzer; web sitesinin bilgilerini hazırlar.
4. **Yemeğin Masana Gelir 🍽️ → Web Sitesi Yüklenir**
   - Garson yemeği getirir ve sen afiyetle yersin. İşte bu, tarayıcının web sitesini ekrana getirmesi gibidir.

Eğer internet olmasaydı, web sitelerine ulaşamazdık. İşte HTTP, interneti kullanarak bizim isteğimizi sunucuya ileten ve bize geri cevap getiren bir haberci gibidir!



Web’in Temelleri

HTTPS Nedir?

HTTPS (Hypertext Transfer Protocol Secure), internet üzerindeki verilerin güvenli bir şekilde iletilmesini sağlayan bir protokoldür. HTTP'nin güvenli bir versiyonudur ve web tarayıcıları ile sunucular arasındaki veri iletiminin şifrelenmesini sağlar. HTTPS, özellikle online işlemler (bankacılık, alışveriş vb.) yaparken kişisel bilgilerinizi korumak için kullanılır. HTTPS kullanan sitelerin adresi "https://" ile başlar.

✨Hayal Ederek Anlama

Düşün ki, internette bir gizli mesaj göndermek istiyorsun. Eğer bu mesajı açık bir şekilde gönderirsen, herkes görebilir. Ama gizli bir kutuya koyarsan, sadece gönderen ve alıcı görebilir. İşte HTTPS bu gizli kutuyu sağlar!

1. **Gizli Mesaj Gönderme** 🔒
   - İnternette bir şey yaparken, kişisel bilgilerin (adın, şifren, kredi kartı numaran) önemli olabilir. HTTPS, bu bilgileri kimsenin göremeyeceği şekilde gizler ve güvenli bir şekilde gönderir.
2. **Güvenli Yolculuk** 🚀
   - Düşün ki, bilmediğin birine mektup gönderiyorsun ve mektubunun yol boyunca açılmaması gerek. HTTPS, mektubunu güvenli bir kutuya koyar ve sadece doğru kişiye ulaştığında açılmasını sağlar.
3. **Güvenlik Kilidi** 🔐
   - HTTPS kullanan bir web sitesinin adresinin başında, genellikle bir kilit simgesi görürsün. Bu, o siteyi güvenle kullanabileceğin anlamına gelir.

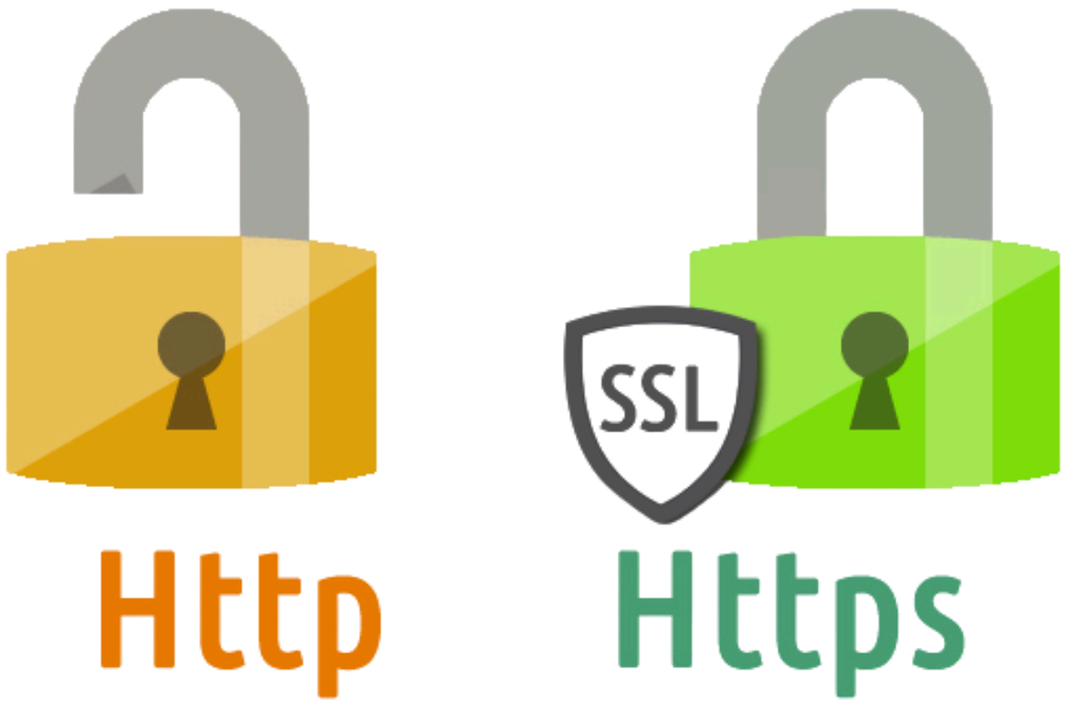
Özetle:

HTTPS, internette gizli ve güvenli iletişim sağlar. Bilgilerin, şifreli bir kutu içinde güvenle yol alır, böylece kötü niyetli kişiler onları göremez.

HTTP Metodları Nelerdir?

HTTP metodları, istemci (tarayıcı) ile sunucu arasındaki iletişimi yöneten komutlardır. En yaygın kullanılan HTTP metodları şunlardır:

- **GET** → Veriyi sunucudan almak için kullanılır. (Sayfa görüntüleme gibi)
- **POST** → Veriyi sunucuya göndermek için kullanılır. (Form gönderme gibi)
- **PUT** → Mevcut bir veriyi güncellemek için kullanılır.
- **DELETE** → Bir veriyi sunucudan silmek için kullanılır.
- **PATCH** → Verinin belirli bir kısmını güncellemek için kullanılır.



Web’in Temelleri

URL Nedir?

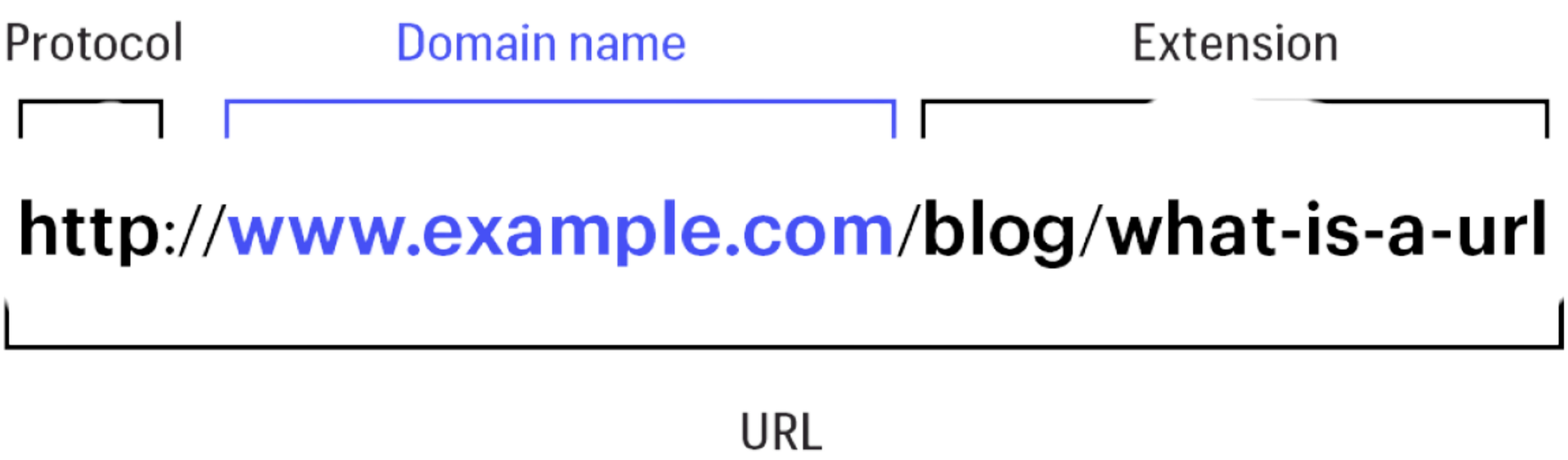
URL (Uniform Resource Locator), yani Tekdüzen Kaynak Bulucu, internet üzerindeki web sitelerinin adresidir. Bir web sitesine veya belirli bir sayfaya ulaşmak için kullanılan özel bir bağlantıdır. Örneğin, **https://www.google.com** bir URL’dir ve Google’ın web sitesine gitmemizi sağlar.

✨Hayal Ederek Anlama

Düşün ki büyük bir kütüphaneye gittin ve binlerce kitap var. Ama sen sadece belirli bir kitabı okumak istiyorsun. Peki, o kitabı nasıl bulursun? İşte URL, kütüphanedeki kitapların adresi gibi çalışır.

1. **Kütüphaneye Gitmek 📚 → İnternete Girmek**
   - Kütüphane, internetin kendisidir. İçinde milyonlarca bilgi var.
2. **Kitabın Adresini Bulmak 🏷️ → URL Kullanmak**
   - Kütüphanedeki her kitabın özel bir numarası (raf numarası) olur. Sen de bu numarayı bulup kitabı alırsın.
   - İşte URL de internet üzerindeki her web sitesine özel bir adres verir, böylece tarayıcı doğru siteyi açabilir.
3. **Kitabı Raftan Almak 📖 → Web Sitesine Gitmek**
   - Raf numarasını takip edip kitabı buldun ve okumaya başladın.
   - Aynı şekilde, tarayıcı da URL’yi takip eder ve seni doğru web sitesine götürür.

Yani URL, internetteki sitelerin ev adresi gibidir. Eğer URL’yi doğru yazmazsan, yanlış bir adrese gitmiş olursun!



Web’in Temelleri

DNS Nedir?

DNS (Domain Name System), yani Alan Adı Sistemi, internet üzerindeki web sitelerinin adlarını (örn. google.com) IP adreslerine çeviren bir sistemdir. Çünkü bilgisayarlar sayılarla (IP adresleriyle) çalışırken, insanlar kelimelerle (web site isimleriyle) çalışır. DNS, bu iki sistemi birbirine bağlayan bir çevirmen gibi çalışır.

✨Hayal Ederek Anlama

Diyelim ki telefon rehberin yok ve arkadaşlarını aramak istiyorsun. Ama onların telefon numaralarını ezberlemiyorsun, sadece isimlerini biliyorsun. İşte DNS, internetin telefon rehberi gibidir!

1. **Arkadaşının Adını Hatırlıyorsun 🔍 → Web Sitesi Adını Yazıyorsun**
   - Diyelim ki "Ahmet’i aramak istiyorum" diyorsun.
   - İnternette de "google.com" yazıyorsun.
2. **Telefon Rehberine Bakıyorsun 📖 → DNS Sunucusuna Sorgu Gönderiyorsun**
   - Telefon rehberine bakarak Ahmet’in numarasını buluyorsun.
   - DNS de "google.com" isminin hangi IP adresine ait olduğunu bulur.
3. **Numarayı Çevirip Arıyorsun ☎ → Web Sitesine Bağlanıyorsun**
   - Telefon numarasını tuşlayıp Ahmet’i arıyorsun.
   - DNS doğru IP adresini bulur ve tarayıcın o web sitesini açar.

Eğer DNS olmasaydı, web sitelerine gitmek için 142.250.186.14 gibi uzun ve karmaşık IP adreslerini ezberlememiz gerekirdi! İşte bu yüzden DNS, interneti bizim için çok daha kolay hale getirir.

Web’in Temelleri

Alan Adı Nedir?

Alan adı (domain name), bir internet sitesine erişim sağlamak için kullanılan adrestir. Bu, web sitesinin dijital dünyadaki ismi gibidir. Örneğin, google.com ya da example.org gibi adresler, internet üzerindeki farklı sitelere ulaşmak için kullanılır. Alan adları, daha önce belirli IP adresleriyle tanımlanan web sitelerine, insanların kolayca hatırlayabileceği isimler verir.

✨Hayal Ederek Anlama

Düşün ki, internette bir yere gitmek istiyorsun ama her yer sayısal bir adresle (örneğin, "192.168.0.1") tanımlanıyor. Bu çok zor olurdu, değil mi? İşte alan adı, bu karmaşık sayılar yerine, web sitesine gitmek için kullandığın kolay bir isim gibidir.

1. **Alan Adı = Web Sitesinin İsmi**
   - Alan adı, tıpkı ev adresi gibidir. Ev adresini yazmadan evine nasıl gidebilirsin? Aynı şekilde, web sitesi adresini yazmadan o siteye gidemezsin.
2. **Mesela, "google.com"**
   - Eğer internette "google.com" yazarsan, bu, Google web sitesine gitmeni sağlar. Bu, adeta bir evin adı gibi, seni doğru yere götürür.
3. **İnternetteki Telefon Rehberi**
   - Alan adı, telefon rehberine benzer. Sen bir isme bakıp o kişinin telefon numarasını buluyorsun. İnternette de, alan adı sayesinde web sitelerine ait IP adreslerini buluyorsun.

Özetle:

Alan adı, internetteki web sitelerinin ismi gibidir. Bir web sitesine gitmek için, tıpkı bir adrese gitmek gibi, o siteye ait alan adını yazarsın ve seni doğru yere götürür.



Web’in Temelleri

Sunucu Nedir?

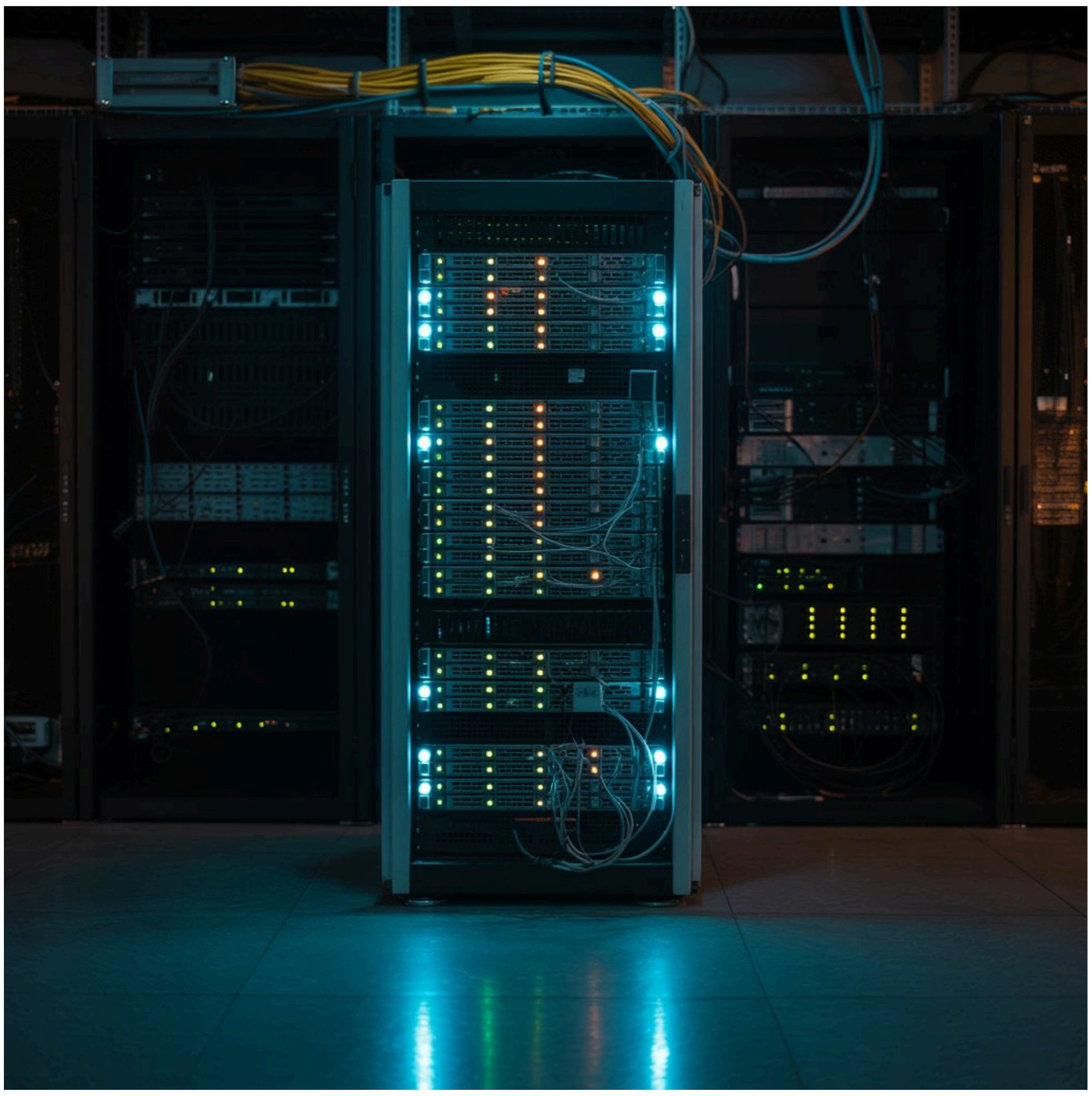
Sunucu (Server), bir ağ üzerinde çalışan ve diğer bilgisayarlara (istemcilere) veri, hizmet veya kaynak sağlayan güçlü bir bilgisayardır. Örneğin, bir web sitesine girdiğinde, o siteyi barındıran bir sunucu vardır ve bu sunucu sana web sayfasını gönderir.

✨Hayal Ederek Anlama

Düşün ki büyük bir restorana gittin ve yemek siparişi verdin. İşte sunucu, bu restorandaki aşçı gibidir! 👨‍🍳

1. **Sen Sipariş Verirsin 🍔 → Web Sitesine Girmek**
   - Garsona "Bir hamburger istiyorum" dersin.
   - Tarayıcına bir web sitesi yazdığında, aslında bir sunucudan o sayfayı istiyorsun.
2. **Aşçı Yemeği Hazırlar 👨‍🍳 → Sunucu Veriyi İşler**
   - Aşçı mutfakta siparişini hazırlar.
   - Sunucu, senin istediğin bilgiyi işler ve hazırlar.
3. **Garson Yemeği Masana Getirir 🍽️ → Web Sitesi Ekranına Gelir**
   - Garson yemeğini getirir ve sen yemeye başlarsın.
   - Sunucu, bilgiyi tarayıcına gönderir ve sen web sitesini görürsün.

Eğer sunucu olmazsa, internetteki hiçbir web sitesine ulaşamazsın. Çünkü tüm bilgiler bu güçlü bilgisayarlarda saklanır ve gerektiğinde kullanıcılara gönderilir!



Web’in Temelleri

İstemci Nedir?

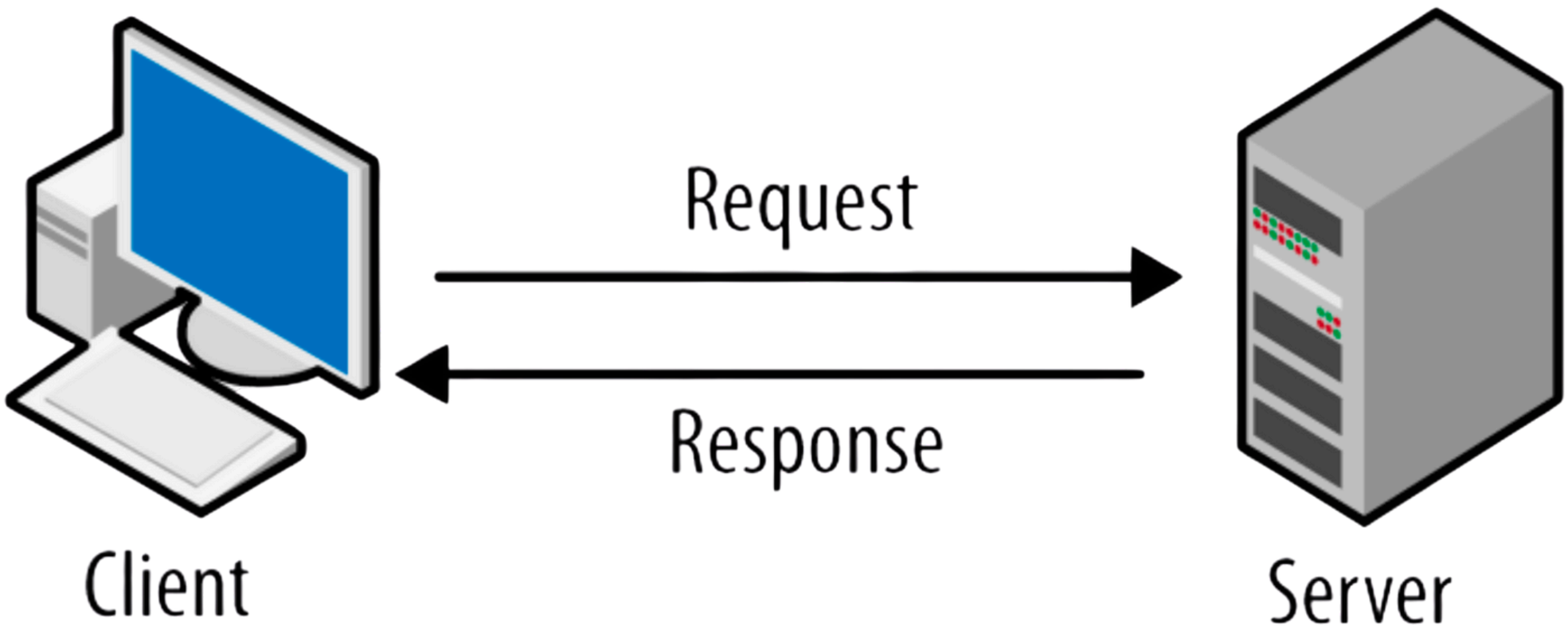
İstemci (Client), bir sunucudan hizmet veya bilgi talep eden cihaz veya yazılımdır. Bilgisayarın, telefonun veya tarayıcın bir istemci olarak çalışabilir. Örneğin, bir web sitesine girdiğinde tarayıcın (Google Chrome, Safari vb.) sunucudan verileri ister ve sana gösterir.

✨Hayal Ederek Anlama

Düşün ki bir restorana gittin ve yemek yemek istiyorsun. **İstemci,** restorandaki müşteri gibidir! 🍔

1. **Sen Yemek Sipariş Edersin 🍽️ → Web Sitesine Girmek**
   - Garsona "Bir pizza istiyorum" dersin.
   - Bilgisayarın da bir sunucuya "Bana bu web sitesini göster" der.
2. **Aşçı Yemeği Hazırlar 👨‍🍳 → Sunucu Bilgiyi Hazırlar**
   - Aşçı pizzanı yapar ve hazırlar.
   - Sunucu da senin istediğin web sitesinin bilgilerini hazırlar.
3. **Garson Yemeği Sana Getirir 🚶‍♂️ → Sunucu Veriyi Gönderir**
   - Garson pizzayı getirir ve sen afiyetle yersin.
   - Sunucu, bilgiyi istemciye yollar ve sen web sitesini görürsün.

Yani, istemci istek yapan, sunucu ise bu isteğe cevap veren taraftır. Eğer istemci olmazsa, kimse sunucudan bir şey istemezdi; eğer sunucu olmazsa, istemci istediklerini alamazdı!



Web’in Temelleri

Tarayıcı Nedir?

Tarayıcı (Web Browser), internet üzerindeki web sitelerini görüntülememizi sağlayan bir yazılımdır. Google Chrome, Mozilla Firefox, Safari ve Microsoft Edge gibi tarayıcılar, web sitelerinin kodlarını alıp bunları bizim anlayabileceğimiz şekilde gösterir.

✨Hayal Ederek Anlama

Düşün ki çok büyük bir kütüphaneye gittin ve kitap okumak istiyorsun. Ama kitaplar yabancı bir dilde yazılmış ve sen bu dili bilmiyorsun. İşte tarayıcı, bu kitapları senin anlayabileceğin dile çeviren bir rehber gibidir! 📖

1. **Kütüphaneye Gidersin 📚 → İnternete Bağlanırsın**
   - Kitaplara ulaşmak için kütüphaneye gidersin.
   - Web sitelerine ulaşmak için tarayıcıyı açarsın.
2. **Bir Kitap Seçersin 🟩 → Web Sitesine Girersin**
   - Kütüphanede okumak istediğin kitabı seçersin.
   - Tarayıcıya bir web sitesi adresi (URL) yazarsın.
3. **Rehber Sana Kitabı Anlatır 👩‍🏫 → Tarayıcı Web Sitesini Okunur Hale Getirir**
   - Rehber kitabın yabancı dildeki içeriğini senin anlayacağın dile çevirir.
   - Tarayıcı, web sitesinin kodlarını çözüp sana resimler, yazılar ve videolar olarak gösterir.

Eğer tarayıcı olmasaydı, web sitelerini sadece anlamsız kodlar olarak görürdük! İşte bu yüzden tarayıcılar, interneti kullanmamızı sağlayan sihirli bir araç gibidir.



Web Teknolojilerine Giriş

HTML Nedir?

HTML (Hypertext Markup Language), web sayfalarını oluşturmak için kullanılan bir dil ve yapıdır. Web sayfalarındaki metin, resimler, bağlantılar ve diğer içeriklerin nasıl görüneceğini tanımlamak için kullanılır. HTML, web sayfasının temel yapısını belirler ve tarayıcının bu sayfayı doğru bir şekilde görüntülemesini sağlar.

✨Hayal Ederek Anlama

Düşün ki bir ev inşa ediyorsun. Evin duvarlarını, pencerelerini, kapılarını ve odalarını nasıl yerleştireceğine karar veriyorsun. İşte HTML de, bir web sayfasının duvarlarını, pencerelerini ve odalarını yapmanı sağlayan bir plan gibidir!

1. **HTML = Evin Planı** 🏠
   - HTML, web sayfasının içindeki her şeyi yerli yerine koyman için bir plan gibidir. Mesela, bir başlık koymak istiyorsan, HTML ile "Bu başlık burada olsun" dersin.
2. **Web Sayfasındaki Öğeler** 🌄
   - HTML, bir sayfada yazıları, resimleri ve bağlantıları yerleştirmeni sağlar. Başlıkları, paragrafları ve resimleri nasıl göstereceğini belirtir. Yani, evin içinde hangi odaya ne koyacağın gibi!
3. **HTML Etiketleri**
   - HTML'de her şey bir etiket ile tanımlanır. Mesela, bir yazıyı başlık yapmak için <h1>Başlık</h1> gibi bir etiket kullanırsın. Bu, sanki odaların etiketlenmesi gibidir.

Özetle:

HTML, bir web sayfasının yapısını kurmanı sağlar. Tıpkı bir ev inşa etmek gibi, sayfadaki metinleri, resimleri ve bağlantıları doğru yerlere koymana yardımcı olur.



Web Teknolojilerine Giriş

XML Nedir?

XML (Extensible Markup Language), verileri saklamak ve taşımak için kullanılan bir işaretleme dilidir. HTML'e benzer şekilde etiketler kullanarak verileri düzenler, ancak HTML'den farklı olarak, XML sadece verilerin yapısını tanımlar, nasıl gösterileceğine dair bir şey belirtmez. **XML, veri alışverişi yapan** sistemler arasında bilgi paylaşımını kolaylaştırır.

✨Hayal Ederek Anlama

Düşün ki, kendi küçük kütüphanen var ve her bir kitabı bir etiketle işaretlemek istiyorsun. Her kitabın bir adı, yazarı, yayın tarihi ve sayfa sayısı var. XML, kitapların bu özelliklerini etiketlerle düzenlemene yardımcı olur!

1. **XML = Etiketli Kitaplar** 📚
   - Her kitabın özelliklerini (adını, yazarını) etiketlerle yazarsın. Örneğin, "Bu kitap Alice'in Maceraları", "Yazarı Lewis Carroll" gibi. XML, işte böyle verileri etiketlerle düzenler.
2. **Veri Taşımak** 📤
   - Şimdi, bu kitapları başka bir kütüphaneye göndereceksin. XML, kitapları düzenli bir şekilde taşımana yardımcı olur. Yani veriyi başka bir yere gönderirken, hangi kitabın ne olduğunu etiketli bir şekilde iletebilirsin.
3. **XML Etiketleri**
   - XML’de her şey bir etiketle belirtilir. Mesela, bir kitabın adı şöyle yazılabilir: <kitap><ad>Alice'in Maceraları</ad><yazar>Lewis Carroll</yazar></kitap>. Yani veriyi anlaşılır bir şekilde düzenler.

Özetle:

XML, bilgiyi etiketlerle düzenlemeni sağlar. Tıpkı kitapların özelliklerini etiketlemek gibi, verileri düzenler ve farklı yerlerde kullanılmasını kolaylaştırır.



Web Teknolojilerine Giriş

CSS Nedir?

CSS (Cascading Style Sheets), web sayfalarının görünümünü ve düzenini belirlemek için kullanılan bir dil ve teknolojidir. HTML, sayfanın yapısını ve içeriğini belirlerken, CSS bu içeriğin nasıl görüneceğini tanımlar. Yazı tiplerini, renkleri, arka planları, hizalamayı ve sayfa düzenini belirlemek için CSS kullanılır.

✨Hayal Ederek Anlama

Düşün ki, bir resim yapıyorsun ama sadece siyah beyaz bir çizim var. CSS, o çizime renkler, desenler, gölgeler ve şekiller ekleyerek ona canlılık ve güzellik katan bir boyama kitabı gibidir!

1. **HTML = Çizim** 🖼️
   - HTML, web sayfanın çizimi gibidir. İçinde başlıklar, yazılar, resimler gibi şeyler bulunur. Ama bu çizimler sadece siyah beyazdır.
2. **CSS = Boyama Kitabı** 🎨
   - CSS, bu çizimlere renkleri, şekilleri, fontları ve düzeni ekler. Yani, sayfanın daha güzel, renkli ve düzenli görünmesini sağlar.
3. **CSS ile Stil Verme** ✨
   - Mesela, bir başlığı daha büyük ve kalın yapmak için CSS kullanırsın. Ya da yazının rengini kırmızı yapmak için bir talimat verirsin. "Bu başlık kırmızı olsun" veya "Yazılar mavi renkte olsun" gibi.

Özetle:

CSS, HTML'in çizimlerini süsler. Web sayfanın ne kadar güzel ve düzenli görüneceğini CSS belirler. O yüzden bir web sayfasının görüntüsünü oluşturur ve renklendirir.



Web Teknolojilerine Giriş

JavaScript Nedir?

JavaScript, web sayfalarına etkileşim ve dinamik özellikler eklemek için kullanılan bir programlama dilidir. HTML ve CSS sayfanın yapısını ve görünümünü belirlerken, JavaScript sayfanın kullanıcıyla nasıl etkileşimde bulunacağını kontrol eder. Örneğin, bir butona tıklandığında bir şeyin değişmesini sağlamak veya form bilgilerini kontrol etmek gibi işlemler JavaScript ile yapılır.

✨Hayal Ederek Anlama

Düşün ki, büyük bir oyuncak araba yapıyorsun. Bu araba harika görünüyor (CSS sayesinde) ve doğru şekilde yerleştirilmiş (HTML sayesinde). Ama bu arabanın hareket etmesini sağlamak istersin. İşte JavaScript, arabanın hareket etmesini sağlayan motor gibidir!

1. **HTML ve CSS = Araba** 🚗
   - HTML, arabayı inşa etmek için kullandığın parçalar gibidir. CSS, arabanın görünümünü güzelleştirir. Ama araba duruyor, hareketsiz.
2. **JavaScript = Motor** 🏎️
   - JavaScript, bu arabanın hareket etmesini sağlar! Bir düğmeye tıkladığında, araba hızlanabilir veya bir şey değişebilir. Yani, JavaScript sayfana hayat verir ve ona etkileşim kazandırır.
3. **Etkileşim ve Değişim** ⚙️
   - Mesela, bir butona tıklayınca bir şeylerin değişmesini istiyorsan, JavaScript ile bunu sağlarsın. "Butona tıklayınca bir mesaj görünsün" gibi. JavaScript, tıpkı bir büyücü gibi, sayfanın istediğin şekilde hareket etmesini sağlar.

Özetle:

JavaScript, web sayfalarını interaktif hale getiren bir dil gibidir. HTML ve CSS sayfayı kurar ve güzel gösterir, ama JavaScript sayfanın hareket etmesini, değişmesini sağlar.



Web Teknolojilerine Giriş

DOM Nedir?

DOM (Document Object Model), HTML veya XML belgelerinin tarayıcı tarafından temsil edilen yapısal modelidir. DOM, web sayfasındaki her öğeyi (başlıklar, paragraflar, resimler, bağlantılar vb.) bir nesne olarak kabul eder ve JavaScript ile bu nesneleri değiştirmemize, eklememize veya silmemize olanak tanır. DOM sayesinde web sayfaları dinamik ve etkileşimli hale gelir.

✨Hayal Ederek Anlama

Düşün ki, bir evin içindesin ve odaların her birini istediğin gibi düzenlemek istiyorsun. DOM, evin her odasını ve eşyalarını bir liste gibi düşünür. Eğer bir oda değiştirmek istersen, bu listeyi görüp istediğin gibi değiştirebilirsin!

1. **Web Sayfası = Ev** 🏠
   - Web sayfası, tıpkı bir evin içi gibidir. İçinde başlıklar, paragraflar, resimler gibi farklı odalar vardır.
2. **DOM = Oda Listesi** 📋
   - DOM, evdeki her odayı bir liste olarak gösterir. Mesela, "bu oda başlık odası", "bu oda yazı odası" gibi. DOM sayesinde her odayı ve içindeki eşyaları görüp değiştirebilirsin.
3. **JavaScript ile Oyun** 🎮
   - DOM sayesinde, mesela bir odadaki koltuğu değiştirmek veya resimleri kaldırmak gibi işlemleri JavaScript ile yapabilirsin. Yani, DOM bir oyun gibi, web sayfasındaki her şeyi istediğin gibi düzenleyebilmeni sağlar.

Özetle:

DOM, web sayfasındaki her şeyi bir nesne gibi düşünmeni sağlar. JavaScript ile bu nesneleri görüp değiştirebilir, ekleyebilir veya silebilirsin. Tıpkı evindeki odaları ve eşyaları düzenlemek gibi!

Web Teknolojilerine Giriş

DOM Manipülasyonu Nedir?

DOM Manipülasyonu, JavaScript kullanarak bir web sayfasındaki öğeleri (başlıklar, metinler, resimler, butonlar vb.) değiştirme, ekleme veya silme işlemidir. Bu sayede web sayfaları dinamik hale gelir ve kullanıcı etkileşimi ile sayfa içeriği anında güncellenebilir. **DOM Manipülasyonu,** web sayfasındaki içeriklerin kullanıcı ile etkileşime girerek değişmesini sağlar.

✨Hayal Ederek Anlama

Düşün ki, bir oyuncak evin var. Bu evde bazı odaları değiştirmek, yeni eşyalar eklemek veya eski eşyaları kaldırmak istiyorsun. DOM Manipülasyonu, işte bu evdeki odaları ve eşyaları istediğin gibi değiştirmene olanak tanır!

1. **Web Sayfası = Oyuncak Ev**
   - Web sayfası, tıpkı bir oyuncak evin içi gibidir. Odalarda başlıklar, metinler, butonlar ve resimler gibi eşyalar vardır.
2. **DOM Manipülasyonu = Oda Düzenlemesi**
   - DOM Manipülasyonu, bu eşyaları değiştirmene, eklemene veya kaldırmana yardımcı olur. Mesela bir odaya yeni bir oyuncak ekleyebilir veya eski bir oyuncak kaldırabilirsin.
3. **JavaScript ile Değişiklik**
   - JavaScript sayesinde, örneğin bir butona tıklayınca odadaki eşyaların değişmesini sağlayabilirsin. Yani DOM Manipülasyonu, tıpkı butona basarak odanı istediğin gibi değiştirmek gibidir.

Özetle:

DOM Manipülasyonu, web sayfasındaki içerikleri değiştirmek, eklemek veya silmek demektir. Tıpkı bir oyuncak evi düzenlemek gibi, sayfanın her öğesini JavaScript ile değiştirebilir ve şekillendirebilirsin!

Web Teknolojilerine Giriş

AJAX Nedir?

AJAX (Asynchronous JavaScript and XML), web sayfalarında veri almak veya göndermek için kullanılan bir tekniktir. AJAX, sayfa yeniden yüklenmeden, yani kullanıcı sayfayı yenilemeden veri almanızı ve göndermenizi sağlar. Bu, web sayfalarının daha hızlı ve daha dinamik hale gelmesini sağlar. Genellikle, dinamik içerik güncellemeleri ve form verisi gönderimleri için kullanılır.

✨Hayal Ederek Anlama

Düşün ki bir yemek siparişi veriyorsun, ama her sipariş vermek istediğinde bütün restoranı tekrar arayıp siparişini yenilemen gerektiğini hayal et. Bu, çok zaman kaybettirir! Ama AJAX, her seferinde restoranı tekrar aramadan sadece yemek siparişi vermeni sağlar.

1. **Sayfayı Yenilemek = Restoranı Yeniden Aramak** 📞
   - Eğer bir şey değiştirmek için sayfayı yeniden yüklesen, bu tıpkı bir yemeği tekrar sipariş vermek gibidir. Zaman kaybı olur ve beklemek zorlaşır.
2. **AJAX = Hızlı Sipariş Verme** 🍕
   - AJAX, sayfayı yeniden yüklemene gerek kalmadan sadece yemek siparişi vermene imkan tanır. Yani, her seferinde bir şey değiştirmek istediğinde, sayfa yenilenmeden bu değişikliği yapabilirsin. Yavaş beklemelere gerek yok!
3. **Arka Planda Çalışma** 🔄
   - AJAX, tıpkı mutfakta çalışan bir garson gibi arka planda veri alır veya gönderir. Sen sayfayı kullanmaya devam edersin, garson ise siparişi hazırlamak için mutfakta çalışır.

Özetle:

AJAX, sayfayı yeniden yüklemeden arka planda veri almanı ve göndermeni sağlar. Tıpkı bir yemek siparişi verirken restoranı tekrar aramadan işlemi yapabilmen gibi, internetin hızlı ve kesintisiz olmasını sağlar.

Frontend Geliştirme Teknikleri

CSS Framework’leri ve Utility-First Yaklaşımı Nedir?

- **CSS Framework’leri:** Önceden hazırlanmış stil dosyaları ve bileşenlerden oluşan, web sitelerinin tasarımını kolaylaştıran kütüphanelerdir. Örneğin, Bootstrap ve Tailwind CSS gibi popüler framework’ler, geliştiricilerin hızlı ve tutarlı bir şekilde web siteleri oluşturmasını sağlar.
- **Utility-First Yaklaşımı:** CSS yazımında, belirli bir tasarım öğesi için özel sınıflar yerine, küçük ve yeniden kullanılabilir "yardımcı sınıflar" (utility classes) kullanmayı esas alan bir yaklaşımdır. Örneğin, "text-red-500" veya "p-4" gibi sınıflar doğrudan bir öğenin stilini belirler.

✨Hayal Ederek Anlama

CSS Framework’lerini bir lego setine benzetebiliriz. Eğer bir evi sıfırdan inşa etmek yerine hazır lego parçaları kullanırsan, hem daha hızlı hem de düzgün bir sonuç elde edersin. İşte CSS Framework’leri de tam olarak bunu yapıyor! Web siteni tasarlarken sana hazır butonlar, menüler ve kutular veriyor.

Utility-First yaklaşımı ise lego parçalarını tek tek kullanarak kendi evini özelleştirmen gibidir. Yani, “Ben hazır bir ev istemiyorum, çatısını kırmızı, pencerelerini büyük yapacağım” diyerek her parçayı ayrı ayrı seçebilirsin. Tailwind CSS gibi utility-first framework’ler, sana her detayı kontrol edebileceğin küçük parçalar sunar.

Bu sayede, ya hazır bir evi alıp hemen içine taşınabilirsin (CSS Framework'leri) ya da tuğlaları tek tek dizerek tam istediğin gibi bir ev inşa edebilirsin (Utility-First Yaklaşımı).



Frontend Geliştirme Teknikleri

Frontend Frameworkleri Nedir?

Frontend Frameworkleri, web sitelerinin kullanıcı arayüzünü (görünen kısmını) oluşturmayı kolaylaştıran, içinde hazır kodlar ve bileşenler barındıran kütüphanelerdir. React, Vue.js ve Angular gibi popüler framework'ler, geliştiricilere modüler ve yeniden kullanılabilir yapılar sunarak web uygulamalarını daha verimli bir şekilde geliştirmelerine yardımcı olur.

Bu framework'ler genellikle bileşen tabanlıdır (component-based), yani web sayfasını küçük, bağımsız parçalar halinde oluşturmanıza olanak tanır.

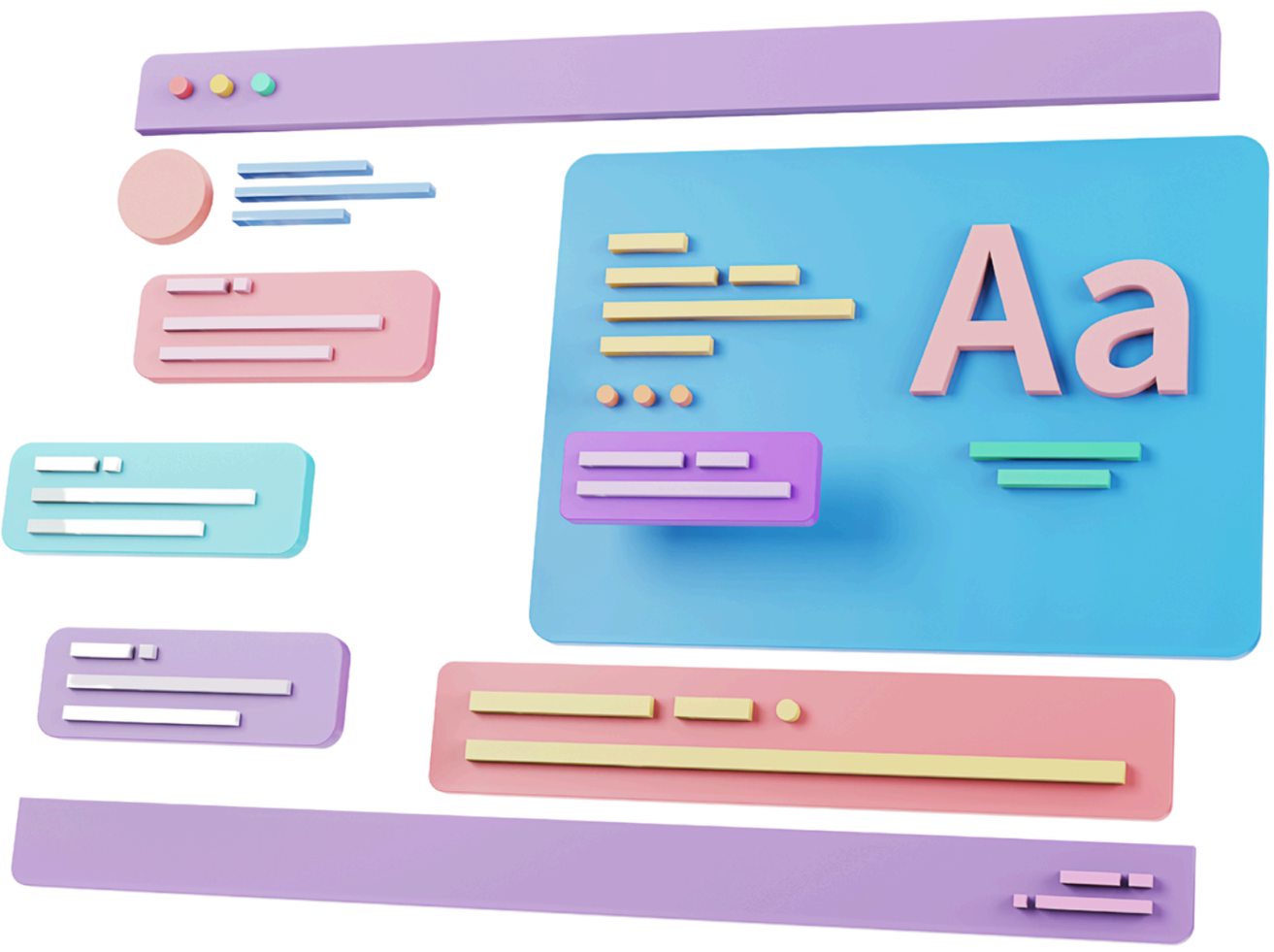
✨Hayal Ederek Anlama

Frontend framework'lerini lego setindeki talimat kitapçığına benzetebiliriz.
Diyelim ki büyük bir şato inşa etmek istiyorsun. Eğer her parçayı sıfırdan oluşturmak zorunda kalsaydın, çok uzun sürebilirdi. Ama Frontend Frameworkleri, sana önceden hazırlanmış parçalar ve nasıl birleştireceğini gösteren bir rehber sunar.

💡 Örneğin:

- React kullanıyorsan, "buton", "kart", "menü" gibi parçaları kolayca ekleyebilirsin.
- Vue.js veya Angular gibi framework'ler ise sana bu parçaları yönetmek ve birbirleriyle nasıl etkileşim kuracağını göstermek için kurallar sağlar.

Sonuç olarak, Frontend Framework'leri sayesinde, web siteleri ve uygulamalar daha hızlı, daha düzenli ve daha kolay yönetilebilir hale gelir.



Frontend Geliştirme Teknikleri

MVVM & Virtual DOM Nedir?

1. MVVM (Model-View-ViewModel) Nedir?

MVVM, modern frontend geliştirmede kullanılan bir tasarım desenidir (design pattern). Uygulamayı üç temel bölüme ayırarak yönetilebilir ve modüler hale getirir:

- **Model (M):** Uygulamanın veri ve iş mantığını temsil eder.
- **View (V):** Kullanıcının gördüğü ve etkileşimde bulunduğu arayüzdür.
- **ViewModel (VM):** Model ve View arasında köprü görevi gören katmandır. Kullanıcıdan gelen girdileri işler ve arayüzü günceller.

Vue.js ve Angular gibi framework'ler bu yapıyı kullanır.

2. Virtual DOM (Sanal DOM) Nedir?

Virtual DOM, gerçek DOM'un (HTML ağacının) daha hızlı güncellenmesini sağlayan bir ara bellek mekanizmasıdır.

- Gerçek DOM'u doğrudan değiştirmek yavaştır çünkü tüm sayfa yeniden çizilir.
- Virtual DOM, sayfanın hafif bir kopyasını oluşturur ve sadece değişen kısımları günceller, böylece performansı artırır.
- React, Virtual DOM kullanarak hızlı ve verimli UI güncellemeleri yapar.

✨Hayal Ederek Anlama

MVVM'yi Açıklayalım

Diyelim ki bir robot garson var ve bir restoranın siparişlerini yönetiyor:

- **Model:** Siparişlerin saklandığı mutfak.
- **View:** Müşterilerin gördüğü menü ve masaya gelen yemekler.
- **ViewModel:** Garson (ViewModel, mutfak ile müşteri arasında iletişim kurarak siparişleri alır ve getirir).

Garson (ViewModel), mutfaktaki siparişleri alır ve müşteriye sunar. Aynı şekilde, müşteri yeni bir sipariş verdiğinde bunu mutfağa bildirir.

Virtual DOM'u Açıklayalım

Gerçek DOM'u güncellemek, bir defterde tüm sayfayı silip baştan yazmak gibidir. Bu çok zaman alır.

Virtual DOM ise önce bir taslak deftere değişiklikleri yazar, sonra sadece gerekli yerleri temizleyip günceller. Böylece her seferinde tüm sayfayı yeniden yazmak yerine, sadece değişen kısımlar düzenlenir.

Bu yüzden, Virtual DOM kullanan React gibi framework'ler çok daha hızlı çalışır!

Frontend Geliştirme Teknikleri

Single Page Application (SPA) Nedir?

Single Page Application (SPA), yani Tek Sayfa Uygulaması, kullanıcıların sayfa yenilemesine gerek kalmadan farklı içeriklere erişmesini sağlayan bir web geliştirme yaklaşımıdır.

Geleneksel web sitelerinde, bir sayfadan başka bir sayfaya geçerken tam sayfa yenilenir. SPA'de ise, sayfa yenilenmeden sadece gerekli veriler güncellenir. Bu, JavaScript'in AJAX ve Virtual DOM gibi teknolojileriyle sağlanır.
Vue.js, React ve Angular gibi framework'ler SPA geliştirmek için sıkça kullanılır.

✨Hayal Ederek Anlama

Diyelim ki bir restoranda yemek siparişi veriyorsun:

- Geleneksel web siteleri bir garsonun her siparişten sonra menüyü geri götürüp yeni menüyü getirmesi gibidir. Her seferinde tüm süreç yeniden başlar (sayfa yenilenir).
- SPA ise garsonun masanda durması ve sadece sipariş ettiğin yemeği getirmesi gibidir. Menü hep aynı kalır, sadece içeriği değişir (sayfa yenilenmez, sadece gereken kısımlar güncellenir).

Bu yüzden SPA'ler daha hızlı ve akıcı bir kullanıcı deneyimi sunar!

Frontend Geliştirme Teknikleri

Router & State Management Nedir?

1. Router (Yönlendirme Sistemi) Nedir?

Router, web uygulamalarında kullanıcıyı farklı sayfalara veya bileşenlere yönlendiren bir sistemdir.

- Geleneksel web sitelerinde her sayfa değişiminde sunucudan yeni bir HTML sayfası alınır.
- SPA'lerde (Tek Sayfa Uygulamaları) Router sayesinde sayfa yenilenmeden URL değişir ve yalnızca gerekli bileşenler güncellenir.

Örneğin, React'te React Router, Vue'da Vue Router gibi kütüphaneler bu iş için kullanılır.

2. State Management (Durum Yönetimi) Nedir?

State Management, bir uygulamanın verilerini yönetme yöntemidir.

- Web uygulamalarında birçok bileşen aynı verilere ihtiyaç duyabilir.
- State Management, bu verilerin merkezi bir yerde saklanmasını ve yönetilmesini sağlar.

Redux (React için), Vuex (Vue için) ve Pinia gibi kütüphaneler, bu süreci kolaylaştırır.

✨Hayal Ederek Anlama

Router'ı Açıklayalım

Diyelim ki büyük bir alışveriş merkezindesin ve mağazalar arasında gezmek istiyorsun.

- Geleneksel sitelerde, her mağazaya girdiğinde yeniden dışarı çıkıp kapıdan girmen gerekir (sayfa yenilenir).
- Router kullanan bir SPA'de ise alışveriş merkezinin içindeki asansörü kullanarak mağazalar arasında hızlıca gezebilirsin (sayfa yenilenmez, içerik değişir).

State Management'ı Açıklayalım

Bir okul kütüphanesinde olduğumuzu düşünelim.

- Öğrenciler kitap ödünç almak için sıraya girmesin diye kütüphaneci (State Management) herkesin hangi kitabı aldığını takip eder.
- Eğer bir öğrenci başka bir kitap isterse, kütüphaneci kitap listesini günceller ve diğer öğrencilere de bildirir.

State Management sayesinde, her bileşen (öğrenci) ihtiyacı olan veriye (kitap) merkezi bir yerden ulaşabilir!

Frontend Geliştirme Teknikleri

MVP, MVC ve MVVM Tasarım Desenleri Nedir?

Tasarım desenleri (Design Patterns), yazılım projelerinde kodun daha düzenli, modüler ve yönetilebilir olmasını sağlayan mimari yaklaşımlardır. MVP, MVC ve MVVM gibi desenler, özellikle frontend ve backend geliştirme süreçlerinde kullanılır.

1. MVC (Model-View-Controller) Nedir?

MVC, yazılımı üç ana katmana ayıran bir modeldir:

- **Model (M):** Uygulamanın verilerini ve iş mantığını yönetir.
- **View (V):** Kullanıcının gördüğü arayüzdür.
- **Controller (C):** Kullanıcıdan gelen girişleri alır ve uygun Model ve View'leri günceller.

Nerede Kullanılır?

PHP, ASP.NET, Django, Ruby on Rails gibi framework'lerde yaygın olarak kullanılır.

2. MVP (Model-View-Presenter) Nedir?

MVP, MVC'ye benzer ama Controller yerine Presenter kullanır:

- **Model (M):** Verileri ve iş mantığını içerir.
- **View (V):** Kullanıcıya gösterilen arayüzdür.
- **Presenter (P):** View ile Model arasında köprü görevi görür.

Farkı Ne?

- **Controller (MVC'de),** kullanıcıdan gelen girdileri işler ve Model ile View'yi bağlar.
- **Presenter (MVP'de),** daha bağımsız çalışır ve View'nin yükünü azaltır.

Nerede Kullanılır?

Android uygulamalarında, özellikle eski sürümlerde popülerdi.

3. MVVM (Model-View-ViewModel) Nedir?

MVVM, özellikle modern frontend framework'lerinde kullanılan bir tasarım desenidir:

- **Model (M):** Verileri ve iş mantığını yönetir.
- **View (V):** Kullanıcının gördüğü arayüzdür.
- **ViewModel (VM):** Model ile View arasında veri bağlantısını (data binding) sağlar.

Nerede Kullanılır?

Vue.js, Angular ve WPF gibi teknolojilerde yaygın olarak kullanılır.

Frontend Geliştirme Teknikleri

✨Hayal Ederek Anlama

Diyelim ki bir restoranda garson olarak çalışıyorsun.

- **MVC:**
  - **Mutfak (Model):** Yemekleri hazırlar.
  - **Garson (Controller):** Müşteriden siparişi alır ve mutfağa iletir. Yemek hazır olduğunda müşteriye getirir.
  - **Müşteri (View):** Menüyü görür ve sipariş verir.
- **MVP:**
  - **Mutfak (Model):** Yemekleri hazırlar.
  - **Başgarson (Presenter):** Siparişleri alır ve mutfağa iletir. Hazırlanan yemekleri müşteriye sunar.
  - **Garson (View):** Sadece yemekleri masaya koyar, mutfakla doğrudan iletişim kurmaz.
- **MVVM:**
  - **Mutfak (Model):** Yemekleri hazırlar.
  - **Akıllı Sipariş Sistemi (ViewModel):** Siparişleri otomatik olarak mutfağa iletir ve yemek hazır olunca garsona haber verir.
  - **Garson (View):** Siparişin durumunu takip eder ve müşteriye yemekleri getirir.

Özetle:

- MVC’de garson her işi manuel yapar.
- MVP’de başgarson (Presenter) işleri yönetir, garson sadece servis yapar.
- MVVM’de akıllı bir sistem (ViewModel) işleri otomatize eder ve her şey daha kolay olur!

Backend Geliştirme Temelleri

PHP Nedir?

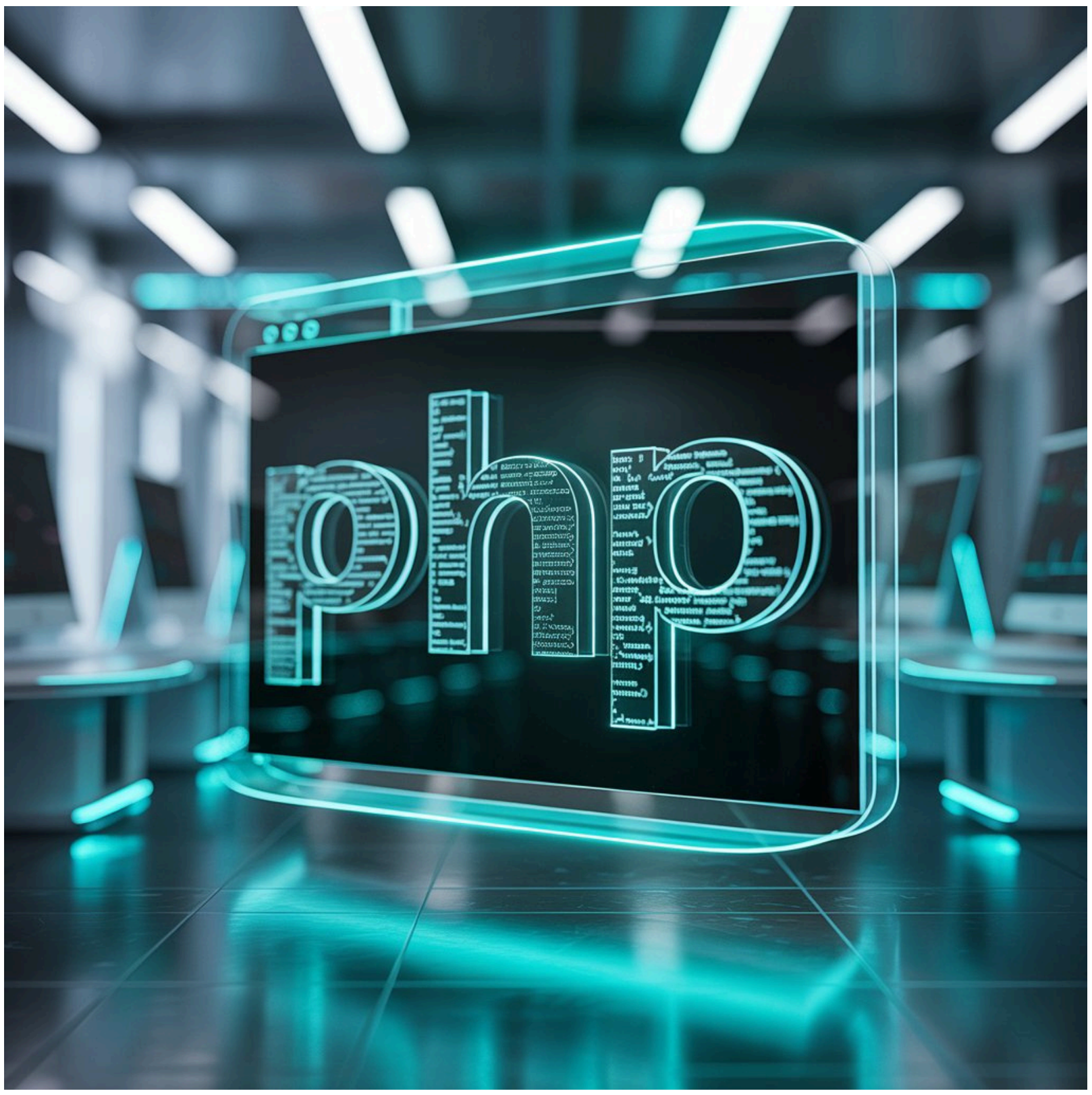
PHP (Hypertext Preprocessor), özellikle web siteleri ve web uygulamaları geliştirmek için kullanılan bir sunucu taraflı programlama dilidir. Dinamik içerikler oluşturmak için kullanılır, yani PHP ile yazılmış bir web sitesi, kullanıcıların taleplerine göre farklı sonuçlar gösterebilir. PHP, HTML ile birlikte çalışarak web sayfalarındaki verileri işler, veritabanlarından bilgi alır ve kullanıcı etkileşimlerini yönetir. PHP, genellikle web sunucularında çalıştırılır ve dosya işlemleri, veritabanı bağlantıları gibi işlemleri sunucu tarafında yapar.

✨Hayal Ederek Anlama

PHP, tıpkı bir bilgisayarın gizli çalışan beyin gibidir! Web sitesini kullanan insanlar, mesela bir butona tıklıyor ya da bir şey yazıyor, işte PHP hemen bu istekleri alır ve yerine getirir.

Düşün ki bir restoran var. Müşteri yemek siparişi veriyor. PHP, garson gibi bu siparişi alır, mutfaktaki aşçılara (sunucuya) gönderir. Mutfakta aşçılar (sunucu tarafındaki PHP kodları) siparişi hazırlayıp garsona (kullanıcıya) teslim eder.

PHP her zaman web sitelerinde çalışan gizli bir yardımcıdır. Örneğin, "Hava durumu nedir?" diye sormak istersen, PHP bunu alır, internetten doğru bilgiyi bulur ve sana gösterir. Yani PHP, web sitesinin her zaman düzgün çalışmasını sağlar.



Backend Geliştirme Temelleri

Apache/Nginx Nedir?

Apache ve Nginx, web sunucusu yazılımlarıdır. Web sunucusu, internetteki web sitelerinin çalışmasını sağlayan yazılımlardır. Apache ve Nginx, web sitelerinin dosyalarını internet kullanıcılarına sunan yazılımlar olup, her ikisi de farklı özellikler ve avantajlara sahiptir.

- **Apache,** uzun yıllardır kullanılan ve en yaygın web sunucularından biridir. Esnek ve özelleştirilebilir özellikleri ile bilinir.
- **Nginx,** daha yeni bir web sunucusu olup, yüksek performansı ve düşük sistem kaynakları kullanımıyla tercih edilir. Özellikle yüksek trafikli sitelerde popülerdir.

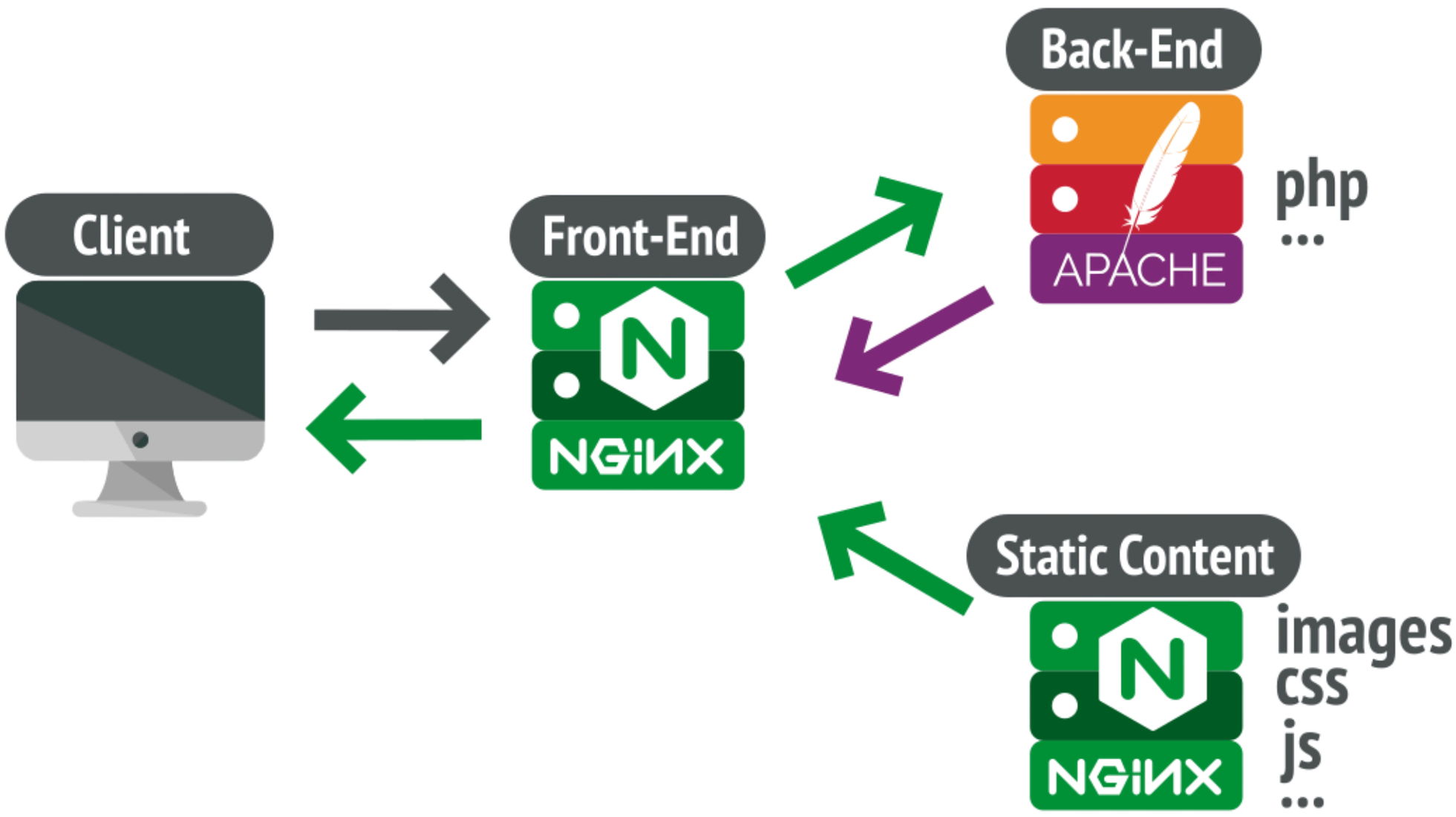
Her iki sunucu da HTTP protokolünü kullanarak, kullanıcılara web sayfalarını ve içerikleri sunar.

✨Hayal Ederek Anlama

Düşün ki bir restoran açıyorsun ve müşteriler gelerek yemek siparişi veriyor. Ama bu restoran, web sitesi gibi çalışıyor. İşte Apache ve Nginx, restoranın garsonları gibi düşün! 🍽️

- Apache biraz daha eski ama çok deneyimli ve her tür siparişi alabilen bir garson. O yüzden farklı müşterilere, farklı yemekler çıkarabilir.
- Nginx ise biraz daha hızlı ve pratik. Daha çok birden fazla müşteriye hızlıca yemek servisi yapabilen bir garson. O yüzden daha yoğun restoranlarda daha fazla tercih edilir.

Yani, Apache ve Nginx, web sitelerinin düzgün çalışması için garsonluk yapan yazılımlar gibidir. Her ikisi de siparişleri alır, yemekleri (web sayfalarını) getirir ve web sitenizi çalışan hale getirir!



Backend Geliştirme Temelleri

Node.js Nedir?

Node.js, JavaScript ile sunucu tarafında kod yazmayı sağlayan bir çalışma ortamıdır (runtime environment). Eskiden JavaScript sadece web tarayıcılarında çalışıyordu. Node.js sayesinde artık JavaScript'i sunucu tarafında da kullanabiliyoruz.

Node.js'in özellikleri:

- **Asenkron ve Olay Tabanlıdır:** Aynı anda birçok isteği hızlı bir şekilde işleyebilir.
- **Tek İş Parçacıklı (Single Threaded) ve Non-Blocking'dir:** Sunucunun aynı anda birçok kullanıcıya yanıt vermesini sağlar.
- **V8 Motoru ile Çalışır:** Google Chrome'un JavaScript motorunu kullanır, bu yüzden çok hızlıdır.
- **npm (Node Package Manager) ile Binlerce Kütüphane Kullanılabilir:** Geliştiricilere büyük kolaylık sağlar.

Nerede Kullanılır?

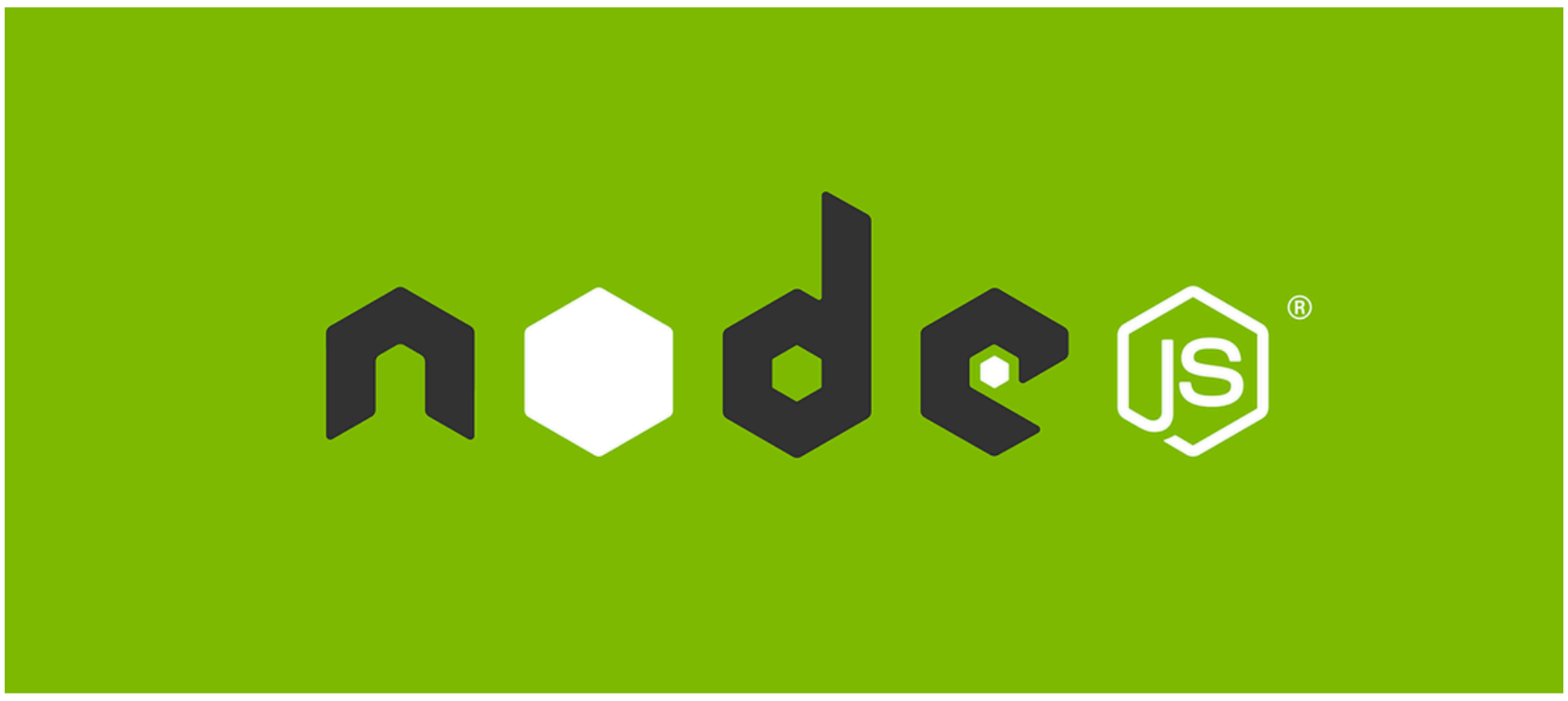
- Web sunucuları (Express.js ile)
- Gerçek zamanlı uygulamalar (Chat, oyunlar)
- API geliştirme
- Mikro servis mimarileri

✨Hayal Ederek Anlama

Diyelim ki hızlı çalışan bir pizzacı açtın. Gelen siparişleri nasıl yönetirsin?

- **Geleneksel sistem (PHP gibi senkron çalışan diller):**
  - Bir müşteri sipariş verdiğinde, önce o sipariş hazırlanır. Sonra sıradaki müşteriyle ilgilenilir.
  - Eğer biri büyük bir pizza isterse, diğer müşteriler beklemek zorunda kalır.
- **Node.js sistemi:**
  - Tüm siparişleri aynı anda alabilen süper hızlı bir garson gibi çalışır.
  - Büyük pizzayı fırına atarken diğer siparişlerle ilgilenebilir.
  - Kimse beklemez, çünkü aynı anda birçok işi yapabilir.

Sonuç: Node.js sayesinde web siteleri ve uygulamalar daha hızlı, verimli ve gerçek zamanlı çalışabilir



Backend Geliştirme Temelleri

CRUD Operasyonları

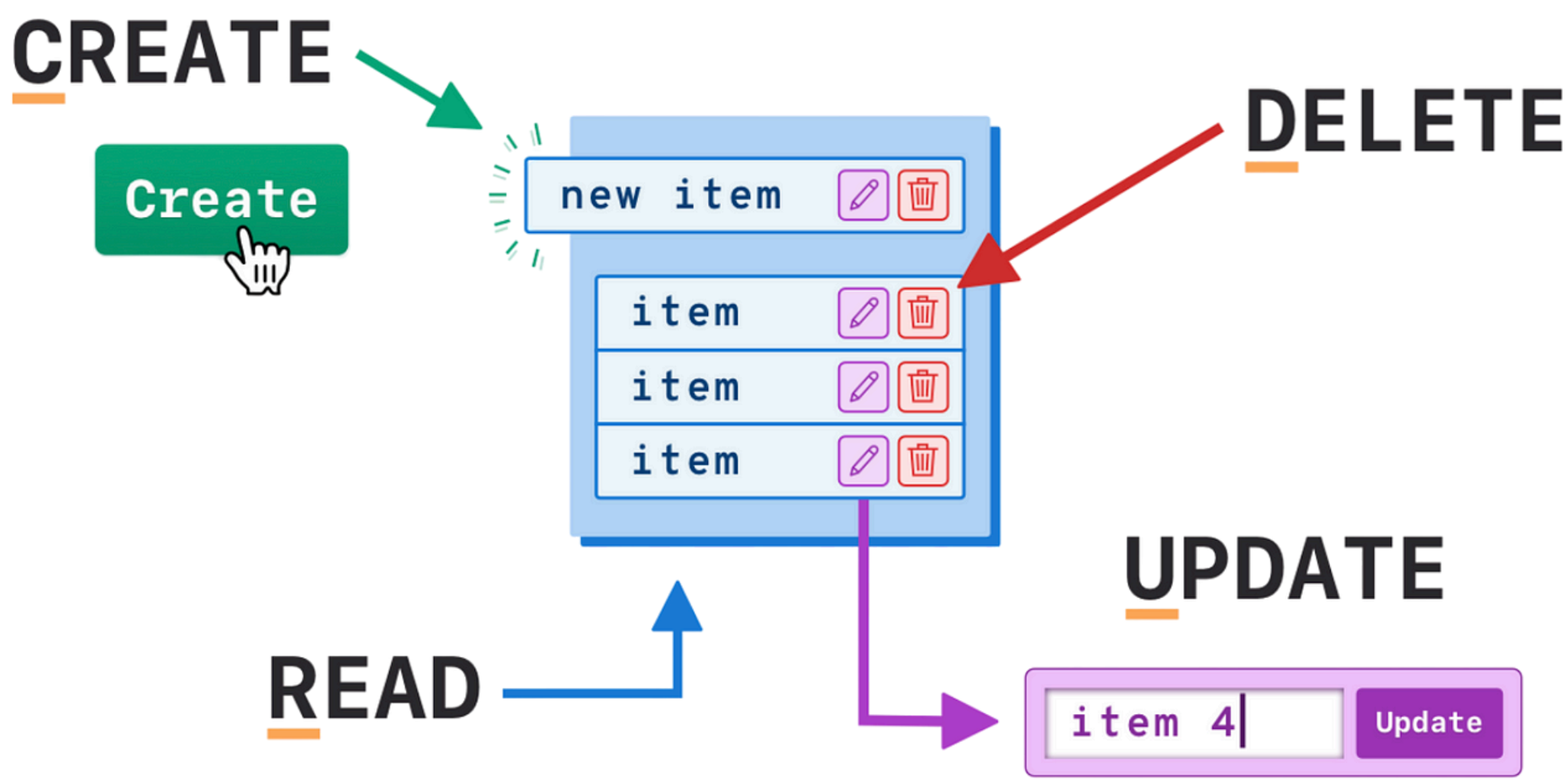
CRUD, Create, Read, Update, ve Delete kelimelerinin kısaltmasıdır. Bu, veritabanlarında veya uygulamalarda veri yönetimi için kullanılan temel dört işlemi ifade eder.

Şimdi, her bir kavramı basit bir şekilde açıklayalım:

1. **Create (Oluşturma):** Yeni bir şey yaratmak demek. Örneğin, bir arkadaşın bir deftere yeni bir sayfa eklerse, bu "Create" işlemidir.
2. **Read (Okuma):** Var olan bir şeyi okumak veya görmek demek. Örneğin, bir sayfada yazanları okumak gibi. Eğer defterin önceki sayfasını açıp oradaki yazıyı okursan, bu "Read" işlemi olur.
3. **Update (Güncelleme):** Var olan bir şeyi değiştirmek demek. Örneğin, defterdeki yazıyı silip yerine yeni bir şey yazmak, bu "Update" işlemidir.
4. **Delete (Silme):** Bir şeyi silmek demek. Yani, defterdeki sayfayı yırtmak ve tamamen silmek gibi.

Bu işlemler, bilgisayarda veri yönetmek için çok önemlidir. Örneğin, bir web sitesi üzerinden kullanıcılar kendi bilgilerini ekleyebilir (Create), var olan bilgileri görebilir (Read), bilgileri değiştirebilir (Update) ve yanlış bilgileri silebilir (Delete). Bu dört işlem sayesinde her şey düzenli bir şekilde yönetilebilir.

Yani, CRUD işlemleri, verilerin nasıl yaratılacağını, okunacağını, güncelleneceğini ve silineceğini belirleyen temel işlemlerdir.



Backend Geliştirme Temelleri

API: RESTful & GraphQL Nedir?

API (Application Programming Interface), farklı yazılımlar arasında veri alışverişi yapılmasını sağlayan bir yapıdır. Yani, bir yazılımın, başka bir yazılımın işlevlerini kullanabilmesi için bir köprü görevi görür.

RESTful API: REST (Representational State Transfer), internet üzerindeki verilerin nasıl taşındığına dair bir prensipler bütünüdür. RESTful API, bu prensiplere dayalı olarak çalışan bir API türüdür. Bu tür API'ler genellikle HTTP protokolü üzerinden çalışır ve standart işlemleri kullanır (Create, Read, Update, Delete).

GraphQL API: GraphQL, veriyi daha esnek bir şekilde almak için kullanılan bir sorgulama dilidir. RESTful API'den farklı olarak, GraphQL istemcisi (kullanıcı) sadece istediği veriyi alır, yani fazla veri almaz. Bu, daha hızlı ve daha verimli bir deneyim sağlar.

✨Hayal Ederek Anlama

RESTful API:
Bir **RESTful API,** bir restoran menüsü gibi düşünülebilir. Menüyü açıp, ne yemek istediğine karar veriyorsun. Örneğin, "Pizza" istiyorsun. Restoran garsonu, sana pizza getirir, ama sen sadece menüdeki seçilen yemekleri alırsın. Eğer sen bir yemek istemezsen, o yemekle ilgilenmezsin. Yani, her seferinde ne yemek istediğini net olarak söylersin.

RESTful API'nin Özellikleri:

- Önceden belirlenmiş standartlar (yemek menüsü gibi).
- İstemci (sen) her seferinde ne istediğini belirtir (veri seçersin).
- Veriyi istediğin şekilde almak zorundasın.

GraphQL API:
Bir GraphQL API ise, biraz daha esnek bir sistemdir. Düşün ki, bir restoran var ve garsona şöyle diyorsun: "Sadece peynirli pizza ve yanında 2 dilim mantar olsun." Burada sen, sadece tam olarak istediğin şeyi söylüyorsun ve garson da tam olarak o şekilde getiriyor. Yani fazla veri almazsın, sadece ihtiyacın olanı alırsın.

GraphQL API'nin Özellikleri:

- Daha esnek, ihtiyaca göre veri alma.
- İstemci (sen) ne kadar veri istediğini tam olarak belirtir.
- Gereksiz veri almazsın, sadece doğru veriyi alırsın.

Özetle:

- **RESTful API** genellikle daha basit ve standardize bir yapıya sahiptir, ama bazen fazla veri alabilirsin.
- **GraphQL API** daha esnek ve verimli bir yapıya sahiptir, sadece tam olarak ne istersen onu alırsın.

Yani, RESTful bir API'de her seferinde menüden bir yemek seçmek gibi, GraphQL API'de ise garsona tam olarak ne istediğini söylemek gibi düşünebilirsin!

Backend Geliştirme Temelleri

cURL Nedir?

cURL (Client URL), internet üzerinde veri almak ve göndermek için kullanılan bir komut satırı aracıdır. cURL, özellikle web üzerinde API'lere, sunuculara veya başka internet hizmetlerine veri göndermek ve almak için kullanılır. Birçok protokolü destekler, en yaygın olanları HTTP, HTTPS, FTP ve daha fazlasıdır. cURL, web sayfalarını veya sunucularla bağlantı kurarak, veri transferi işlemlerini hızlı ve verimli bir şekilde gerçekleştirir.

✨Hayal Ederek Anlama

Düşün ki, sen bir posta kuryesisin ve çok önemli bir mektubu belli bir adrese göndermen gerekiyor. cURL, işte bu mektubu doğru adrese göndermen için kullandığın bir posta arabası gibidir!

1. **Web = Mektup Göndermek** ✉️
   - İnternette bir şey göndermek veya almak, tıpkı bir mektup göndermek gibi. Her zaman doğru adrese ulaşması gerekiyor.
2. **cURL = Posta Arabası** 🚗
   - **cURL,** mektubunu doğru adrese gönderen posta arabasına benzer. Hangi adrese gitmesi gerektiğini söylersin, cURL de o adrese doğru bir şekilde gönderir.
3. **Veri Gönderme ve Alma** 📬
   - **cURL ile,** tıpkı posta arabasıyla mektup gönderirken olduğu gibi, internet üzerinden veri gönderip alabilirsin. Örneğin, bir siteye bir mesaj gönderdiğinde, cURL bunu yapar ve geri cevap aldığında, o cevabı da getirir.

Özetle:

cURL, internette veri göndermek ve almak için kullanılan bir araçtır. Tıpkı bir posta arabası gibi, gönderdiğin verileri doğru yere taşır ve gerekli yanıtları alır!

Form Nedir?

Form (Formlar), web sitelerinde kullanıcıların veri girmesine ve bu verilerin sunucuya gönderilmesine olanak tanıyan HTML öğeleridir. Genellikle giriş kutuları (input), butonlar ve seçim kutuları içerir. Kullanıcı bir formu doldurduğunda, veriler HTTP protokolü ile işlenir ve sunucuya iletilir.

Backend Geliştirme Temelleri

WebSockets Nedir?

WebSockets, web tarayıcısı ile bir sunucu arasında iki yönlü (bidirectional) ve sürekli açık bir iletişim kanalı sağlayan bir teknolojidir. WebSockets, tarayıcı ile sunucu arasında sürekli olarak veri alışverişi yapılmasını mümkün kılar, bu da özellikle gerçek zamanlı uygulamalar için oldukça kullanışlıdır.

Normalde **HTTP protokolü,** her veri isteği için bağlantıyı açıp kapatır. Ancak WebSockets, bağlantıyı sürekli açık tutarak her iki taraftan veri gönderilmesini sağlar. Bu özellik, uygulamaların çok hızlı ve düşük gecikmeli veri iletmesini mümkün kılar. WebSockets genellikle chat uygulamaları, canlı oyunlar, borsa verileri, gerçek zamanlı bildirimler gibi durumlar için kullanılır.

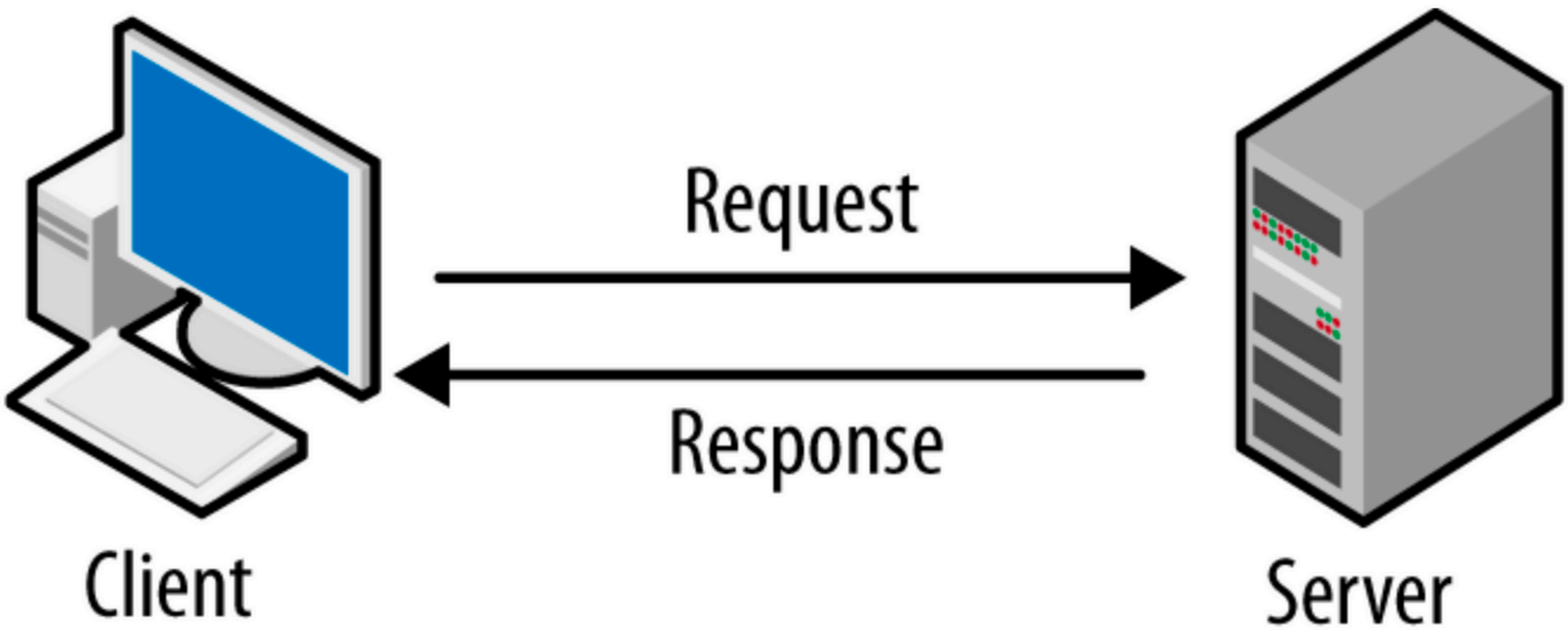
✨Hayal Ederek Anlama

Diyelim ki bir telefonla konuşuyorsun:

- **HTTP'yi kullanmak,** her konuşmada telefonu kapatıp, tekrar açarak konuşmak gibidir. Yani, her mesajı göndermek için telefonu her seferinde tekrar araman gerekir.
- **WebSockets** ise telefonu açıp sürekli açık tutarak istediğin zaman konuşabilmeni sağlar. Yani, bir kez bağlantıyı kurduğunda, her iki taraf da istedikleri zaman mesaj gönderebilir.

Örneğin, bir chat uygulamasında WebSockets kullanıyorsan, mesajlar anında gider ve hemen gelir. Her mesaj için ayrı ayrı bağlantı kurmazsın, çünkü bağlantı sürekli açıktır.

Sonuç olarak, WebSockets, gerçek zamanlı veri iletimi sağlayarak uygulamaların daha hızlı ve etkileşimli olmasına olanak tanır!



Backend Geliştirme Temelleri

JWT, OAuth, OpenID, SAML Nedir?

Bu kavramlar, web uygulamalarında kimlik doğrulama ve yetkilendirme (authentication and authorization) işlemlerini yönetmek için kullanılan teknolojilerdir. Birçok uygulama, kullanıcıların kimliğini doğrulamak ve onlara belirli erişim hakları vermek için bu yöntemleri kullanır.

1. JWT (JSON Web Token)

JWT, iki sistem arasında güvenli bir şekilde bilgi taşıyan bir token formatıdır. Bu token, genellikle bir kullanıcı giriş yaptıktan sonra sunucu tarafından üretilir ve sonrasında kullanıcının kimliğini doğrulamak için her istekte sunucuya gönderilir.

- Kimlik doğrulama için kullanılır.
- Veri taşımak için güvenli bir yol sağlar.
- Üç bölümden oluşur: Header (başlık), Payload (yük) ve Signature (imza).

2. OAuth

OAuth, bir kullanıcının, bir uygulamaya diğer bir uygulamanın verilerine güvenli bir şekilde erişebilmesi için bir protokoldür. Yani, OAuth, başka bir hesabın bilgilerini paylaşmadan kullanıcıya izin verir.

Örneğin, bir kullanıcı Google hesabını kullanarak başka bir uygulamaya giriş yapabilir.

- Yetkilendirme sağlar, ancak kimlik doğrulama yapmaz.
- Bir hizmete başka bir hizmet üzerinden yetki verme protokolüdür.

3. OpenID

OpenID, kullanıcıların tek bir kimlik ile birden fazla web sitesine veya uygulamaya giriş yapabilmesini sağlayan bir protokoldür. Bu, kullanıcıların tek bir hesapla farklı platformlarda oturum açmasına olanak tanır.

- Kimlik doğrulama için kullanılır.
- OAuth'un üzerine inşa edilmiştir ve kullanıcıyı tanımak için daha fazla bilgi sunar.

4. SAML (Security Assertion Markup Language)

SAML, özellikle kurumsal uygulamalarda kullanılan, federasyon kimlik doğrulaması yapan bir protokoldür. Bu, bir kullanıcının bir şirketin farklı sistemlerine veya platformlarına tek bir kimlik ile erişmesini sağlar.

- Kimlik doğrulama sağlar ve çok büyük organizasyonlarda kullanılır.

Backend Geliştirme Temelleri

✨Hayal Ederek Anlama

Diyelim ki okulda öğrenci kartı kullanarak bir odaya giriyorsun:

- **JWT (JSON Web Token):**
  - Okul kartın gibi düşün. Okul, sana kartı verdiğinde onunla her odaya girebilirsin. Her oda, bu kartı doğrulayarak sana giriş izni verir.
  - Her gittiğinde okul kartını kullanarak, okul bunu her seferinde kontrol eder.
- **OAuth:**
  - Diyelim ki başka bir arkadaşın, sana sınıfına girmesi için izin veriyor. Ama o arkadaşının kartını almıyorsun. Sen sadece izin alarak sınıfına girebiliyorsun.
  - Burada senin kimliğini doğrulamaz, sadece izin verir.
- **OpenID:**
  - Tek bir öğrenci kartı ile birçok farklı odadan geçebilmen gibi. Okulun tüm odalarında aynı kartı kullanarak her yere girebilirsin. Yani, her okula girdiğinde tek bir kartla kimliğin doğrulanır.
  - Yani tek bir kimlik ile birçok yerde oturum açarsın.
- **SAML:**
  - Büyük bir okul kompleksi düşün, burada her bina farklı bir kimlik doğrulama sistemine sahip. Ama bir okul kartınla tüm binalara giriş yapabilirsin.
  - Yani, kurum içinde tüm sistemlerde tek bir kartla işlem yapabilirsin.

Sonuç olarak, bu teknolojiler, kullanıcıların farklı sistemlere güvenli ve hızlı bir şekilde erişebilmesini sağlayan yöntemlerdir!

Backend Geliştirme Temelleri

MVC ve Backend Mimarileri Nedir?

1. MVC (Model-View-Controller) Mimarisi

MVC, bir yazılım uygulamasının yapısını üç ana bileşene ayıran bir tasarım desenidir:

- **Model (M):** Uygulamanın verilerini ve iş mantığını tutar. Veritabanıyla etkileşimde bulunur.
- **View (V):** Kullanıcının gördüğü ve etkileşime geçtiği arayüzdür. Veriler Model'den alınır ve ekranda gösterilir.
- **Controller (C):** Kullanıcıdan gelen istekleri alır, bunları işler ve gerekli değişiklikleri Model ile View üzerinde yapar.

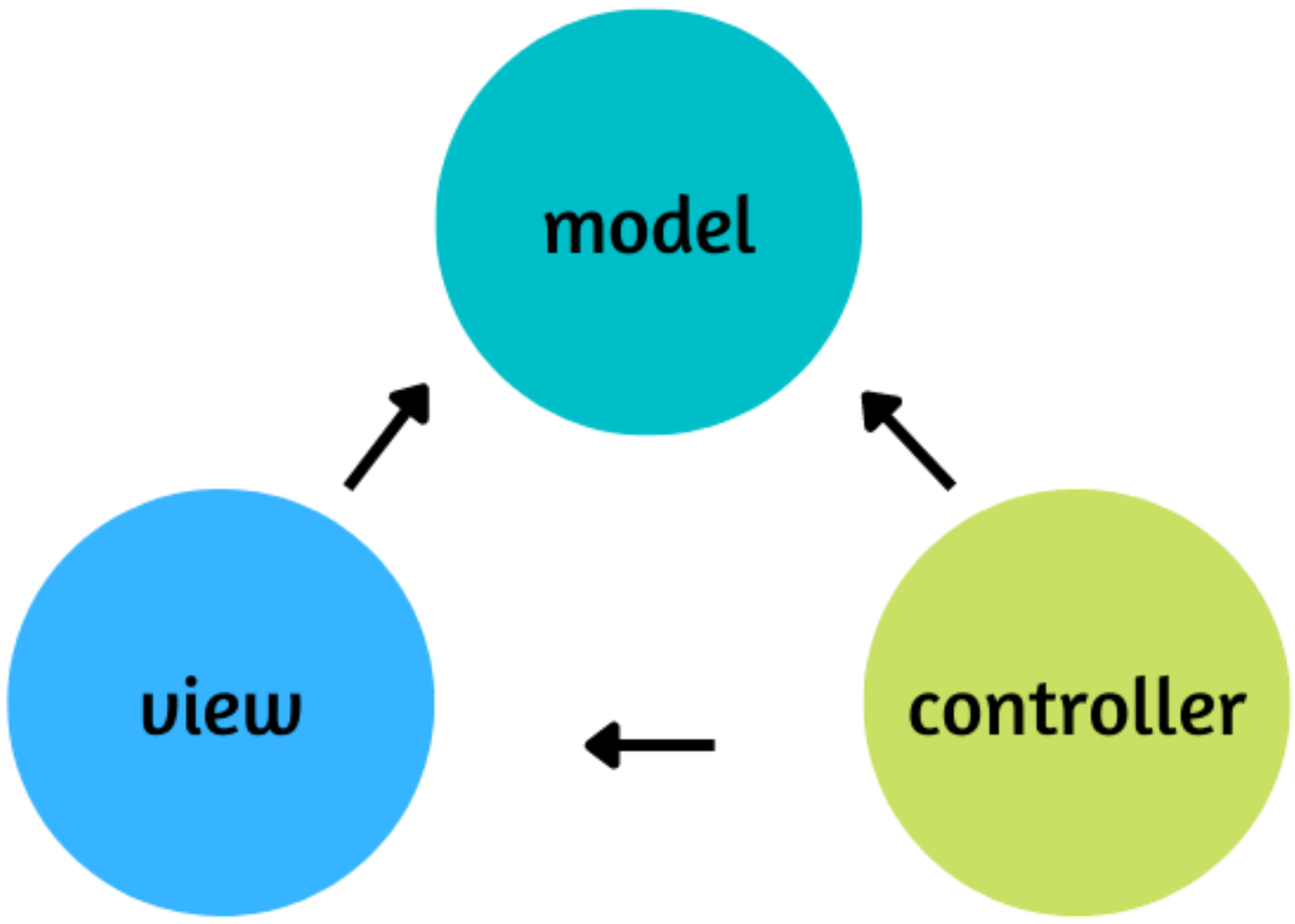
MVC Mimarisi, web uygulamalarını daha düzenli hale getirir, çünkü her bir bileşen farklı sorumluluklara sahiptir ve birbirinden bağımsız olarak geliştirilebilir.

2. Backend Mimarileri

Backend, web uygulamalarının sunucu tarafında çalışan kısmıdır. Backend mimarileri, bu sunucuların nasıl çalıştığını, veritabanlarıyla nasıl etkileşime girdiğini ve kullanıcı isteklerine nasıl yanıt verdiğini belirler.

Başlıca backend mimarileri şunlardır:

- **Monolithic Architecture (Tek Parçalı Mimari):** Tüm uygulama tek bir yapı içinde bulunur. Hem iş mantığı hem de kullanıcı arayüzü birlikte çalışır.
- **Microservices Architecture (Mikro Hizmetler Mimari):** Uygulama küçük bağımsız hizmetlere (servislere) bölünür. Her bir hizmet kendi görevini yerine getirir ve diğer hizmetlerle iletişim kurar.
- **Serverless Architecture (Sunucusuz Mimari):** Uygulama, bir sunucuya bağlı olmadan bulut platformları üzerinde çalışır ve gerektiğinde kaynak kullanımı otomatik olarak ölçeklenir.



Backend Geliştirme Temelleri

✨Hayal Ederek Anlama

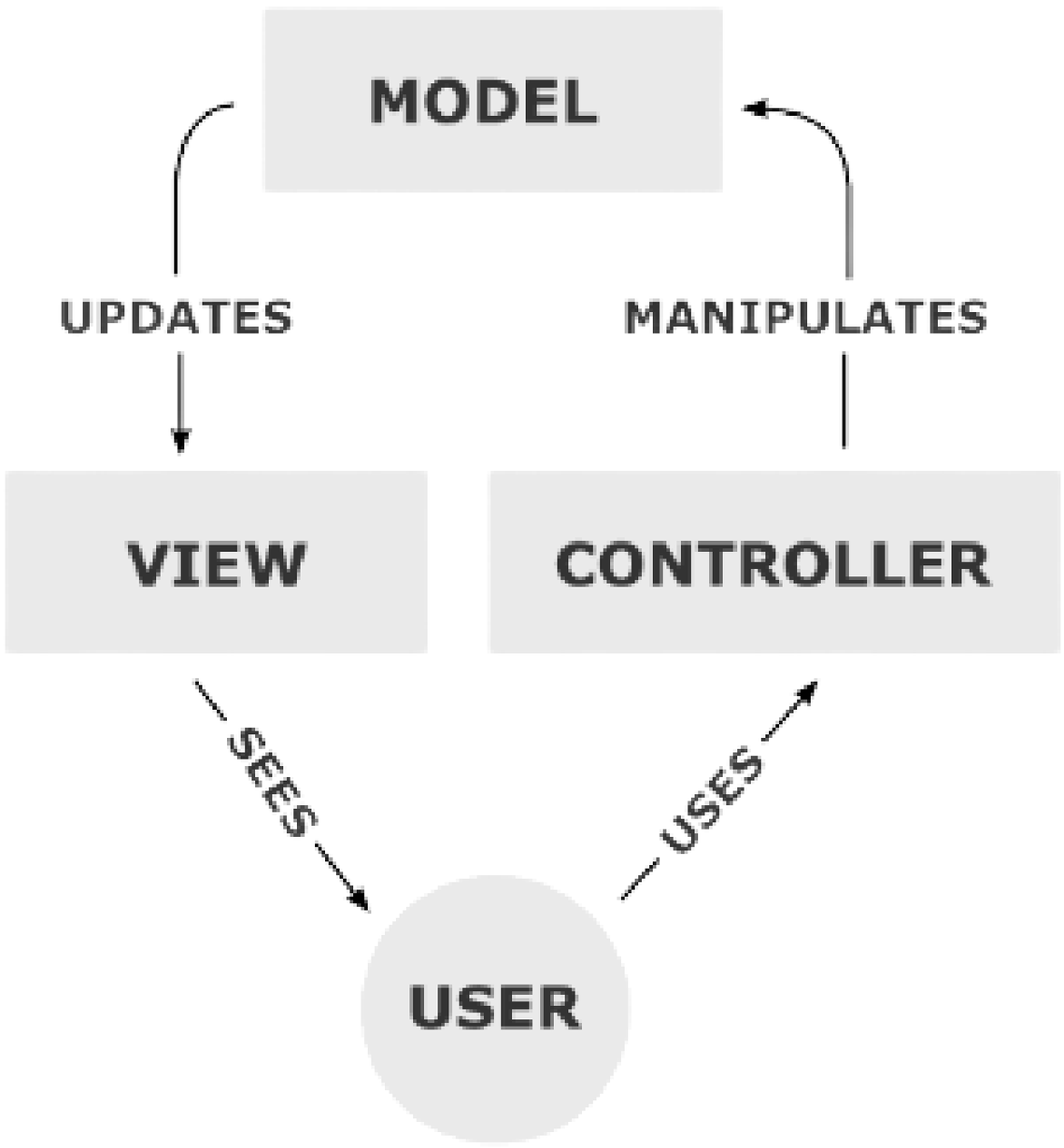
MVC'yi Açıklayalım:
Diyelim ki bir restoranda garson olarak çalışıyorsun:

- **Model (M):** Mutfakta, yemek tarifleri ve malzemeler ile ilgilenir. Hangi yemeğin nasıl yapılacağını ve hangi malzemelere ihtiyacı olduğunu bilirsin.
- **View (V):** Menü gibi, müşterinin görebileceği şeyler. Müşterinin siparişini aldığı menüyü gösterirsin.
- **Controller (C):** Garson olarak, sipariş alır ve mutfağa iletirsin. Mutfağa gittiğinde gerekli yemek tarifini alır, yemek hazır olduğunda da müşteriye getirirsin.

Backend Mimarilerini Açıklayalım:
Şimdi ise bir okulun idareci ofisini düşün:

- **Monolithic Architecture (Tek Parçalı Mimari):** Tüm okul işleri tek bir ofiste yapılır. Hem öğrencilerle, hem öğretmenlerle, hem de malzeme işlerine bakılır. Yani her şey tek bir yerden yönetilir.
- **Microservices Architecture (Mikro Hizmetler Mimari):** Okul, her biri farklı bir görevle ilgili olan birkaç ofise ayrılmıştır. Bir ofis öğrenci işleriyle ilgilenirken, başka bir ofis öğretmen işleriyle ilgilenir. Her ofis kendi işini yapar ve gerektiğinde diğer ofislerle bilgi alışverişi yapar.
- **Serverless Architecture (Sunucusuz Mimari):** Okulda öğretmenler ve öğrenciler her şeyi internet üzerinden yapar. Öğretmenler, internet üzerinde öğrencilere ödev verir ve öğrenciler de yine internet üzerinden bu ödevleri teslim eder. Her şeyin nasıl çalıştığını internet platformları otomatik olarak halleder.

Sonuç olarak, MVC, yazılımın daha düzenli ve yönetilebilir olmasını sağlarken, Backend Mimarileri uygulamanın nasıl çalıştığını, ölçeklendiğini ve yönetildiğini belirler!



Backend Geliştirme Temelleri

Backend Framework'leri ve Tasarım Prensipleri Nedir?

1. Backend Framework'leri

Backend frameworkleri, bir web uygulamasının sunucu tarafında çalışan kısmını hızlı ve verimli bir şekilde geliştirmenize yardımcı olan yazılım çerçeveleridir. Bu framework'ler, geliştiricilerin veritabanı bağlantıları, kullanıcı kimlik doğrulaması, API oluşturma gibi temel işlemleri daha hızlı ve daha düzenli bir şekilde yapmalarını sağlar.

Yaygın kullanılan backend framework'leri şunlardır:

- Express.js (Node.js)
- Django (Python)
- Ruby on Rails (Ruby)
- Laravel (PHP)
- Spring (Java)

2. Tasarım Prensipleri

Tasarım prensipleri, yazılım geliştirme sürecinde uygulamaların daha sağlam, esnek ve sürdürülebilir olmasını sağlamak için izlenen temel kurallardır. Tasarım prensiplerinin amacı, kodun daha kolay okunabilir, yönetilebilir ve genişletilebilir olmasını sağlamaktır.

Öne çıkan bazı temel prensipler şunlardır:

- **SOLID:** Nesne yönelimli programlamada, yazılımın daha sağlam ve esnek olmasını sağlayan beş temel prensipten oluşur.
- **KISS (Keep It Simple, Stupid):** Karmaşıklıktan kaçının ve mümkün olduğunca basit çözümler üretin.
- **DRY (Don't Repeat Yourself):** Aynı kodu birden fazla kez yazmaktan kaçının; kod tekrarını azaltın.
- **YAGNI (You Aren't Gonna Need It):** Gelecekte gerekecek şeyleri kodlamadan önce gerekliliğini doğrulayın.



Backend Geliştirme Temelleri

✨Hayal Ederek Anlama

Backend Framework'lerini Açıklayalım:
Diyelim ki bir pizza restoranı açıyorsun.

- Backend framework bir restoran mutfağında kullandığın standart araçlar ve talimatlar gibidir.
    - **Express.js** gibi bir framework, pizzaları yaparken pizza hamuru açma makinesi gibi işler görür. Sen bu makineleri kullanarak hızlıca pizza yapabilirsin.
    - **Django gibi bir framework,** pizza yapımında özel sos tarifini verir. Kendi tarifini oluşturmak yerine hazır bir tarif kullanarak zaman kazanırsın.
    - **Laravel,** senin pizzaları hızlıca sipariş alıp teslim etmeni sağlayacak bir sistem gibi çalışır.

Tasarım Prensiplerini Açıklayalım:
Şimdi ise, pizza restoranındaki işi düzenli ve verimli hale getirmek için kurallar oluşturalım:

- **SOLID:**
    - Her çalışan sadece bir işi yapsın. Örneğin, bir çalışan sadece pizza hamuru açsın, diğeri malzemeleri koysun, diğeri ise pizzayı pişirsin. Böylece her şey düzenli olur ve işler birbirine karışmaz.
- **KISS (Keep It Simple, Stupid):**
    - Pizza tarifini basit tut. Çalışanlara karmaşık tarifler vermek yerine, daha kolay bir tarifle hızlıca pizza yapmalarını sağla.
- **DRY (Don't Repeat Yourself):**
    - Tarifi her seferinde yeniden yazma. Aynı tarifi tekrar tekrar yazmak yerine, bir defa yazıp çalışanların o tarife uymasını sağla.
- **YAGNI (You Aren't Gonna Need It):**
    - Gelecekte ihtiyacın olmayacak malzemeleri sipariş etme. Sipariş verdiğin malzemeleri yalnızca ihtiyacın olan kadar al, gereksiz harcamalardan kaçın.

Sonuç olarak, Backend framework'leri uygulamanı daha hızlı ve düzenli yapmanı sağlarken, tasarım prensipleri uygulamanın uzun vadede daha sağlam ve sürdürülebilir olmasını sağlar!

Backend Geliştirme Temelleri

RESTful Nedir?

RESTful (Representational State Transfer), web servisleri için bir tasarım yaklaşımıdır. Bu yaklaşımda, veriler genellikle HTTP protokolü üzerinden erişilir ve bu veriler, belirli standartlara göre yönetilir. RESTful servisler, istemci ve sunucu arasında bağımsız bir iletişim sağlar ve verileri kaynaklar (resources) olarak temsil eder. Her kaynağın bir URL'si vardır ve bu kaynaklarla etkileşim HTTP metodları (GET, POST, PUT, DELETE) ile yapılır.

✨Hayal Ederek Anlama

Düşün ki bir bakkal dükkanına gidiyorsun. Bu dükkanda birçok farklı ürün var: ekmek, süt, oyuncaklar... Şimdi, eğer bir ürünü almak istersen, bakkal senin ne almak istediğini sorar ve o ürünün adını söylemeni ister. Mesela, "Ekmek istiyorum" dediğinde, bakkal sana ekmek verir. Eğer oyuncak almak istersen, o zaman oyuncak ismini söylersin ve sana onu verir.

RESTful da tıpkı böyle çalışır! Bir web sitesi bir şey almak istediğinde, o şeyi belirtir ve HTTP yöntemleriyle istek yapar. Mesela, "Bana şu veriyi ver" der (bu GET isteği). Eğer yeni bir şey eklemek isterse, "Yeni veri eklemek istiyorum" der (bu POST isteği). Eğer bir veriyi güncellemek isterse, "Şu veriyi değiştir" der (bu PUT isteği). Ve son olarak, bir veriyi silmek için "Bunu kaldır" der (bu DELETE isteği).

Yani, RESTful'da her şey, sanki bir bakkalda alışveriş yapıyormuşsun gibi, ürünler ve işlemler üzerinden düzenlenir!



Backend Geliştirme Temelleri

JSON Nedir?

JSON (JavaScript Object Notation), veri iletimi ve saklama için kullanılan, insan tarafından okunabilir bir veri formatıdır. JSON, verileri anahtar-değer çiftleri olarak saklar ve genellikle web uygulamaları arasında veri gönderimi için kullanılır. JSON, JavaScript ile uyumlu olsa da, başka programlama dillerinde de kullanılabilir. JSON, veriyi basit, hafif ve kolay anlaşılır bir şekilde temsil eder, bu yüzden veri paylaşımı için çok yaygın olarak kullanılır.

✨Hayal Ederek Anlama

Düşün ki, bir arkadaşına hediye almak istiyorsun. Ama ona ne aldığını yazılı olarak açıklaman gerek. Bunun için bir liste yapıyorsun ve her bir şeyin ne olduğunu ve ne kadar olduğunu yazıyorsun.

Örneğin, şöyle bir liste olabilir:

- Gül: 1 tane
- Çikolata: 2 tane
- Balon: 1 tane

Bu listeyi arkadaşına veriyorsun. Arkadaşın, yazdığın her şeyin ne olduğunu ve miktarını kolayca anlayabiliyor.

JSON de buna benzer bir şeydir! JSON, veriyi listeler ve açıklamalar şeklinde düzenler. Bu sayede bir bilgisayar ya da başka bir program, veriyi çok rahat bir şekilde okuyabilir ve anlayabilir. Yani, JSON bir liste gibi düşünülebilir; burada veriler anahtar (ne olduğu) ve değer (miktarı veya içeriği) şeklinde sıralanır.



Veritabanı Yönetimi

Veritabanı Nedir?

Veritabanı, bilgilerin düzenli bir şekilde saklandığı, bilgileri kolayca bulup erişebileceğimiz bir yerdir. Örneğin, bir okulun veritabanında öğrencilerin isimleri, notları ve iletişim bilgileri gibi veriler düzenli olarak saklanır. Veritabanları, büyük miktarda veriyi güvenli bir şekilde depolayabilir ve bu verilere hızlıca erişim sağlar. Veritabanı, genellikle tablo şeklinde düzenlenir. Bu tablolarda veriler, satırlar (kayıtlar) ve sütunlar (alanlar) şeklinde düzenlenir.

SQL Nedir?

SQL (Structured Query Language), veritabanındaki verilere erişmek, düzenlemek ve yönetmek için kullanılan bir dildir. SQL ile veritabanından veri alabilir, ekleyebilir, silebilir veya güncelleyebilirsiniz. SQL, veritabanı ile konuşmamızı sağlayan dil gibidir.

✨Hayal Ederek Anlama

Veritabanı tıpkı bir kitaplık gibidir. Bu kitaplıkta çok fazla kitap (bilgi) vardır ve her kitabın içinde sayfalar (bilgiler) vardır. Eğer bir konuda bilgi arıyorsan, kitaplıkta arama yaparak o bilgilere kolayca ulaşabilirsin.

SQL ise, bu kitaplıkla konuşarak hangi kitabı ve hangi sayfayı istediğini garsona söyler gibi bir dil! Mesela, "Bu kitabı bana getir" ya da "Bu sayfayı değiştir" gibi komutlar verirsin.

Yani, veritabanı bilgileri düzenli tutan bir kitaplık, SQL ise bu kitaplıkla konuşmamızı sağlayan dildir!



Veritabanı Yönetimi

NoSQL ve SQL Farkları Nedir?

SQL (Structured Query Language), veritabanında verilerin düzenli ve tablolar şeklinde saklandığı bir sistemdir. SQL veritabanları, her şeyin belirli bir yapıya sahip olduğu ve birbirine bağlı olduğu ilişkisel veritabanlarıdır. SQL, genellikle verilerin karmaşık ilişkilerle düzenlendiği durumlarda kullanılır. Örneğin, bir okulun öğrencilerini, derslerini ve öğretmenlerini düzenlemek için SQL veritabanları kullanılır.

NoSQL (Not Only SQL), veritabanlarının esnek ve yapısız olduğu bir türdür. NoSQL veritabanlarında veriler tablolar yerine farklı şekillerde saklanabilir. Genellikle büyük veriler veya çok hızlı veri işlemeye ihtiyaç duyan uygulamalar için kullanılır. NoSQL, veritabanlarında şemaya ihtiyaç duymaz yani veriler çok farklı şekilde saklanabilir. Örneğin, sosyal medya platformlarında, kullanıcıların paylaşımlarını ve mesajlarını saklamak için NoSQL veritabanları tercih edilir.

✨Hayal Ederek Anlama

SQL, tıpkı düzenli ve sıralı bir kütüphane gibidir. Her kitap, düzgün bir şekilde yerleştirilmiş ve her kitapta neler olduğunu tam olarak biliyoruz. Yani kitapların, yazarları, başlıkları, sayfa numaraları gibi her şeyi belli. Her şeyin düzenli olduğu bir yer.

NoSQL ise daha esnek ve rahat bir yer gibi. Burası bir tür serbest kütüphane. Kitaplar her türlü şekilde olabilir, bazıları çok düzenli, bazılarıysa dağınık. Yani burada her şeyin yapısı belli değil, kitaplar serbest bir şekilde düzenlenmiş ve daha fazla özgürlük var. Çok büyük ve hızlı kitaplar var, ama her kitabın nasıl düzenlendiği konusunda bir kısıtlama yok.

Kısacası, SQL düzenli ve yapılandırılmış bir kütüphane gibi, NoSQL ise daha esnek ve hızlı çalışan bir kütüphane gibi!



Veritabanı Yönetimi

ORM (Object-Relational Mapping) Nedir?

ORM (Object-Relational Mapping), bir programlama dili ile veritabanı arasında köprü kurarak, veritabanı işlemlerini daha kolay ve anlaşılır hale getiren bir tekniktir. ORM, veritabanındaki tablolarda bulunan verileri, programlama dilindeki nesnelerle (objelerle) ilişkilendirir. Bu sayede yazılımcılar, veritabanı ile çalışırken SQL kodları yazmak yerine, daha kolay bir şekilde programlama dili kullanarak işlem yapabilirler. **Örneğin,** PHP'de Eloquent, JavaScript'te TypeORM gibi ORM kütüphaneleri, veritabanındaki verilerle nesneler arasında dönüşüm yaparak, yazılımcıların işini kolaylaştırır.

✨Hayal Ederek Anlama

ORM, veritabanı ve programlama dili arasında bir çevirmen gibidir. Düşün ki, veritabanında büyük bir kitaplık var ve kitaplar (veriler) çok karışık bir şekilde yerleştirilmiş. Ama sen bunları okumak için kitaplara isim takmak ve onları daha düzenli bir şekilde bulmak istiyorsun.

ORM, işte burada devreye giriyor. Kitapların isimlerini ve yerlerini sana düzgün bir şekilde gösteren bir rehber gibi çalışıyor. Sen kitaplara (verilere) nesne (object) gibi yaklaşarak, veritabanındaki karmaşayı görmeden işlerini yapabiliyorsun.

Yani ORM, veritabanındaki karmaşayı gizler ve sana sadece basit ve anlaşılır bir yol sunar. Artık veritabanı ile uğraşmak yerine, sadece kitapları (nesneleri) yönetebilirsin!



Veritabanı Yönetimi

Veritabanı Optimizasyonu (Indexing, Query Optimization) Nedir?

Veritabanı optimizasyonu, veritabanlarının daha hızlı ve daha verimli çalışmasını sağlamak için yapılan işlemlerdir. Veritabanları, çok büyük miktarda veriyi depoladığında, bu verilere erişim zaman alabilir. Veritabanı optimizasyonu ile bu erişim süreleri kısaltılır ve veritabanı daha hızlı çalışır.

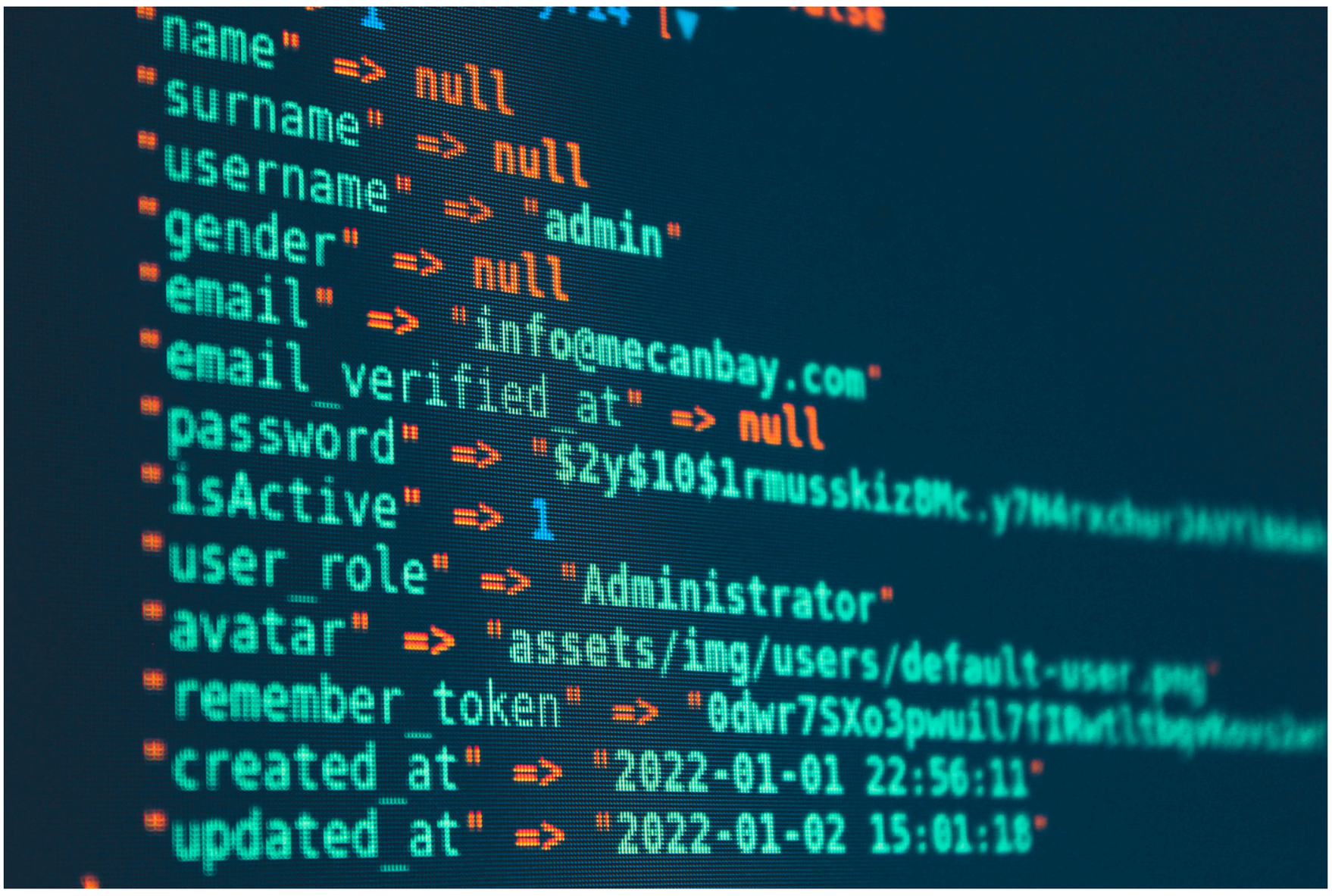
- **Indexing (İndeksleme):** Veritabanındaki verilere hızlı bir şekilde erişebilmek için indeks adı verilen bir yapının oluşturulmasıdır. İndeks, verilerin belirli bir sıra ve düzen içinde saklanmasını sağlar. Tıpkı bir kitabın arkasındaki indeks kısmı gibi, istediğiniz veriyi hızlıca bulmanızı sağlar.
- **Query Optimization (Sorgu Optimizasyonu):** Veritabanına yapılan sorguların daha verimli çalışmasını sağlamak amacıyla, sorgu yazımının düzenlenmesidir. Amaç, veritabanına yapılan sorguların daha hızlı çalışması için gereksiz işlemleri azaltmak ve daha az kaynak kullanarak sonuç almak.

✨Hayal Ederek Anlama

Veritabanı optimizasyonu, tıpkı büyük bir kütüphaneyi düzenlemek gibidir. Düşün ki kütüphanenin içinde binlerce kitap var ve bir tane kitap arıyorsun. Eğer kitaplar rastgele yerleştirilmişse, istediğin kitabı bulman çok uzun sürer. Ama kitapları düzenli bir şekilde sıralarsan, hemen istediğin kitabı bulabilirsin. İşte bu düzenleme işlemi, indexing (indeksleme) işidir!

Query Optimization (Sorgu optimizasyonu) ise, kütüphanedeki kitapları bulmak için daha hızlı bir yol bulmak gibidir. Örneğin, kitapları daha az yerden arayarak zaman kazanabilirsin. Yani gereksiz aramalardan kaçınarak, sadece doğru yere bakmak işinizi hızlandırır.

Veritabanı optimizasyonu, bilgileri düzenleyip daha hızlı bulmamızı sağlayan bir sistemdir!



Veritabanı Yönetimi

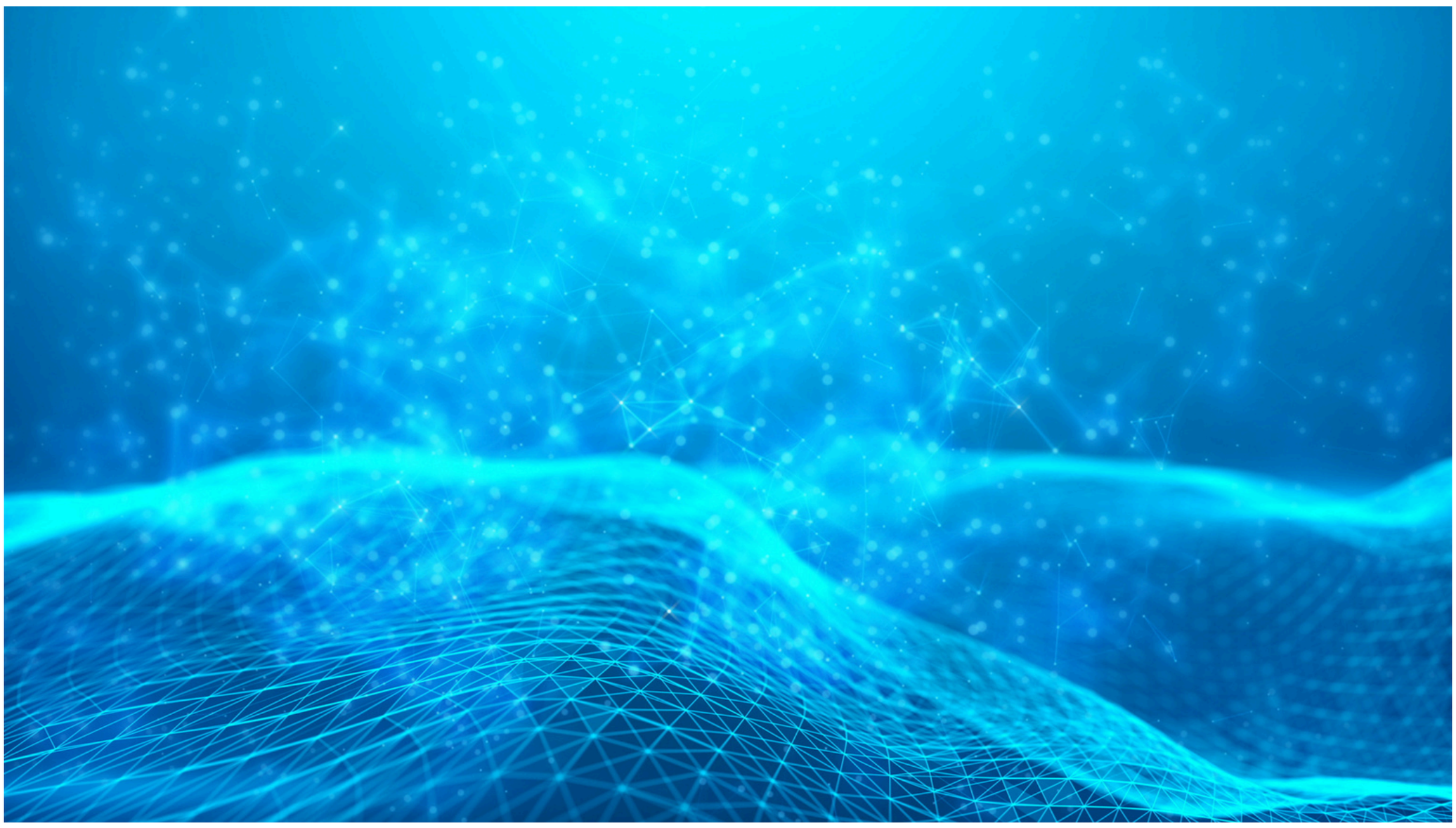
Redis & Cache Nedir?

Redis, çok hızlı çalışan, veri saklama ve veri erişim sistemi sağlayan bir yazılımdır. Özellikle veri önbellekleme (cache) işlemleri için çok kullanılır. Redis, veriyi bellekte (RAM) tutarak, veriye çok hızlı bir şekilde erişilmesini sağlar. Web uygulamalarında, veri tabanından sık sık aynı veriyi sorgulamak yerine, önbellekte tutarak hız kazanılır. Redis, verilerin anahtar-değer (key-value) çiftleri şeklinde saklanmasını sağlar ve çok hızlı çalıştığı için yüksek performanslı uygulamalarda tercih edilir.

✨Hayal Ederek Anlama

Düşün ki bir kütüphaneye gidiyorsun ve her seferinde bir kitap almak için uzun süre beklemek zorunda kalıyorsun. Ama bir gün, en popüler kitaplar hemen önünde bulunuyor ve her seferinde yeni bir kitap almak için zaman kaybetmiyorsun. Bu kitaplar hep önceden hazırlanmış ve hemen alabileceğin şekilde yerleştirilmiş.
İşte Redis de tam böyle çalışıyor!

Web siten bir şey soruyor (mesela, hava durumu), eğer bu bilgi daha önce çok hızlı bir şekilde elde edilmişse, artık her defasında yeniden aramıyor. Redis, bu bilgiyi çok hızlı bir şekilde bulmanı sağlar ve her seferinde tekrar gözden geçirme gereksiz olur. Cache kullanımı ise, önceden hazırlanan bilgilerin bir kenarda saklanması gibidir. Böylece, bilgiye daha hızlı erişebilirsin ve her defasında yeniden işlem yapmana gerek kalmaz!



Web Güvenliği ve Performans Optimizasyonu

Session Nedir?

Session (Oturum), bir kullanıcının bir web sitesiyle gerçekleştirdiği etkileşimin başlangıcından bitişine kadar geçen süreyi temsil eder. Web siteleri genellikle her bir kullanıcıyı tanımak ve o kullanıcıyla ilgili verileri saklamak için session kullanır. Örneğin, bir kullanıcı bir e-ticaret sitesine giriş yaptıktan sonra, alışveriş sepetindeki ürünlerin saklanması veya kullanıcının giriş yapıp yapmadığının kontrol edilmesi gibi işlemler session ile yapılır. Session, verileri sunucuda saklar ve kullanıcı siteyi terk ettiğinde veya oturum sona erdiğinde silinir.

✨Hayal Ederek Anlama

Düşün ki bir arkadaşınla bir oyun oynuyorsun. Oyunun başında sen ve arkadaşın oyuna başlıyorsunuz, birlikte ilerliyorsunuz ve oynadığınız süre boyunca her şeyin kaydı tutuluyor. Mesela, oyunun başında hangi karakteri seçtiğini ya da hangi seviyeyi geçtiğini hatırlıyorsun. Ama oyunu kapattığında tüm bu bilgiler kayboluyor. İşte Session, tıpkı oyunda senin ve arkadaşının oynamaya başladığı andan oyunu kapatana kadar olan süreyi hatırlayan bir şeydir. Web siteleri, sen bir şeyler yaparken seninle ilgili bazı bilgileri saklar ve sen o siteyi gezdiğin sürece bu bilgileri unutmaz. Ama sen siteyi kapattığında ya da bir süre sonra site seni unutacak. Session, sitedeki gezintin sırasında seni tanıyan bir bellek gibidir!



Web Güvenliği ve Performans Optimizasyonu

Cookie Nedir?

Cookie, bir web sitesinin kullanıcı hakkında bazı bilgileri kullanıcının tarayıcısında saklamak için kullandığı küçük bir dosyadır. Web sitesi, bu dosyaları kullanıcıyı tanımak, siteyi kişiselleştirmek veya kullanıcının davranışlarını takip etmek amacıyla kullanabilir. Cookies, genellikle oturum bilgileri, kullanıcı tercihleri veya alışveriş sepeti verileri gibi bilgileri içerir. Tarayıcılar bu bilgileri saklar ve web sitesine her yeni sayfa yüklemesinde geri gönderir.

✨Hayal Ederek Anlama

Düşün ki bir oyun parkına girdin ve oyun oynamaya başladın. Oyun oynarken, parkın gizli bir arkadaşı seni çok seviyor ve her seferinde oynadığın oyunları not alıyor. Ama bu arkadaş, sen parkı terk ettiğinde de seni unutmasın diye, bir kağıt bırakıyor.
Bu kağıt sayesinde, sen bir başka gün tekrar parka gittiğinde, arkadaşın seni hatırlıyor ve daha önce oynadığın oyunları sana tekrar öneriyor. Yani, o kağıt sayesinde sen parka her geldiğinde, ne yaptığını unutmaz.

Cookie de tam olarak böyle çalışıyor! Web siteleri, tarayıcında küçük bir kağıt bırakıyor. Bu kağıt sayesinde, web sitesi seni tekrar gördüğünde, senin önceki gezintilerini ve tercihlerini hatırlayabiliyor. Yani, bir sonraki sefer siteye gittiğinde, seni unutmaz ve daha önce ne yaptığını hatırlayarak sana daha iyi bir deneyim sunar.



Web Güvenliği ve Performans Optimizasyonu

CORS, CSRF, XSS Nedir?

Bu kavramlar, web güvenliği ile ilgili önemli terimlerdir. Web uygulamalarında kullanıcıların verilerinin güvenli bir şekilde korunması için çeşitli güvenlik önlemleri ve mekanizmalar vardır. CORS, CSRF ve XSS bu güvenlik önlemleriyle ilgilidir.

1. CORS (Cross-Origin Resource Sharing)

CORS, bir web sayfasının, farklı bir kaynaktan (domain) gelen veriye erişebilmesini sağlayan bir güvenlik mekanizmasıdır. Web tarayıcıları, aynı kaynağa ait olmayan veriye doğrudan erişimi engeller. CORS, bu kısıtlamayı aşmak için güvenli bir şekilde dış kaynaklardan veri alınmasına izin verir.

- **Neden kullanılır?** Farklı siteler arasında veri paylaşımını güvenli hale getirmek için.

2. CSRF (Cross-Site Request Forgery)

CSRF, bir kullanıcının bilgisini veya yetkisini kullanarak, o kullanıcının tarayıcısı üzerinden izinsiz bir işlem yapılması anlamına gelir. Örneğin, bir kullanıcının hesabında, kullanıcının bilgisi olmadan para transferi yapılması.

- **Neden tehlikelidir?** Kötü niyetli bir kişi, kullanıcının giriş yapmış olduğu bir web uygulamasına sahte talepler göndererek istenmeyen işlemler yapabilir.

3. XSS (Cross-Site Scripting)

XSS, bir saldırganın, kullanıcıların web sayfalarına zararlı JavaScript kodları eklemesi yoluyla gerçekleştirilen bir saldırıdır. Bu zararlı kod, kullanıcının tarayıcısında çalışarak kişisel verileri çalabilir veya sayfanın içeriğini değiştirebilir.

- **Neden tehlikelidir?** XSS, kötü amaçlı kodlar ekleyerek veri çalabilir veya kullanıcıları manipüle edebilir.

Web Güvenliği ve Performans Optimizasyonu

✨Hayal Ederek Anlama

CORS'u Açıklayalım:
Diyelim ki bir kütüphaneye gidiyorsun ve farklı bir kütüphanedeki kitapları almak istiyorsun.

- **CORS,** kütüphanenin diğer kütüphaneye kitap göndermesine izin veren bir kurallar setidir. Ama sadece güvenilir kütüphaneler bu kitapları alabilir, diğerleri alamaz.
- Yani, farklı yerlerden veri almak, ama güvenliği sağlamak için CORS kullanılır.

CSRF'i Açıklayalım:
Bir doktorun bilgisini çaldığını ve onun hesabına girip bazı işlemler yaptığını düşün.

- **CSRF,** kötü niyetli bir kişinin senin kimliğini kullanarak başka bir işlem yapması gibidir. Yani senin hesabını, bilgin olmadan ve iznin olmadan başkası kullanır.
- **CSRF,** senin hesabını kendi isteği gibi kullanmalarına olanak tanır.

XSS'i Açıklayalım:
Düşün ki bir parkta çok güzel bir oyun oynuyorsun ama birisi gelip oyununu bozacak kötü bir şey koyuyor.

- **XSS,** kötü niyetli birinin oyunun içine kötü bir şey ekleyip oyunu bozması gibidir.
- Bu kötü şey, oyuncuların oyun sırasında zarar görmesine sebep olabilir, tıpkı kişisel verilerin çalınmasına veya sayfanın değişmesine neden olabilmesi gibi.

Sonuç olarak, CORS farklı kaynaklardan veri almanın güvenli yollarını sağlar, CSRF kötü niyetli kişilerin başkalarının hesaplarını çalmasını engeller, ve XSS kötü kodların sayfalara zarar vermesini engeller!

Web Güvenliği ve Performans Optimizasyonu

HTTPS, SSL/TLS Nedir?

1. HTTPS (HyperText Transfer Protocol Secure)

HTTPS, web üzerindeki güvenli iletişim için kullanılan bir protokoldür. HTTP'nin güvenli versiyonudur. HTTPS, web sitesi ile tarayıcı arasındaki verilerin şifrelenmesini sağlar. Bu sayede, kötü niyetli kişiler, internet üzerindeki verileri görüp değiştiremez.

- **Neden kullanılır?** Bilgilerin gizliliğini korumak için ve web sitenizin güvenli olduğunu göstermek için kullanılır.

2. SSL/TLS (Secure Sockets Layer / Transport Layer Security)

SSL ve TLS, internet üzerindeki veri iletişimini şifreleyerek güvenliği sağlayan protokollerdir. SSL, daha eski bir protokolken, TLS günümüzde daha güvenli ve yaygın kullanılan versiyonudur. SSL/TLS, verilerin internet üzerinden güvenli bir şekilde iletilmesini sağlamak için kullanılır.

- **Neden kullanılır?** Web siteleriyle yapılan iletişimin, üçüncü şahıslar tarafından izlenememesi veya değiştirilememesi için kullanılır.

✨Hayal Ederek Anlama

HTTPS'i Açıklayalım:

Diyelim ki bir postaneye bir mektup gönderiyorsun.

- Eğer mektubu açık bir şekilde gönderirsen, herkes mektubu okuyabilir. Ancak, eğer mektubu şifreli bir zarfla gönderirsen, sadece alıcı mektubu okuyabilir.
- **HTTPS,** işte bu şifreli zarf gibidir. Web sitenle gönderdiğin bilgilerin şifrelenmesini sağlar ve sadece doğru kişilerin okumasına izin verir.

SSL/TLS'i Açıklayalım:

Şimdi, düşün ki bir gizli bilgi (mesela bir şifre) paylaşman gerekiyor.

- **SSL/TLS,** gizli bilgini güvenli bir şekilde biriyle paylaşmanı sağlayan özel bir güvenlik kapısı gibidir.
- Bu kapı, bilgilerin yolda kaybolmasını veya çalınmasını engeller. Yani, bir mesajı bir yerden başka bir yere gönderirken kimse araya girip onu değiştiremez ya da okuyamaz.

Sonuç olarak, HTTPS, web siteleriyle olan iletişimi güvenli hale getirirken, SSL/TLS de bu iletişimdeki verilerin şifrelenmesini sağlar, böylece gizliliğiniz korunur!

Web Güvenliği ve Performans Optimizasyonu

API Güvenliği Nedir?

API Güvenliği, web uygulamaları ve servisler arasındaki veri alışverişini güvenli hale getirmek için alınan önlemler ve uygulanan yöntemlerdir. API'ler (Application Programming Interfaces), farklı yazılımların birbiriyle iletişim kurmasını sağlar. API güvenliği, kimlik doğrulama, yetkilendirme, veri şifreleme gibi önlemlerle verilerin güvenliğini sağlamak amacıyla kullanılır.

API Güvenliği Amaçları:

- **Kimlik Doğrulama:** Kullanıcıların ve servislerin kimliklerinin doğru şekilde doğrulanması.
- **Yetkilendirme:** Kullanıcıların sadece yetkili oldukları verilere ve kaynaklara erişmesini sağlamak.
- **Veri Şifreleme:** Verilerin kötü niyetli kişilerin eline geçmesini engellemek için şifrelenmesi.
- **Rate Limiting (İstek Sınırlandırma):** API'nin kötüye kullanılmasını engellemek için yapılabilecek isteklerin sayısını sınırlamak.

✨Hayal Ederek Anlama

API Güvenliğini Açıklayalım:

Diyelim ki bir gizli hazine haritası var ve sadece doğru kişi bu haritayı görebiliyor.

- **API güvenliği,** bu haritayı gizli tutmak ve sadece hak edene göstermek için alınan önlemler gibidir. Yani, harita sadece belirli kişilere veya belirli servislere gösterilebilir.
  a. **Kimlik Doğrulama:**
     - Bu, haritayı almak isteyen kişinin kim olduğunu anlamak gibidir. Örneğin, harita sadece seni tanıyorsa sana gösterilir.
  b. **Yetkilendirme:**
     - Bu, haritayı alacak kişinin onu görme hakkı olup olmadığını kontrol etmektir. Örneğin, sadece hazinenin sahibi haritayı görebilir.
  c. **Veri Şifreleme:**
     - Harita, gizli bir kutuya konmuş ve sadece özel anahtarla açılabilir. Yani harita yolda giderken kimse tarafından görülmez ya da çalınamaz.
  d. **Rate Limiting (İstek Sınırlandırma):**
     - Birisi haritayı çok fazla istemeye çalışıyorsa, bu kişi engellenebilir. Yani, haritayı isteyen kişi daha fazla izin almaz ve kötüye kullanım engellenir.

Sonuç olarak, API güvenliği, verilerinizi koruyarak sadece yetkili kişilerin erişmesini sağlar, böylece gizliliğiniz ve güvenliğiniz korunur!

Web Güvenliği ve Performans Optimizasyonu

RegExp ve Güvenlik Nedir?

1. RegExp (Regular Expressions - Düzenli İfadeler)

RegExp, metin içerisindeki belirli desenleri bulmak, değiştirmek veya doğrulamak için kullanılan bir kural setidir. Yani, düzenli ifadeler, bir metni belirli kurallara göre analiz etmeyi sağlayan araçlardır. Örneğin, bir e-posta adresini doğrulamak, bir telefon numarasını bulmak veya belirli kelimeleri değiştirmek için kullanılır.

- **Neden kullanılır?** Metin üzerinde hızlıca işlem yapmak ve belirli kurallara göre verileri işlemek için kullanılır.

2. Güvenlik ve RegExp

RegExp güvenlik için önemlidir çünkü yanlış kullanıldığında güvenlik açıklarına yol açabilir. Örneğin, kötü niyetli bir kullanıcı, bir web uygulamasına zararlı kodlar veya komutlar ekleyebilir. Eğer bu kodlar doğru şekilde kontrol edilmezse, güvenlik açığı oluşabilir.

RegExp güvenliği sağlamak için:

- **Doğru doğrulama:** Kullanıcıdan alınan verilerin doğru formatta olduğundan emin olunmalıdır (örneğin, e-posta veya telefon numarası).
- **Güvenli kullanım:** Kötü niyetli kodlar ve istekler (örneğin, XSS, SQL injection) engellenmelidir.
- **Özelleştirilmiş Desenler:** Kötüye kullanım için tasarlanmış bazı düzenli ifadeler kullanılmamalıdır.

✨Hayal Ederek Anlama

RegExp'i Açıklayalım:

Diyelim ki bir şifre listesi var ve biz güçlü şifreleri seçmek istiyoruz.

- **RegExp**, tıpkı bir şifre avcısı gibi çalışır. Şifreyi kontrol eder ve güçlü olup olmadığını belirler. Örneğin, bir şifrede büyük harf, küçük harf, rakam ve özel karakter bulunmasını kontrol eder.
- **RegExp,** yazdığın kurallar doğrultusunda metinleri hızlıca tarar ve istenmeyenleri bulup temizler.

RegExp ve Güvenliği Açıklayalım:

Diyelim ki bir evdeki güvenlik kapısını kontrol ediyorsun.

- **RegExp,** evin kapılarını denetler ve sadece güvenli ve doğru şekilde açılanları geçirebilir. Yani, kapıyı açan kişinin güvenli olup olmadığını kontrol eder.

Ama eğer kapı yanlış kontrol edilirse ve zararlı bir kişi güvenlik sistemini atlatırsa, o kişi evin içine girebilir ve zarar verebilir. İşte bu yüzden, **RegExp güvenliği** çok önemlidir!

- Kötü niyetli kişiler, yanlış kullanılan bir **RegExp** ile güvenlik duvarlarını aşabilirler. Bu yüzden, doğru ve güvenli kurallar ile metinler doğrulanmalı ve her şey kontrol edilmelidir.

Sonuç olarak, RegExp, metinler üzerinde işlem yapmanın güçlü bir aracı olsa da, güvenlik açısından doğru ve dikkatli kullanılması gerekmektedir. Yanlış kullanımı, güvenlik açıklarına yol açabilir!

Web Güvenliği ve Performans Optimizasyonu

Performans İyileştirme (Lazy Loading, Minification) Nedir?

1. Lazy Loading (Tembel Yükleme)

Lazy loading, bir web sayfasında veya uygulamada, gerekli olmayan verilerin sadece gerektiğinde yüklenmesini sağlayan bir tekniktir. Yani, tüm içerik bir anda yüklenmek yerine, kullanıcı sayfayı kaydırdıkça veya etkileşime girdikçe sadece gerekli olan öğeler yüklenir. Bu sayede sayfa daha hızlı açılır.

- **Neden kullanılır?** Web sayfasının ilk açılış hızını artırmak ve gereksiz veri yüklemesini engellemek için kullanılır.

2. Minification (Minifikasyon)

Minification, web sitelerinde kullanılan JavaScript, CSS ve HTML dosyalarının boyutunu küçültmek için yapılan işlemdir. Bu işlemde, dosyalarda gereksiz boşluklar, yeni satırlar ve yorum satırları kaldırılır. Bu sayede, dosyaların boyutu küçülür ve sayfa daha hızlı yüklenir.

- **Neden kullanılır?** Sayfa yüklenme hızını artırmak ve sunucudan daha hızlı veri iletmek için kullanılır.

✨Hayal Ederek Anlama

Lazy Loading'i Açıklayalım:

Diyelim ki bir kütüphaneye gidiyorsun ve orada binlerce kitap var.

- **Lazy loading,** tüm kitapları ilk anda değil, sadece okuduğun sayfanın kitaplarını getirmek gibidir. Yani, sadece bakmak istediğin kitaplar getirilir, diğerleri gereksiz yere getirilmez.
- Bu, kütüphanede dolaşmanı daha hızlı hale getirir, çünkü sadece gereken kitaplar gelir, diğerleri ise istek üzerine yüklenir.

Minification'ı Açıklayalım:

Düşün ki bir kargo kutusu var ve içinde bazı eşyaların yer alması gerekiyor.

- Eğer kutuda gereksiz boşluklar varsa, bu kutunun büyük olmasına sebep olur ve daha uzun sürede teslim edilir.
- **Minification,** kutunun içindeki gereksiz boşlukları kaldırır, böylece kutunun boyutunu küçültür ve daha hızlı teslimat yapılmasını sağlar.

Sonuç olarak:

- **Lazy loading,** gereksiz öğeleri yüklememek ve sayfayı hızlı açmak için kullanılır.
- Minification, dosyaların boyutunu küçültmek ve sayfanın daha hızlı yüklenmesini sağlamak için kullanılır.

Her iki yöntem de performans iyileştirme sağlar, böylece daha hızlı ve verimli web siteleri ve uygulamalar oluşturulabilir!

İşletim Sistemleri ve Sistem Yönetimi

İşletim Sistemi Nedir?

İşletim sistemi (OS - Operating System), bilgisayar ve diğer akıllı cihazların donanım ve yazılımlarını yöneten temel bir yazılımdır. Kullanıcıların programları çalıştırmasını, dosyaları yönetmesini ve cihazları (klavye, fare, yazıcı vb.) kullanmasını sağlar. En yaygın işletim sistemleri Windows, macOS, Linux ve mobil cihazlarda Android ile iOS'tur.

İşletim Sistemlerinin Farkları Nelerdir?

Farklı işletim sistemleri, kullanım alanlarına ve kullanıcı ihtiyaçlarına göre farklılık gösterir:

- **Windows:** En yaygın kullanılan işletim sistemlerinden biridir, özellikle ofis ve oyunlar için tercih edilir.
- **macOS:** Apple'ın işletim sistemidir, genellikle tasarımcılar ve profesyoneller için optimize edilmiştir.
- **Linux:** Açık kaynaklıdır, geliştiriciler ve sunucu sistemleri için idealdir.
- **Android & iOS:** Mobil cihazlar için kullanılan işletim sistemleridir. Android daha özelleştirilebilirken, iOS daha kapalı ve stabil çalışır.

✨Hayal Ederek Anlama

İşletim sistemi, bilgisayarın beynidir! Bilgisayarı bir insan gibi düşünelim:

- **Windows, macOS ve Linux,** beynin farklı çalışma tarzlarıdır. Bazıları oyun oynamayı sever (Windows), bazıları sanat yapmayı sever (macOS), bazıları ise bilim adamı gibi araştırma yapmayı sever (Linux).
- **Android ve iOS,** telefonların beynidir. Android, oyuncaklarını istediğin gibi düzenleyebileceğin bir oda gibiyken, iOS her şeyin belirli bir yerde olduğu düzenli bir oda gibidir.

Yani işletim sistemleri, cihazlarımızın nasıl çalıştığını belirler ve onlarla en iyi şekilde iletişim kurmamızı sağlar!



İşletim Sistemleri ve Sistem Yönetimi

UNIX Nedir?

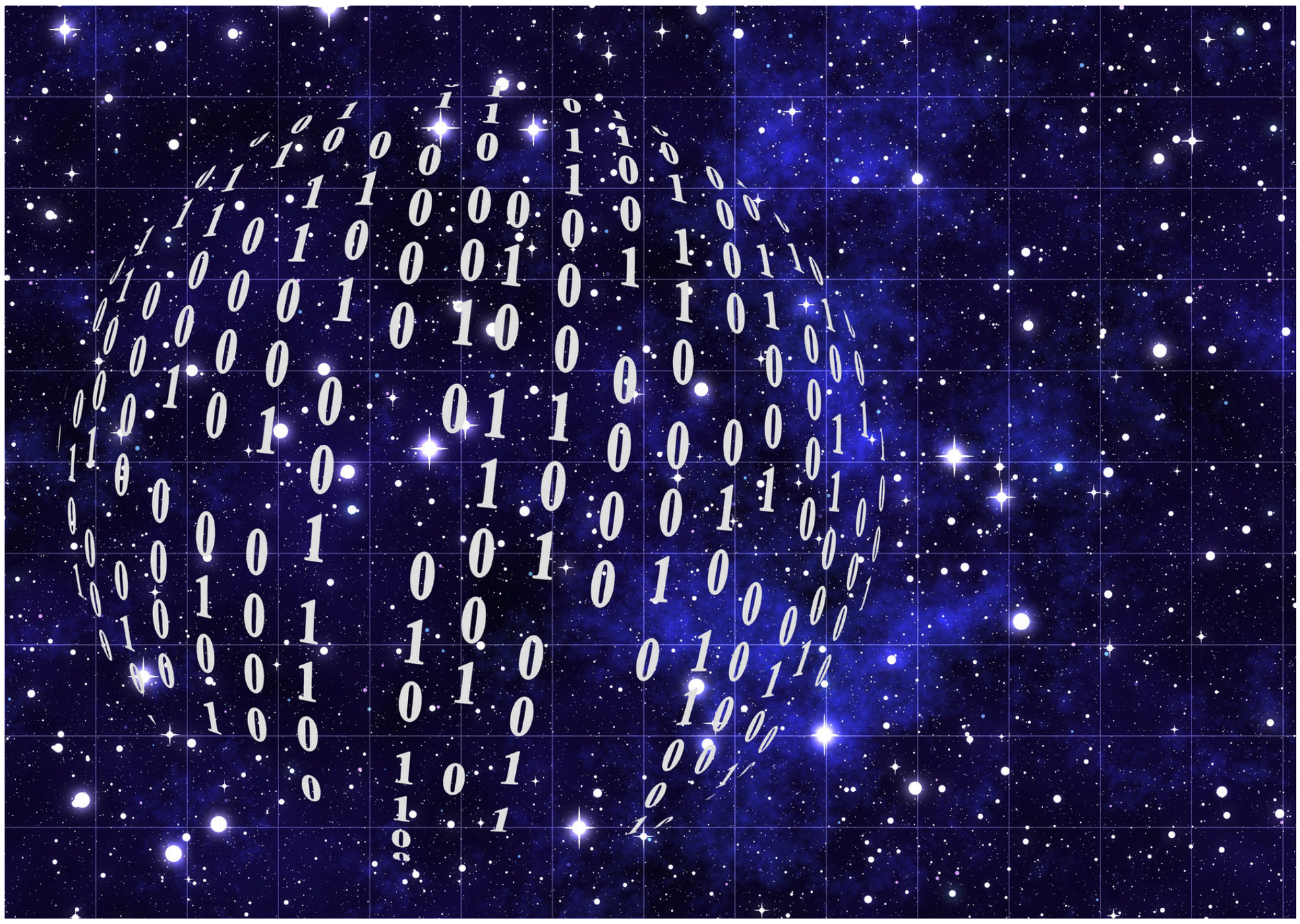
UNIX, çok kullanıcılı ve çok görevli (multi-user, multi-tasking) bir işletim sistemidir. 1969 yılında AT&T Bell Labs'te geliştirilmiş ve günümüzde birçok modern işletim sistemine (Linux, macOS gibi) temel olmuştur. Güçlü, güvenli ve esnek yapısıyla özellikle sunucular, büyük bilgisayar sistemleri ve geliştirme ortamlarında tercih edilir.

✨Hayal Ederek Anlama

Düşün ki büyük bir oyuncak dükkanın var ve birçok çalışan aynı anda oyuncakları düzenlemekle meşgul. UNIX, bu dükkanın düzenini sağlayan bir süper kahraman gibi çalışır!

- Her çalışana (kullanıcıya) işleri paylaşır.
- Aynı anda birçok işi yapabilir (oyuncakları yerleştirmek, sipariş almak, temizlik yapmak).
- Her şeyin düzgün çalıştığından emin olur.

UNIX, bilgisayarların arka planda süper güçlü ve düzenli çalışmasını sağlayan gizli kahramanlardan biridir!



İşletim Sistemleri ve Sistem Yönetimi

GNU/Linux Nedir?

GNU/Linux, açık kaynaklı ve özgür bir işletim sistemidir. Linux çekirdeği ve GNU araçları bir araya gelerek oluşturulmuştur. İlk olarak 1991'de Linus Torvalds tarafından geliştirilen Linux çekirdeği, özgür yazılım hareketi sayesinde yaygınlaşmıştır. Günümüzde Ubuntu, Debian, Fedora gibi birçok farklı Linux dağıtımı bulunmaktadır. Güvenli, esnek ve özelleştirilebilir yapısıyla sunucularda, bilgisayarlarda ve gömülü sistemlerde yaygın olarak kullanılır.

✨Hayal Ederek Anlama

Düşün ki bir lego seti alıyorsun ama bu setle istediğin her şeyi yapabilirsin! İşte GNU/Linux, tıpkı böyle bir oyun alanı gibidir.

- Windows ve macOS gibi hazır oyuncaklar yerine, GNU/Linux sana parçaları verir ve istediğin gibi bir şeyler inşa edebilirsin.
- Açık kaynaklı olduğu için herkes bu oyuna katılıp yeni parçalar ekleyebilir.
- Bilgisayarını tamamen kişiselleştirebilir, hızlandırabilir ve özgürce kullanabilirsin.

Yani, GNU/Linux bilgisayar dünyasının lego seti gibidir—kurallar senin elinde!



İşletim Sistemleri ve Sistem Yönetimi

Terminal Komutları Nedir?

Terminal komutları, işletim sistemleriyle yazılı olarak etkileşim kurmamızı sağlayan komutlardır. Özellikle Linux, macOS ve bazı Windows sistemlerinde kullanılan terminal (komut satırı), bilgisayara çeşitli görevleri yaptırmak için metin tabanlı bir arayüz sunar. Bu komutlarla dosya yönetimi, program çalıştırma, sistem ayarlarını değiştirme gibi işlemler yapılabilir.

Temel Terminal Komutları

- **ls** → Bulunduğun klasördeki dosyaları listeler.
- **cd [klasör_adı]** → Başka bir klasöre geçiş yapar.
- **pwd** → Hangi klasörde olduğunu gösterir.
- **mkdir [klasör_adı]** → Yeni bir klasör oluşturur.
- **rm [dosya_adı]** → Bir dosyayı siler.
- **cp [kaynak] [hedef]** → Bir dosyayı kopyalar.
- **mv [kaynak] [hedef]** → Bir dosyayı taşır veya adını değiştirir.
- **touch [dosya_adı]** → Yeni bir dosya oluşturur.
- **cat [dosya_adı]** → Dosyanın içeriğini gösterir.
- **echo "metin" >** dosya.txt → Bir dosyanın içine yazı ekler.
- **clear** → Terminal ekranını temizler.



İşletim Sistemleri ve Sistem Yönetimi

Localhost & Sunucu Tarafı Kodlama Nedir?

1. Localhost

Localhost, bir bilgisayarın kendi kendine hizmet veren yerel sunucusudur. Kendi bilgisayarınızda çalışan bir web uygulamasını test etmek ve geliştirmek için kullanılır. Yani, internet üzerinde olmayan ancak sadece kendi bilgisayarınızda erişilebilen bir sunucudur.

- **Neden kullanılır?** Web siteleri veya uygulamalar geliştirilirken, canlı bir sunucuya göndermeden önce yerel ortamda denemeler yapmak için kullanılır.

2. Sunucu Tarafı Kodlama (Server-Side Scripting)

Sunucu tarafı kodlama, web uygulamalarının ve sitelerinin veritabanlarıyla etkileşime girdiği, kullanıcı taleplerini işlediği ve web sayfalarını oluşturduğu kodların sunucuda çalıştığı bir yaklaşımdır. Bu tür kodlar, kullanıcıların tarayıcılarına gönderilmeden önce sunucuda işlenir.

- **Neden kullanılır?** Veritabanı işlemleri, kullanıcı doğrulaması, veri işleme gibi işlemleri sunucuda güvenli bir şekilde yapmak için kullanılır.

✨Hayal Ederek Anlama

Localhost'u Açıklayalım:

Diyelim ki bir çizim yapıyorsun ama bunu hızlıca görmek istiyorsun.

- **Localhost,** kendi odanda yapılan çizimi anında görmek gibidir. Çizimin sadece senin bilgisayarında göründüğü ve kimseye gösterilmediği bir ortamda çalışırsın.
- **Bu sayede,** yaptığın her değişikliği direkt olarak görebilir ve hızlıca test edebilirsin. Bu, bir web sitesini geliştirdiğinde başka birine gösterilmeden önce sadece senin bilgisayarında çalışan bir test alanıdır.

Sunucu Tarafı Kodlamayı Açıklayalım:

Diyelim ki bir restoranın garsonu sipariş alıyor ve siparişi mutfakta şefin hazırlaması gerekiyor.

- **Sunucu tarafı kodlama,** siparişi garson alır, ancak yemek mutfakta şef tarafından pişirilir. Garson sadece siparişi alır, şef de mutfakta yemekleri yapar ve sonra garson yemekleri masaya getirir.
- **Burada şef (sunucu)** siparişi işler, veritabanından yemek tariflerini alır ve sonrasında yemek hazır olduğunda garson (tarayıcıya) yemekleri getirir.
- **Yani,** veri işleme, doğrulama ve yemek pişirme işlemleri sunucuda gerçekleşir.

Sonuç olarak, localhost, kendi bilgisayarında uygulamaları test etmeni sağlarken, sunucu tarafı kodlama, verilerin işlenmesini ve güvenli işlemleri sunucuda yapmanı sağlar. Bu ikisi de uygulama geliştirmede hızlı, verimli ve güvenli işlem yapmayı sağlar!

İşletim Sistemleri ve Sistem Yönetimi

SSH Nedir?

SSH (Secure Shell), bir bilgisayarın veya sunucunun uzaktan güvenli bir şekilde yönetilmesini sağlayan bir protokoldür. SSH, verilerin şifreli bir şekilde gönderilmesini ve alınmasını sağlar, böylece internet üzerinden yapılan işlemler gizli ve güvenli kalır. Genellikle, bir sunucuya bağlanmak ve komutlar çalıştırmak için kullanılır.

- **Neden kullanılır?** Sunuculara ve uzak sistemlere güvenli bir şekilde erişmek, dosyaları aktarabilmek ve komutları çalıştırabilmek için kullanılır.

✨Hayal Ederek Anlama

SSH'yi Açıklayalım:
Diyelim ki bir gizli odam var ve bu odaya sadece belirli kişilerin girmesine izin veriyorsun.

- **SSH,** gizli odanın kapısı gibidir. Yalnızca doğru anahtara sahip kişiler odaya girebilir.
- Herhangi biri odaya girmeye çalışsa bile, anahtarı doğru olmazsa kapı açılmaz. Bu sayede, odada ne olduğunu kimse göremez ve sadece güvenli kişilerin girişi sağlanır.

SSH’nin Güvenliği:

- **Kapı (sunucu**) sadece güvenli bir şekilde açılır. Eğer sen uzaktan odaya girmek istiyorsan, doğru anahtarı kullanman gerekir. Bu, odaya girmeyi isteyen herkesin yalnızca şifreli ve güvenli bir bağlantı ile girmesini sağlar.
- **Yani**, SSH ile internet üzerinden çok uzakta olsan bile, gizli odadaki (sunucudaki) dosyalarına erişebilirsin, ancak güvenli bir yol izleyerek!

Sonuç olarak, SSH, uzak bir bilgisayara veya sunucuya güvenli bir şekilde bağlanmanı ve işlemler yapmanı sağlayan bir güvenlik aracıdır. Verilerinizi korur ve sadece yetkili kişilerin bağlanmasını sağlar!

Domain, Hosting ve DNS Yönetimi

Domain Alımı Nedir?

Domain alımı, bir web sitesinin adresinin (domain) satın alınması işlemidir. Web sitesi adresi, internet üzerinde bir siteyi tanımlayan özel bir isme verilen addır. Domain, genellikle "www.siteadi.com" şeklinde görünür ve kullanıcıların bir siteye kolayca erişebilmesini sağlar. Domain adı satın alındığında, bir yıl boyunca o adresin sahibi olursunuz ve bu adresi kullanarak kendi web sitenizi kurabilirsiniz.

- **Neden kullanılır?** Web siteleri oluştururken, internet kullanıcılarının kolayca ulaşabileceği bir adres (domain) sahibi olmak için kullanılır.

✨Hayal Ederek Anlama

Domain Alımını Açıklayalım:
Diyelim ki bir evin kapısına bir numara koyman gerekiyor.

- Bu numara, senin evinin adresi olur. O evin kapısına bakarak, kimse seni ve evini bulabilir. Bu evin adı gibi, domain de bir web sitesinin adı olur.
- **Domain alımı,** internette kendine ait bir adres almak gibidir. Yani, web siten için bir ev numarası satın alırsın, bu numara ile insanlar senin web sitene girebilir.

Domain Alımının Kullanımı:

- Eğer www.mutluyum.com adlı bir domaini satın alırsan, artık bu adı kullanarak insanlar mutluyum.com adresine girebilirler.
- Web siten "mutluyum.com" adresiyle tanınır ve sen de o adı kullanmaya başlarsın. Eğer başka biri aynı adresi almak isterse, senin iznin olmadan bu adı alamaz.

Sonuç olarak, domain alımı, bir web sitesine ait benzersiz bir adres almak demektir. Bu adres, sitenin kolayca bulunmasını sağlar ve sana ait olur!



Domain, Hosting ve DNS Yönetimi

Domain Yönlendirme & NS Nedir?

1. Domain Yönlendirme (Domain Forwarding)

Domain yönlendirme, bir domain adresini (web sitesi adresini), başka bir web sitesi adresine otomatik olarak yönlendirme işlemidir. Yani, bir kullanıcı belirli bir domaini yazdığında, sistem onu başka bir domain veya web sitesi adresine yönlendirir. Örneğin, www.eskiadres.com adresine girdiğinde, seni www.yenadres.com adresine yönlendiren bir sistem, domain yönlendirme kullanır.

- **Neden kullanılır?** Birçok farklı domain adresi üzerinden aynı web sitesine ulaşmak, eski adresi yenisiyle değiştirmek veya kullanıcıları doğru adrese yönlendirmek için kullanılır.

2. NS (Name Server) Nedir?

NS (Name Server), domain adlarının hangi sunucularda barındığını belirleyen ve yönlendiren bir sistemdir. Yani, bir domainin hangi web sunucusuna ait olduğunu söyleyen bir "adres rehberi" gibi çalışır. DNS (Domain Name System) üzerinden yönlendirme yaparak, internet kullanıcılarının doğru web sitesine ulaşmasını sağlar.

- **Neden kullanılır?** Domain adlarını doğru bir şekilde sunuculara yönlendirmek ve kullanıcıların doğru web sitesine gitmesini sağlamak için kullanılır.

✨Hayal Ederek Anlama

Domain Yönlendirmesini Açıklayalım:

Diyelim ki bir yolculuk yapıyorsun ve bir yol levhası sana bir yere gitmeni söylüyor.

- Ancak, yol levhası seni, gitmen gereken farklı bir yola yönlendiriyor. Yani, senin gittiğin yer aslında başka bir yol üzerinde.
- Domain yönlendirmesi, bir adresin seni başka bir adrese yönlendirilmesi gibidir. Eğer birisi eski adresine gitse bile, yeni adresine doğru otomatik olarak yönlendirilir.

NS'yi Açıklayalım:

Diyelim ki bir telefon rehberin var ve bu rehberde her kişinin numarası yazılı.

- **NS (Name Server),** telefon rehberi gibi çalışır. Eğer birisi seni aramak isterse, rehberdeki numaraya göre doğru kişiye ulaşır.
- NS, domain adresinin hangi sunucularda olduğunu söyler ve böylece kullanıcılar doğru web sitesine ulaşır.

Sonuç olarak, domain yönlendirmesi, bir web sitesinin adresinin başka bir yere gitmesini sağlarken, NS ise domain adresinin doğru sunucularda barındığını ve kullanıcıların doğru siteye gitmesini sağlar!

Domain, Hosting ve DNS Yönetimi

Hosting Türleri (Paylaşımlı, VPS, Cloud) Nedir?

1. Paylaşımlı Hosting (Shared Hosting)

Paylaşımlı hosting, bir web sitesinin bir sunucuyu diğer birçok siteyle paylaşarak barındırıldığı bir hosting türüdür. Yani, sunucunun kaynakları (işlemci gücü, bellek, depolama vb.) birden fazla site tarafından ortaklaşa kullanılır.

- **Neden kullanılır?** Genellikle küçük ve orta büyüklükteki web siteleri için, daha düşük maliyetli ve kolay bir çözüm olarak kullanılır.

2. VPS (Virtual Private Server)

VPS, bir fiziksel sunucunun bölünerek her bir bölüme ayrı bir sunucuymuş gibi çalışma imkanı sağlar. Her kullanıcıya özelleştirilmiş kaynaklar (işlemci, RAM, depolama vb.) tahsis edilir. Yani, her kullanıcı özel bir alan kullanır ama yine de tek bir sunucu üzerinde barındırılır.

- **Neden kullanılır?** Daha fazla özelleştirme, kontrol ve performans gereksinimleri olan siteler için tercih edilir. Paylaşımlı hostingden daha güçlüdür.

3. Cloud Hosting (Bulut Hosting)

Cloud hosting, sitenizin birden fazla sunucu üzerinde barındırılmasını sağlayan bir hosting türüdür. Yani, site verileri ve kaynakları bulut sistemlerine dağıtılır ve gerektiğinde daha fazla sunucu eklenebilir.

- **Neden kullanılır?** Yüksek esneklik, ölçeklenebilirlik ve performans gereksinimleri olan projeler için tercih edilir. Web sitesinin daha fazla trafik alması durumunda, sistem çok kolay bir şekilde genişletilebilir.



Domain, Hosting ve DNS Yönetimi

✨Hayal Ederek Anlama

Paylaşımlı Hosting'i Açıklayalım:
Diyelim ki bir apartmanda yaşıyorsun ve aynı daireyi birçok kişiyle paylaşıyorsun.

- Bu durumda, herkesin ortak kullandığı bir yer var. Evdeki odalar (sunucu kaynakları) herkes tarafından kullanılıyor.
- Paylaşımlı hosting, tıpkı bir apartmanda birçok kişiyle aynı evi paylaşmak gibidir. Bütçe dostu olsa da, herkesin ihtiyaçları birbirini etkileyebilir.

VPS'i Açıklayalım:
Diyelim ki bir apartmanda kendi dairen var, ama dairenin dışarıda diğer dairelerden bağımsız bir alanı var.

- Bu durumda, kendi dairenin (VPS) kendi odalarını ve kaynaklarını bağımsız bir şekilde kullanabilir. Diğer dairelerden bağımsız bir alanın vardır.
- VPS, tıpkı kendi özel dairenin olması gibidir. Bağımsız bir alanda olursun ve daha fazla kontrol ve güç sahibi olursun.

Cloud Hosting'i Açıklayalım:
Diyelim ki bir şirketin çok büyük bir depoya sahipsin ve bu depo farklı noktalarda bulunan birkaç odadan oluşuyor.

- Her odada farklı malzemeler bulunuyor ve her odadaki malzeme gerektiğinde başka bir oda ile kolayca değiştirilebilir.
- Cloud hosting, tıpkı çok büyük bir depoya sahip olmak gibidir. İhtiyaç duyduğun kadar alan alırsın ve gelişen ihtiyaçlarına göre bu alanı kolayca genişletebilirsin.

Sonuç olarak, paylaşımlı hosting, küçük ve bütçe dostu siteler için uygunken, VPS daha gelişmiş kontrol isteyenler için, cloud hosting ise büyüyen ve yüksek esneklik gerektiren projeler için idealdir!



Domain, Hosting ve DNS Yönetimi

Sunucu Seçimi Nedir?

Sunucu seçimi, bir web sitesi ya da uygulama için en uygun sunucu tipinin belirlenmesidir. Web sitenizi, veritabanınızı ya da uygulamanızı barındıracak sunucu türünü, ihtiyaçlarınıza göre performans, güvenlik ve maliyet gibi faktörlere göre seçmek önemlidir. Sunucu, verilerinizi saklayan ve internet üzerinden erişilebilir kılan bir bilgisayardır.

- **Neden kullanılır?** Web sitenizin hızını, güvenliğini ve kullanıcı deneyimini optimize etmek için doğru sunucuyu seçmek gereklidir.

✨Hayal Ederek Anlama

Sunucu Seçimini Açıklayalım:
Diyelim ki bir restoran açıyorsun ve misafirlerine yemek servisi yapacaksın.

- **Sunucu,** restoranın mutfağı gibidir. Mutfak ne kadar büyük ve güçlü olursa, yemekler (veriler) daha hızlı pişer ve misafirlerine daha hızlı ulaşır.
- Eğer küçük bir restoran açıyorsan, küçük bir mutfak seçmen yeterli olabilir. Ama büyük bir restoran açacaksan, daha büyük ve güçlü bir mutfak seçmen gerekebilir.

Sunucu Seçiminin Önemi:

- **Hızlı bir restoran (sunucu),** misafirlerine daha hızlı yemekler sunar. Eğer çok fazla kişi geliyorsa, restoranın güçlü bir mutfağa ihtiyacı vardır.
- Eğer küçük bir restoran açıyorsan, daha küçük bir mutfak yeterli olacaktır ve maliyetler de düşük olur.
- Ancak büyük bir restoran için daha büyük bir mutfak, daha fazla alan ve daha güçlü ekipmanlar gerekir. Bu da daha fazla yemek servis etmeni sağlar.

Sonuç olarak, doğru sunucu seçimi, web sitenizin hızlı çalışmasını, güvenli olmasını ve yükselen ihtiyaçlara göre büyüyebilmesini sağlar. Bu seçim, restoranındaki mutfağını seçmek gibidir, ne kadar güçlü olursa o kadar iyi hizmet verir!



Domain, Hosting ve DNS Yönetimi

cPanel Nedir?

cPanel, bir web sunucusunun yönetimini kolaylaştıran ve kullanıcı dostu bir arayüz sunan bir kontrol panelidir. Web sitesi sahiplerinin sunucularını yönetmelerini, dosya yüklemelerini, veritabanı yönetimlerini ve e-posta hesaplarını düzenlemelerini sağlar. cPanel, genellikle paylaşımlı hosting ve VPS hosting hizmetlerinde kullanılan bir araçtır.

- **Neden kullanılır?** Web sitesi yöneticilerinin, teknik bilgisi olmasa da kolayca web sitesinin ayarlarını yapabilmesini sağlar.

✨Hayal Ederek Anlama

cPanel'i Açıklayalım:
Diyelim ki bir evde yaşıyorsun ve evin içinde her şeyi düzenliyorsun.

- **cPanel,** evdeki her şeyi düzenlemek için kullandığın bir kontrol panelidir. Mesela, odaların ışığını açmak, buzdolabını düzenlemek veya kapıların kilidini değiştirmek gibi işlemleri cPanel ile yaparsın.
- Web sitesini yönetmek için de cPanel'i kullanırsın. Örneğin, web sitene yeni dosyalar yüklemek, e-posta adresi oluşturmak veya veritabanlarını düzenlemek gibi işlemleri çok kolay yapabilirsin.

cPanel'in Önemi:

- **cPanel,** her şeyin düzenli bir şekilde çalışmasını sağlar. Bir web sitesi için, cPanel tıpkı evin yönetimini kolaylaştıran bir anahtar gibidir.
- Bütün ayarları ve yönetim işleri, tek bir yerden, kolayca yapılabilir. Böylece teknik bilgin olmasa bile web sitenin yönetimini kolayca halledebilirsin.

Sonuç olarak, cPanel, web sitesi sahiplerine kolayca site yönetme imkanı sunan bir kontrol paneli gibidir. Her şeyin tek bir yerden yönetilmesini sağlar!



DevOps ve Deployment Süreçleri

Git & Versiyon Kontrolü Nedir?

Git, yazılım projelerinde yapılan değişiklikleri takip etmeye ve yönetmeye yarayan bir versiyon kontrol sistemidir. Versiyon kontrolü ise, bir dosya veya kod üzerinde yapılan değişiklikleri kaydetmeye ve gerektiğinde eski sürümlere geri dönmeye olanak sağlayan bir sistemdir.

✨Hayal Ederek Anlama

Diyelim ki bir hikâye defterin var ve her gün hikâyene yeni bölümler ekliyorsun. Ama bazen, eklediğin bir bölümü beğenmeyip geri almak istiyorsun. Eğer her değişiklikten önce hikâyenin bir kopyasını almazsan, eski haline geri dönmek zor olabilir. İşte Git tam olarak bunu yapar!

Git, senin yazdığın hikâyenin her aşamasını kaydeder. Böylece istediğin zaman eski versiyonlara dönebilir, yanlışlıkla yaptığın hataları düzeltebilir ve hatta arkadaşlarınla aynı hikâye üzerinde birlikte çalışabilirsin.



DevOps ve Deployment Süreçleri

CI/CD Nedir?

CI/CD, yazılım geliştirme sürecinde otomasyonu sağlayan iki önemli kavramdır:

- **CI (Continuous Integration – Sürekli Entegrasyon):** Geliştiricilerin yazdığı kodun düzenli olarak bir ana projeye eklenmesini ve test edilmesini sağlar.
- **CD (Continuous Deployment/Delivery – Sürekli Dağıtım/Teslimat):** Kodun testlerden geçtikten sonra otomatik olarak canlı sisteme (kullanıcılara sunulacak hale) dağıtılmasını ifade eder.

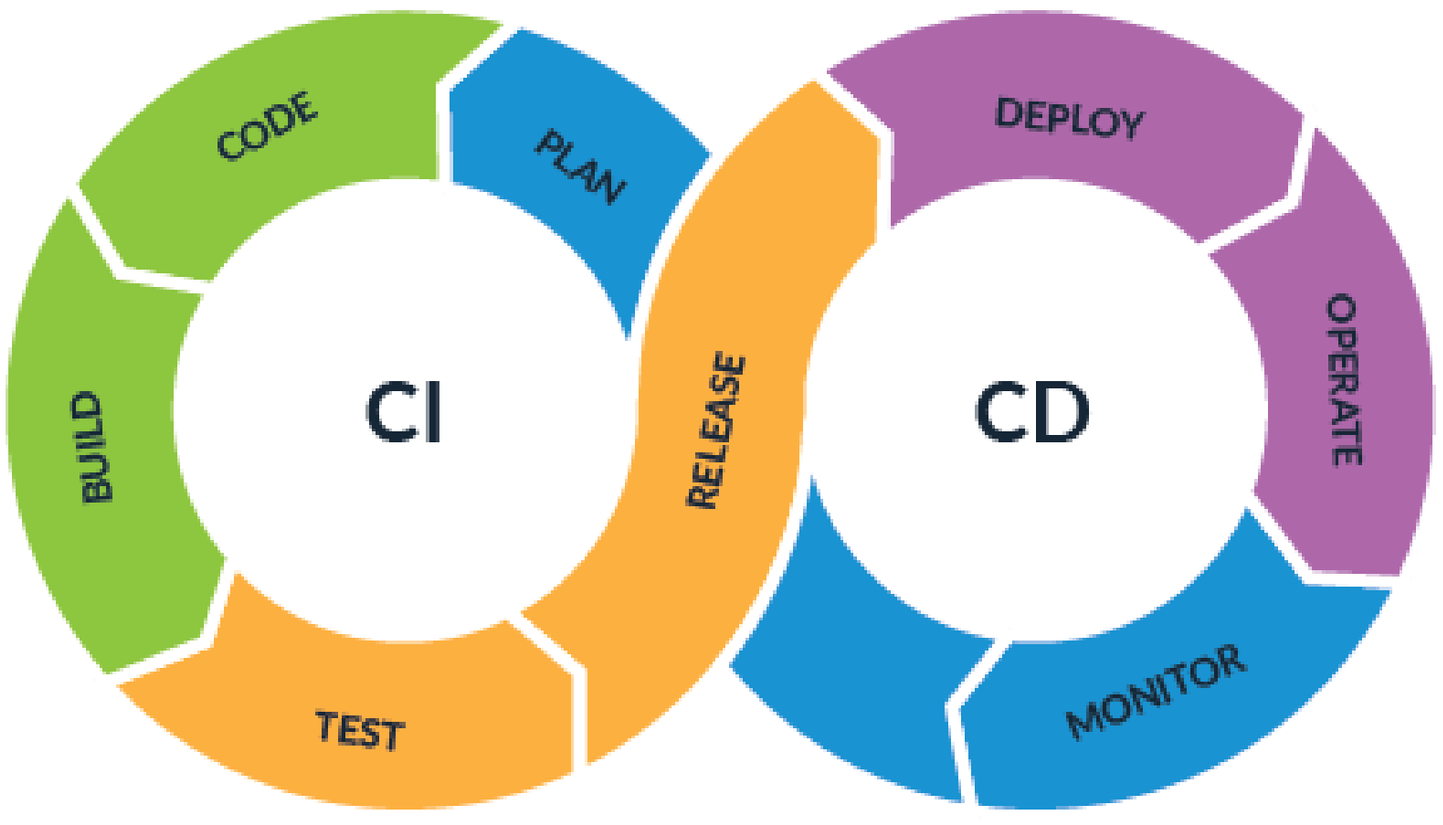
Bu süreçler sayesinde yazılım geliştirme daha hızlı, güvenilir ve hatasız hale gelir.

✨Hayal Ederek Anlama

Diyelim ki bir Lego kulesi yapıyorsun ve her gün yeni parçalar ekliyorsun. Ama yanlış bir parça eklersen kule yıkılabilir. İşte burada CI devreye giriyor! Her yeni eklediğin parçanın kuleye uygun olup olmadığını kontrol ediyor. Eğer yanlış bir parça koyarsan seni uyarıyor.

Sonra, kuleni arkadaşlarına göstermek istiyorsun ama her seferinde elinle taşımak zahmetli olur. İşte burada da CD devreye giriyor! Kule tamamlandığında, otomatik olarak vitrinde en güzel haliyle sergilenmesini sağlıyor.

Bu sayede kule yapımın hem düzenli hem de hatasız ilerliyor!



DevOps ve Deployment Süreçleri

Infrastructure as Code (IaC) Nedir?

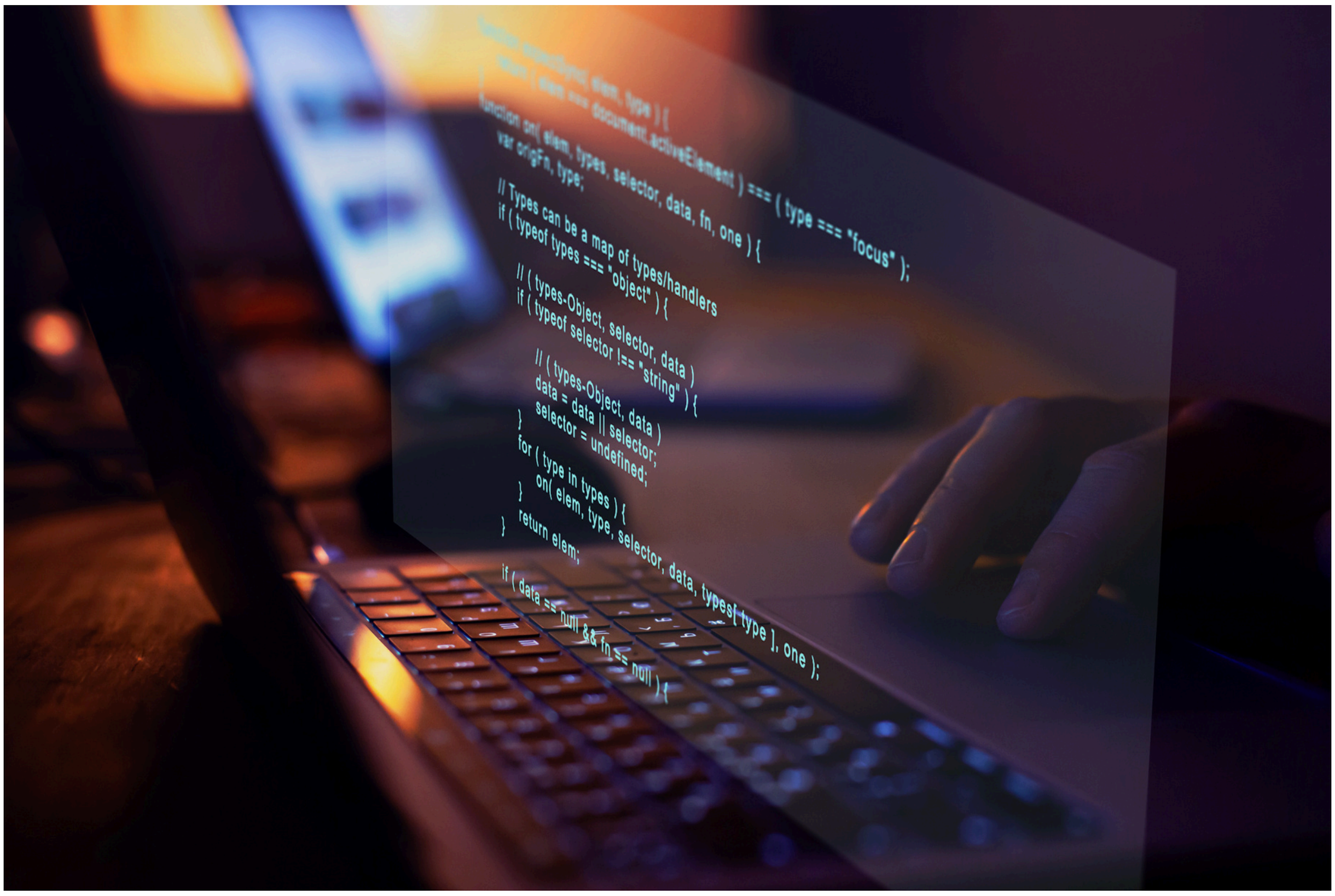
Infrastructure as Code (IaC), altyapıyı manuel olarak kurmak yerine kod yazarak otomatik olarak oluşturma ve yönetme yöntemidir. Bu sayede sunucular, ağ yapılandırmaları, veri tabanları ve diğer altyapı bileşenleri hızla ve tutarlı bir şekilde oluşturulabilir, güncellenebilir ve yönetilebilir.

IaC, altyapının bir yazılım gibi ele alınmasını sağlar, böylece tekrar eden işler otomatikleştirilir ve hatalar minimize edilir. Popüler IaC araçları arasında **Terraform, Ansible, CloudFormation** gibi çözümler bulunur.

✨Hayal Ederek Anlama

Diyelim ki bir oyuncak ev yapıyorsun. Eğer her seferinde tüm odaları, duvarları ve eşyaları tek tek elle yerleştirirsen, bu çok zaman alır ve bazen yanlış yerleştirebilirsin. Ama sihirli bir tarifin olsaydı ve her seferinde bu tarife göre oyuncak ev kendiliğinden kurulabilseydi, işler çok daha hızlı ve hatasız olurdu, değil mi?

IaC işte tam olarak bunu yapar! Bir altyapı tarifini (kodunu) yazarak sunucuların, ağların ve diğer bileşenlerin otomatik olarak kurulmasını sağlar. Böylece her seferinde aynı doğru ve güvenilir yapı oluşturulur.



DevOps ve Deployment Süreçleri

Docker ve Containerization Nedir?

Docker, uygulamaları ve bağımlılıklarını paketleyerek taşınabilir hale getiren bir platformdur.

Containerization (Konteynerleştirme) ise, yazılım ve tüm gerekli bileşenlerini bir konteyner içine alarak her ortamda (geliştirici bilgisayarı, test sunucusu, bulut vb.) aynı şekilde çalışmasını sağlayan bir teknolojidir.

Docker, yazılımın "Benim bilgisayarımda çalışıyordu ama sunucuda çalışmıyor!" gibi klasik sorunlarını ortadan kaldırır.

✨Hayal Ederek Anlama

Diyelim ki çok sevdiğin bir oyuncak setin var ve onunla her yerde oynamak istiyorsun. Ama parçaları çantan içine rastgele koyarsan, eksik veya kaybolmuş olabilir.

Şimdi düşün ki sihirli bir kutun var! Bu kutunun içine tüm oyuncaklarını ve ihtiyacın olan her şeyi düzenli bir şekilde koyuyorsun. Kutuyu nereye götürürsen götür, açtığında içindeki her şey tam ve çalışır durumda!

Docker işte tam olarak bunu yapar! Uygulamaları ve ihtiyaç duyduğu her şeyi bir "konteyner" içine koyarak, her ortamda sorunsuz çalışmasını sağlar.



DevOps ve Deployment Süreçleri

Kubernetes ve Container Orchestration Nedir?

Kubernetes, konteynerleştirilmiş uygulamaları yönetmek, dağıtmak ve ölçeklendirmek için kullanılan açık kaynaklı bir sistemdir.

Container Orchestration (Konteyner Orkestrasyonu) ise, birden fazla konteyneri otomatik olarak yönetme, yük dengeleme, hata tespiti ve onarımı gibi işlemleri yapma sürecidir.

Kubernetes, Docker gibi konteynerleri çalıştıran sistemleri organize eder, yönetir ve otomatikleştirir.

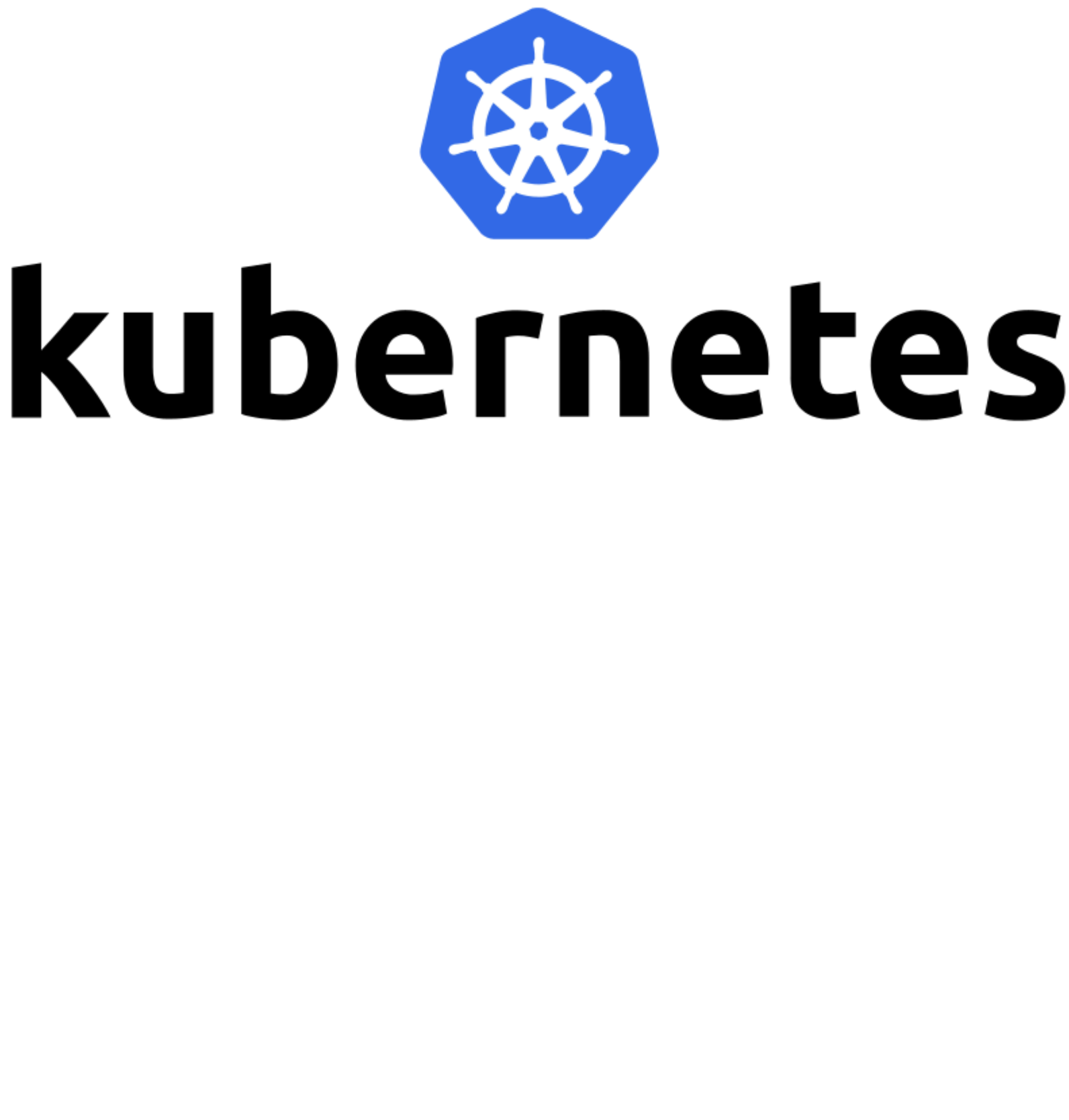
✨Hayal Ederek Anlama

Diyelim ki bir dondurma dükkânın var ve çok fazla sipariş alıyorsun. Ama tek başına her siparişi hazırlamak ve yetiştirmek çok zor! İşte burada bir dükkan yöneticisi **(Kubernetes)** devreye giriyor.

Bu yönetici:

- Dondurmaları kimin hazırlayacağını belirler (konteynerleri yönetir).
- Sipariş çok olduğunda ekstra çalışan çağırır (ölçeklendirir).
- Bozulan dondurma makinelerini fark edip onarır (hata tespiti yapar).
- Müşterilere en hızlı şekilde hizmet verilmesini sağlar (yük dengeleme yapar).

Kubernetes işte tam olarak bunu yapar! Konteynerleri yöneterek, uygulamaların her zaman sorunsuz ve verimli çalışmasını sağlar.



DevOps ve Deployment Süreçleri

Load Balancing, CDN ve Cloud Nedir?

Modern web altyapısında yüksek performans ve erişilebilirlik sağlamak için bazı temel sistemler kullanılır. Bunların başında Load Balancing (Yük Dengeleme), CDN (İçerik Dağıtım Ağı) ve Cloud (Bulut Bilişim) gelir. Gelin bu kavramları hem teknik olarak hem de hayal ederek daha iyi anlayalım.

1. Load Balancing (Yük Dengeleme)

Load Balancing, yani Yük Dengeleme, gelen web trafiğini birden fazla sunucuya eşit şekilde dağıtan bir sistemdir. Bu sayede tek bir sunucuya fazla yük binmez, sistem performansı artar ve hizmet kesintilerinin önüne geçilir.

✨Hayal Ederek Anlama

Bir pizzacıda çalıştığını düşün. Müşterilerden sipariş yağıyor ve içeride üç pizza ustası var. Eğer tüm siparişleri sadece bir ustaya verirsen, o usta çok yorulur ve siparişler gecikir. Ama siparişleri eşit şekilde üç ustaya dağıtırsan, herkes rahat çalışır ve müşteriler pizzalarını zamanında alır.

İşte Load Balancer tam da bu işi yapar: gelen istekleri sunucular arasında dengeli bir şekilde dağıtır.

2. CDN (Content Delivery Network - İçerik Dağıtım Ağı)

CDN, web sitelerinin resim, video, JavaScript, CSS gibi içeriklerini dünyanın farklı bölgelerindeki sunucularda (node) saklar. Böylece kullanıcı, kendisine en yakın sunucudan içeriği alır ve site daha hızlı yüklenir.

✨Hayal Ederek Anlama

Bir dondurma fabrikası düşün. Sadece İstanbul'da bulunuyor. Ankara'dan biri dondurma sipariş ettiğinde teslimat uzun sürer. Ama eğer Ankara, İzmir ve Trabzon'da da dondurma depoların varsa, siparişler bu şehirlerdeki en yakın depodan gönderilir.

CDN de bu mantıkla çalışır: içerikleri farklı yerlerde tutar, böylece kullanıcıya en yakından en hızlı şekilde ulaştırır.

3. Cloud (Bulut Bilişim)

Cloud, yani Bulut Bilişim, fiziksel sunucular yerine internet üzerinden sunulan sanal sunucu, veri depolama ve yazılım hizmetlerini ifade eder. Google Drive, Amazon Web Services (AWS) ve Microsoft Azure gibi platformlar buna örnektir.

✨Hayal Ederek Anlama

Oyunlarını bilgisayarında sakladığını düşün. Bilgisayarın bozulursa tüm oyunların silinir. Ama eğer oyunlarını internette, güvenli bir bulut sisteminde saklarsan, farklı bir cihazdan da oyunlarına kaldığın yerden ulaşabilirsin.

İşte Bulut Bilişim, bilgilerini güvenli bir şekilde saklar ve her yerden erişmeni sağlar.

DevOps ve Deployment Süreçleri

Monitoring ve Logging Nedir?

Monitoring (İzleme)
Bir sistemin, uygulamanın veya sunucunun performansını ve sağlığını gerçek zamanlı olarak takip etme sürecidir. CPU kullanımı, bellek tüketimi, ağ trafiği gibi kritik metrikler izlenir ve sorunlar tespit edilerek hızlı müdahale sağlanır.

Logging (Günlükleme/Kayıt Tutma)
Bir sistemin geçmişte yaptığı tüm işlemleri kayıt altına almasıdır. Hata mesajları, kullanıcı aktiviteleri ve sistem olayları gibi bilgiler log dosyalarına kaydedilir, böylece geçmişte ne olduğu incelenebilir.

✨Hayal Ederek Anlama

Monitoring (İzleme)
Diyelim ki bir oyuncak robotun var ve onun çalışıp çalışmadığını anlamak istiyorsun. Üzerinde bir gösterge paneli olsun ve bu panel sana pil seviyesi, motor sıcaklığı ve bağlantı durumu hakkında bilgi versin. Eğer pil azalıyorsa, hemen fark edip şarj edebilirsin. Monitoring işte tam olarak bunu yapar!

Logging (Günlükleme/Kayıt Tutma)
Şimdi, robotun yaptığı her hareketi bir deftere yazdığını düşün. "Saat 10:00'da yürüdü, saat 10:05'te konuştu, saat 10:10'da şarj bitti" gibi kayıtlar tutuyorsun. Eğer robot bir gün aniden durursa, bu deftere bakarak neden durduğunu anlayabilirsin. Logging işte tam olarak bunu yapar!

Monitoring anlık takip sağlar, Logging geçmişte ne olduğunu gösterir.

Paralel ve Dağıtık Sistemler

Thread ve Process Nedir?

Process (Süreç)
Bir programın çalıştırıldığında oluşturulan bağımsız bir çalışma birimidir. Her process, kendi bellek alanına ve kaynaklarına sahiptir.

Thread (İş Parçacığı)
Bir process içinde çalışan daha küçük ve bağımsız görevlerdir. Bir process içinde birden fazla thread olabilir ve bu thread'ler aynı bellek alanını paylaşarak paralel çalışabilir.

✨Hayal Ederek Anlama

Process (Süreç)
Diyelim ki bir restoran açtın. Bu restoran bir process gibi çalışıyor. Restoranın kendi mutfağı, çalışanları ve malzemeleri var. Başka bir restoran (başka bir process) tamamen farklı bir mutfağa ve çalışanlara sahip.

Thread (İş Parçacığı)
Şimdi, restoranındaki çalışanları düşünelim. Şef yemek yapıyor, garson sipariş alıyor, temizlikçi masaları siliyor. Bu çalışanlar thread'ler gibi davranıyor. Hepsi aynı restoranın (process'in) içinde çalışıyor ve aynı mutfak, masa ve malzemeleri paylaşıyor.

Process, çalışan bir programın kendisi, Thread ise o programın içinde aynı anda çalışan küçük görevlerdir.

Paralel ve Dağıtık Sistemler

Senkron ve Asenkron Programlama Nedir?

Senkron Programlama
İşlemlerin sırasıyla, biri bitmeden diğerinin başlamadığı çalışma modelidir. Her adım tamamlanana kadar beklenir.

Asenkron Programlama
İşlemlerin birbirinden bağımsız olarak, aynı anda yürütülebileceği çalışma modelidir. Bir işlem devam ederken başka bir işlem başlayabilir ve beklemeye gerek kalmaz.

✨Hayal Ederek Anlama

Senkron Programlama (Sırayla Yapma)
Diyelim ki mutfaktasın ve önce çay demleyip sonra tost yapacaksın. Senkron çalışıyorsan, önce çayın demlenmesini bekler, sonra tost yaparsın. Yani her işi tek tek sırayla yapıyorsun.

Asenkron Programlama (Aynı Anda Yapma)
Asenkron çalışıyorsan, önce çayı ocağa koyarsın ve demlenmesini beklerken tost yapmaya başlarsın. Böylece çay demlenirken boş durmaz, zaman kazanırsın!

Senkron programlama adım adım ilerlerken, asenkron programlama işleri paralel yaparak zamanı daha verimli kullanır.

Paralel ve Dağıtık Sistemler

Multithreading, Concurrency ve Parallelism Nedir?

Multithreading (Çoklu İş Parçacığı)
Bir programın içinde birden fazla thread (iş parçacığı) çalıştırarak görevleri aynı anda yürütme yöntemidir. Her thread, aynı programın farklı bölümlerini işler ve aynı bellek alanını paylaşır.

Concurrency (Eşzamanlılık)
Birden fazla işlemin birbirine geçişli (kesintili) şekilde çalıştırılmasıdır. İşlemler gerçekten aynı anda çalışmaz, ancak hızla geçiş yaparak aynı anda çalışıyormuş gibi görünür.

Parallelism (Paralellik)
Gerçekten aynı anda (paralel) çalışan işlemleri ifade eder. İşlemler farklı işlemciler veya çekirdekler üzerinde aynı anda yürütülür.

✨Hayal Ederek Anlama

Multithreading (Çoklu İş Parçacığı)
Bir tiyatro oyununda birden fazla oyuncunun sahnede aynı anda farklı diyalogları oynadığını düşün. Her oyuncu bir thread, oyun ise process gibidir. Hepsi aynı oyunun (programın) içinde birlikte çalışır.

Concurrency (Eşzamanlılık)
Diyelim ki tek bir garsonsun ve aynı anda üç masaya hizmet ediyorsun. Önce bir masadan sipariş alıp mutfağa söylüyor, sonra diğer masaya gidip sipariş alıyor, sonra ilk masanın yemeğini getiriyorsun. Yani gerçekte aynı anda yapmıyorsun ama hızla geçiş yaparak hepsini aynı anda yapıyormuş gibi görünüyorsun.

Parallelism (Paralellik)
Bu sefer restoranında üç garson var ve her biri farklı masalara aynı anda hizmet ediyor. Yani her işlem gerçekten aynı anda, bağımsız olarak yapılıyor.

Concurrency hızlı geçişler yaparak işleri aynı anda yapıyormuş gibi gösterir, Parallelism ise gerçekten aynı anda işler yürütülmesini sağlar!

Paralel ve Dağıtık Sistemler

Message Queue (Mesaj Kuyruğu) Nedir?

Message Queue (Mesaj Kuyruğu), bilgisayar sistemlerinde birden fazla işlem veya programın birbirleriyle iletişim kurabilmesi için kullanılan bir yapıdır. Temelde, mesajların sırayla işlendiği bir kuyruğa benzer şekilde çalışır. Mesajlar bu kuyruğa eklenir ve sırasıyla alınarak işlenir.

✨Hayal Ederek Anlama

Düşün ki, bir oyun oynuyorsun ve seninle birlikte oyun oynayan bir sürü arkadaşın var. Her arkadaşın sana bir şey söylemek istiyor, ama bir anda herkesin aynı anda konuşması seni karıştırır. Bu yüzden, arkadaşlarının söyledikleri sırayla bir kutuya koyulur, ve sen sırayla bu kutudaki mesajları okursun.

İşte bu kutu aslında bir Message Queue'dur. Arkadaşlarının her biri, sırayla mesajlarını bu kutuya bırakır ve sen de sırayla her birini okur ve yanıt verirsin. Bu sistem sayesinde, kimse birbiriyle karışmaz ve her şey düzenli bir şekilde olur.

Kısacası, Message Queue, bilgisayarların veya sistemlerin birbirlerine mesaj göndermesini ve bu mesajların düzenli bir şekilde sırasıyla işlenmesini sağlar.



Paralel ve Dağıtık Sistemler

Veri Tutarlılığı

Veri tutarlılığı, bir sistemin içindeki verilerin doğru ve güncel olmasını sağlamaktır. Yani, verilerin her zaman aynı ve doğru olduğundan emin olmak.

CAP Teoremi:

CAP Teoremi, dağıtık sistemlerde (yani, birden fazla bilgisayarın birlikte çalıştığı sistemlerde) 3 önemli kavramı ele alır:

- **C (Consistency):** Verilerin her zaman doğru ve güncel olduğundan emin olunması. Yani, veriyi bir yerde güncellediğinizde, her yerde aynı olması.
- **A (Availability):** Sistemin her zaman çalışır durumda olması ve veri isteklerine her zaman yanıt vermesi.
- **P (Partition Tolerance):** Eğer sistemin bir kısmı koparsa veya ayrılırsa, sistemin hala çalışması ve veri kaybı olmaması.

CAP Teoremi'ne göre, bir dağıtık sistem aynı anda bu üç özellikten yalnızca ikisini sağlayabilir. Yani, her zaman her şeyin doğru, her zaman her şeyin ulaşılabilir ve her zaman her şeyin ayrılmaması mümkün değildir. Birini seçmeniz gerekir.

Saga Pattern:

Saga Pattern, dağıtık sistemlerde uzun süren işlemler için kullanılan bir yöntemdir. Bu işlem, bir dizi küçük adım (mikro işlem) şeklinde bölünür ve her adım tamamlandığında diğerine geçilir. Eğer bir adım başarısız olursa, sistem bu hatayı düzeltmek için geri adımlar yapar. Yani, her şeyin düzgün gitmediği durumlarda, önceki adımların telafi edilmesi sağlanır.

✨Hayal Ederek Anlama

1. CAP Teoremi:

Birden fazla bilgisayarın birlikte çalıştığı bir sistem düşün. Ama bu bilgisayarların bazen birbirinden kopması, bağlantılarının kesilmesi olabilir. Şimdi senin bir bilgisayarına bir şey yazdığında, bu bilgisayarın diğer bilgisayarlara da bunu söylemesi gerekiyor. Ama bazen, her bilgisayarın her şeyin doğru ve güncel olduğundan emin olmasını ya da her zaman hemen ulaşılabilir olmasını sağlamak zor olur. CAP Teoremi işte bu dengeyi anlamaya çalışır: Sistemde her zaman doğru mu olmalı, her zaman ulaşılabilir mi olmalı, yoksa bağlantılar koptuğunda bile sistem çalışmaya devam mı etmeli?

2. Saga Pattern:

Birçok adım içeren büyük bir görevi düşün. Mesela, tatlının yapıldığı bir yemek tarifi. Tarifin her adımı tamamlanmalı, yoksa tatlı düzgün olmaz. Ama bir adım yanlış giderse, o zaman önceki adımları geri alman gerekebilir. Saga Pattern, bu tür büyük görevlerde her adımı dikkatlice ve güvenle yönetmeni sağlar. Eğer bir adım başarısız olursa, diğer adımlar da otomatik olarak geri alınır, böylece işin tamamı bozulmaz.

Kısacası, CAP Teoremi sistemlerin nasıl çalışacağına dair kurallar koyar ve Saga Pattern ise, uzun süren ve karmaşık işlerin adım adım yapılmasına yardımcı olur.

Yazılım Mimarileri ve Pratikleri

Monolitik ve Mikroservis Nedir?

Monolitik Mimari

Monolitik mimari, tüm bir uygulamanın tek bir büyük parça olarak çalıştığı bir yazılım yapısıdır. Yani, uygulamanın tüm işlevleri (veri yönetimi, kullanıcı arayüzü, işlem mantığı, vb.) aynı kod içinde yer alır ve tek bir bütün olarak çalışır.

Mikroservis Mimari

Mikroservis mimarisi ise, büyük bir uygulamanın küçük, bağımsız ve birbirinden bağımsız çalışan parçalara (mikroservislere) bölündüğü bir yapıdır. Her bir mikroservis, bir işlevi yerine getirir ve kendi veritabanına sahip olabilir. Mikroservisler birbirleriyle iletişim kurarak uygulamanın tamamını oluşturur.

✨Hayal Ederek Anlama

1. Monolitik Mimari:
Bir monolitik uygulama, büyük bir Lego kulesine benzer. Bu kuleyi yapmak için birçok farklı parça kullanabilirsin, ama hepsi tek bir büyük kuleyi oluşturur. Eğer bu kuleyi değiştirmek istersen, tüm kuleyi tekrar yapman gerekir. Yani, bir parçayı değiştirmek bütün yapıyı etkiler.

Monolitik bir sistemde, tüm işler bir arada yapılır. Bir şey yanlış giderse, bütün uygulama etkilenebilir. Ama başlangıçta basit ve kolaydır çünkü her şey tek bir yerde toplanmıştır.

2. Mikroservis Mimari:
Mikroservis ise, çok sayıda küçük Lego parçası ile yapılan bir yapıya benzer. Her parça (mikroservis), kendi başına bir iş yapar. Mesela bir parça sadece pencere, bir diğeri sadece kapı yapar. Eğer bir parçayı değiştirmek istersen, sadece o parça değişir ve diğer parçalar etkilenmez.

Mikroservis sisteminde, uygulama küçük parçalara bölünür. Her parça bağımsızdır ve kendi işini yapar. Eğer bir parça bozulursa, sadece o parça çalışmaz; diğer parçalar ise hala düzgün çalışır.

Kısacası:

- **Monolitik,** her şeyin tek bir büyük yapı içinde birleştirildiği bir sistemdir.
- **Mikroservis,** işleri küçük, bağımsız parçalara böler ve her parça kendi başına çalışır.

Her iki yöntem de farklı avantajlar ve zorluklar taşır. Monolitik, basit projelerde hızlı çalışabilir, ama büyüdükçe zorlaşabilir. Mikroservisler, büyük ve karmaşık projeler için uygundur çünkü her parça bağımsız çalışır ve kolayca güncellenebilir.

Yazılım Mimarileri ve Pratikleri

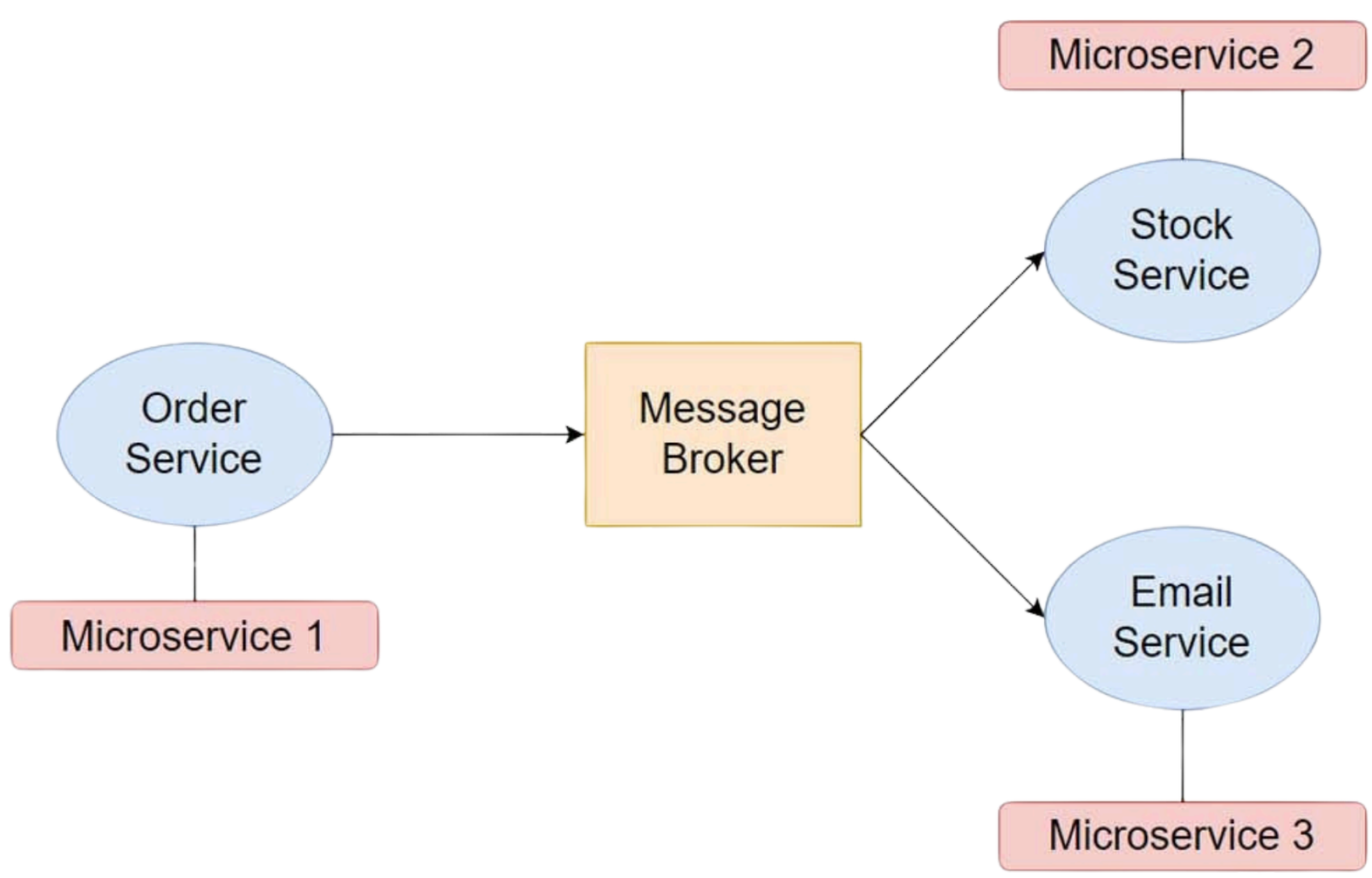
Event-Driven Architecture (EDA) Nedir?

Event-Driven Architecture (EDA), yazılım sistemlerinde farklı bileşenlerin birbirleriyle iletişim kurduğu ve tepki verdiği bir mimaridir. Bu mimaride, sistemdeki her şey bir "olay" (event) üzerinden tetiklenir. Bir olay meydana geldiğinde, sistemin ilgili bölümü bu olaya tepki verir ve işlem yapar. Olaylar genellikle veri değişiklikleri, kullanıcı eylemleri veya dış sistemlerden gelen bilgiler olabilir.

✨Hayal Ederek Anlama

Event-Driven Architecture (EDA), sistemin, bir şey olduğunda, yani bir "olay" meydana geldiğinde tepki verdiği bir yapıdır. Bu, tıpkı bir oyunda, bir karakter bir düğmeye basınca bir şeyler olup bitmesi gibi. Mesela bir oyuncu bir kapıyı açarsa, kapı açılır. Bu olay, "kapı açma" olayıdır ve sistemin buna tepki vermesi gerekir. Düşün ki, bir oyun oynuyorsun. Bu oyunda birçok şey olabilir: mesela bir kapı açılabilir, bir düğme basılabilir, bir robot hareket edebilir. Her ne olursa olsun, oyundaki her şey bir olay üzerine çalışır. Mesela, bir düğmeye bastığında, kapı hemen açılır. Bu, "olay"ın bir sonucu olarak gerçekleşir. Yani, bir şey olduğu zaman, sistem de ona tepki verir.

Event-Driven Architecture işte böyle çalışır: Sistem sürekli bekler ve bir olay (mesela "kapı açma") gerçekleştiğinde, o olaya göre hemen bir şeyler yapar. Her şey, olaylara tepki vererek işler. Eğer hiçbir olay olmazsa, sistem de sakin kalır ve bir şey yapmaz.



Yazılım Mimarileri ve Pratikleri

Hexagonal Architecture ve Clean Architecture Nedir?

Hexagonal Architecture (Altın Üçgen Mimari)

Hexagonal Architecture, bir yazılımın çekirdek işlevselliğini (yani iş mantığını) dış dünyadan ayıran bir mimari modeldir. Bu mimari, sistemin iş mantığını, sistemin dış etkileşimlerinden (veritabanı, kullanıcı arayüzü, dış API'ler, vb.) bağımsız hale getirmeyi amaçlar. Dış dünyadaki her şey, port (bağlantı noktası) ve adapter (adaptör) kullanarak iç iş mantığıyla iletişim kurar. Bu sayede yazılımın iş mantığına dokunmadan dış etkileşimleri değiştirmek mümkündür.

Clean Architecture

Clean Architecture, yazılımın katmanlarını belirli kurallara göre ayıran bir mimaridir. Amaç, yazılımı daha esnek, bakımı kolay ve değişime dayanıklı hale getirmektir. Temelde, uygulamanın iç mantığının dış katmanlardan bağımsız olması sağlanır. İç katmanlar (iş mantığı) dış katmanlara (veritabanı, kullanıcı arayüzü vb.) bağımlı olamaz. Dış katmanlar ise iç katmanlara bağımlı olabilir, bu sayede iç mantık, dış faktörlerden etkilenmeden değiştirilebilir.

✨Hayal Ederek Anlama

1. Hexagonal Architecture:

Hexagonal Architecture'ı bir oyun gibi düşün. Oyunda bir karakter (yazılımın iş mantığı) var ve bu karakterin oyun dünyasında etkileşimde bulunması gereken farklı kapılar (bağlantılar) var. Bir kapı, mesela bir kullanıcı arayüzü olabilir, diğer bir kapı ise veritabanı olabilir.

Bu mimaride, karakterin kendisi (yazılımın iş mantığı), sadece kapıları açmak için hazırlanmıştır. Eğer bir kapı değişirse, mesela kullanıcı arayüzü değişirse, karakterin işleyişi etkilenmez, çünkü o yalnızca kapıları kullanır. Bu şekilde, yazılımın iş mantığı, dış etkileşimlerden bağımsız olur.

2. Clean Architecture:

Clean Architecture'ı bir okul gibi düşün. Okulda, öğretmenler (iş mantığı) ders anlatır, ama dersin içeriği okulun dışındaki öğrenciler (dış katmanlar) ile ilgilenmez. Öğretmenler sadece dersin içeriğiyle ilgilenir, yani dersin kalitesiyle.

Okulda başka işler de vardır, örneğin yönetim, öğrenci kayıtları, okul kuralları gibi. Ama öğretmen sadece dersini verir ve dış dünyaya dair hiçbir şeyle uğraşmaz. Eğer okulda bazı yönetim değişiklikleri olursa, öğretmen hala dersine odaklanır. Bu da Clean Architecture'ın mantığına benzer: iş mantığı dış dünyadan bağımsız çalışır.

Kısacası:

- **Hexagonal Architecture,** yazılımın iç mantığını dış dünyadan bağımsız hale getirir ve dış dünyadaki değişiklikler iç mantığı etkilemez.
- **Clean Architecture,** yazılımın katmanlarını ayırarak, iç iş mantığının dış etkileşimlerden bağımsız olmasını sağlar ve bu sayede uygulama daha esnek ve bakımı kolay hale gelir.

Yazılım Mimarileri ve Pratikleri

CQRS (Command Query Responsibility Segregation) Nedir?

CQRS, yazılım mimarisinde "Komut ve Sorgu Sorumluluğunun Ayrılması" anlamına gelir. Bu yaklaşımda, veri okuma işlemleri (sorgular) ve veri yazma işlemleri (komutlar) birbirinden ayrı tutulur. Yani, veriyi güncellemek için kullanılan sistemler (komutlar) ile veriyi sorgulamak için kullanılan sistemler (sorgular) birbirinden farklıdır. Bu sayede, her iki işlem tipi için farklı optimizasyonlar yapılabilir, böylece sistem daha verimli ve ölçeklenebilir hale gelir.

✨Hayal Ederek Anlama

1. CQRS:
Düşün ki bir restoran işletiyorsun ve orada iki tür müşteri var:

- Yemek siparişi verenler (Komutlar): Bu müşteriler yemek isterler, yani bir şeyin yapılmasını isterler.
- Yemek yemek isteyenler (Sorgular): Bu müşteriler yemeklerini yemeyi isterler, yani sadece var olan yemekleri görmek ve almak isterler.

Eğer her iki tür müşteri aynı sırada beklerse, işler karmaşık ve yavaş olur. Ama, CQRS sayesinde, yemek siparişi verenler ve yemek yemek isteyenler farklı sırada beklerler. Yani bir mutfak yemekleri yapmak için çalışırken, bir başka mutfak da yemekleri müşterilere sunar.

Bu şekilde, yemek siparişi almak ve yemek sunmak daha verimli olur çünkü her iş kendi alanında odaklanır. CQRS, bu mantığı yazılım dünyasına taşır: Veri güncellemeleri ve veriyi sorgulama işlemleri birbirinden ayrılır.

2. CQRS Açıklaması:
CQRS, yazılımda veri okuma ve yazma işlemlerini ayırarak her ikisini de daha verimli hale getirmeyi amaçlar. Mesela, bir oyun düşün. Oyuncular bir yandan yeni seviyelere geçmek (yazma - komut) isterken, diğer yandan liderlik tablosunu (okuma - sorgu) görmek isterler. Bu iki işlem birbirine karışmaz ve her biri kendi başına çalışır.

Kısacası:

- **CQRS,** bir işlemi (veri yazma) ve bir diğerini (veri okuma) birbirinden ayırarak her iki işin de daha verimli çalışmasını sağlar. Bu şekilde, yazılımın hem daha hızlı çalışması sağlanır hem de daha kolay yönetilir.

Yazılım Mimarileri ve Pratikleri

SOLID Prensipleri Nedir?

SOLID, yazılım geliştirmede daha kaliteli ve sürdürülebilir kod yazmak için kullanılan 5 temel prensibin baş harflerinden oluşan bir kısaltmadır. Bu prensipler, yazılım tasarımını daha esnek, bakımı kolay ve anlaşılır hale getirmeyi amaçlar.

SOLID prensipleri şunlardır:

1. **S – Single Responsibility Principle (SRP):** Her sınıf, yalnızca bir işlevi yerine getirmelidir.
2. **O – Open/Closed Principle (OCP):** Sınıflar genişlemeye açık, ancak değişime kapalı olmalıdır.
3. **L – Liskov Substitution Principle (LSP):** Alt sınıflar, üst sınıfların yerine geçebilmelidir.
4. **I – Interface Segregation Principle (ISP):** Kullanıcıların kullanmadıkları metodlarla uğraşmalarını engellemek için daha küçük ve özel arayüzler oluşturulmalıdır.
5. **D – Dependency Inversion Principle (DIP):** Yüksek seviyeli modüller, düşük seviyeli modüllere bağlı olmamalıdır. Bunun yerine, her ikisi de soyutlamalara bağlı olmalıdır.



Yazılım Mimarileri ve Pratikleri

✨Hayal Ederek Anlama

1. Single Responsibility Principle (SRP):
Düşün ki bir çalışan yalnızca tek bir iş yapıyor: temizlik yapmak. Eğer bu çalışan temizlik yapmak dışında başka işler de yapmaya başlarsa (örneğin yemek yapmak veya rapor hazırlamak), işler karışır ve işlerin kalitesi düşer. Yani, her çalışanın bir görevi olmalı, çünkü bir kişinin her şeyi yapması karmaşık hale gelir. İşte SRP de şunu söyler: Her sınıf sadece bir iş yapmalı.

2. Open/Closed Principle (OCP):
Bir kitap yazdığını düşün. Kitabın içeriğini değiştirmen gerekirse, kitapları yıpratmadan yeni bölümler eklemek istersin. Kitap, yeni eklemelere açık olmalı, ama eski bölümleri değiştirmemelisin. OCP de aynı şekilde: Kod, yeni özellikler eklemek için açık olmalı, ama mevcut kodu değiştirmemek için kapalı olmalıdır.

3. Liskov Substitution Principle (LSP):
Bir araba ve onun elektrikli versiyonu düşün. Elektrikli araba, normal arabadan beklediğin tüm özellikleri taşımalı. Mesela, aynı şekilde yola çıkabilmeli ve dönme yeteneğine sahip olmalı. Eğer elektrikli araba yolda giderken farklı davranırsa, bu LSP'yi ihlal eder. Yani, alt sınıflar, üst sınıfların yerine geçebilmeli ve aynı şekilde çalışabilmelidir.

4. Interface Segregation Principle (ISP):
Bir tablet düşün. Tabletinde sadece 5 buton olsa, her buton farklı bir işlevi yerine getirir. Ama bir tabletin 50 butonu olsa ve bazıları hiç kullanılmasa, bu tablet karışık ve zor kullanılır hale gelir. ISP'ye göre, insanlar yalnızca ihtiyaç duydukları fonksiyonları içeren arayüzlerle çalışmalıdır. Her şey karmaşık olmamalıdır.

5. Dependency Inversion Principle (DIP):
Bir televizyon düşün. Televizyon, sadece uzaktan kumanda aracılığıyla çalışabilir. Eğer televizyon, kumandayı direkt olarak kullanmak zorunda olursa, işler karmaşıklaşır. Ama DIP'ye göre, televizyon ve kumanda arasındaki bağımlılık, bir aracılık katmanıyla (soyutlama) çözülmelidir. Yani, televizyon ve kumanda arasındaki doğrudan ilişki kesilir ve her ikisi de bir tür soyutlama ile bağlı olur.

Kısacası, SOLID prensipleri, yazılımcıların daha temiz, daha esnek ve kolay bakım yapılabilir yazılımlar geliştirmelerine yardımcı olur. Bu prensipler, yazılımın büyüdükçe daha sağlam ve daha sürdürülebilir olmasını sağlar.

Yazılım Mimarileri ve Pratikleri

Dependency Injection (DI) Nedir?

Dependency Injection (DI), yazılım tasarımında bir nesnenin ihtiyaç duyduğu diğer nesneleri (bağımlılıkları) dışarıdan sağlamaktır. Yani, bir sınıf veya nesne, çalışabilmesi için gerekli olan diğer nesneleri kendisi yaratmak yerine, bu nesneler dışarıdan ona enjekte edilir. DI, yazılımda bağımlılıkların yönetimini daha kolay ve esnek hale getirir. Bu yaklaşım, kodun daha test edilebilir, yeniden kullanılabilir ve bakımı kolay olmasını sağlar.

✨Hayal Ederek Anlama

1. Dependency Injection:
Düşün ki, bir pasta yapmak istiyorsun. Ama pastayı yapabilmen için un ve yumurta gibi malzemelere ihtiyacın var. Eğer her seferinde pasta yaparken un ve yumurtayı kendin alman gerekirse, bu çok zaman alır ve işler karmaşıklaşır.
Dependency Injection tam burada devreye girer. Birisi, sana her seferinde un ve yumurtayı getirip verir. Böylece, pastayı yapmaya odaklanabilirsin, malzemeleri düşünmene gerek kalmaz. Yani, pastayı yapabilmen için gerekli olan malzemeler (bağımlılıklar) sana dışarıdan sağlanır.

2. Dependency Injection Açıklaması:
Bu durumu yazılımda şöyle düşün: Eğer bir sınıf (örneğin, bir oyuncak yapıcı sınıfı) başka sınıflardan (örneğin, uzaktan kumanda, motor, batteri gibi) bazı şeylere ihtiyaç duyuyorsa, bu ihtiyaçları kendisi yaratmak yerine, dışarıdan ona enjekte ederiz.

Yani, her oyuncak yapıcı, her seferinde kendi motorunu ve bataryasını yapmaz. Bunun yerine, motor ve batarya gibi parçalar başka bir yerden gelir ve oyuncak yapıcı sadece bunları kullanır.

Neden Dependency Injection Kullanılır?

- **Esneklik sağlar:** Eğer motorun türünü değiştirmek istersen, sadece motoru değiştiririz, yapıcıyı değiştirmeye gerek yoktur.
- **Test Edilebilirlik:** Eğer oyuncak yapıcıyı test etmek istiyorsan, motoru kolayca değiştirebilirsin, böylece test için daha uygun bir motor kullanabilirsin.
- **Yeniden Kullanılabilirlik:** Motoru farklı yapıcılarda da kullanabilirsin. Yani bir motoru sadece bir yapıcıya değil, başka yapıcılara da verebilirsin.

Kısacası, Dependency Injection ile yazılımda her nesne, kendi bağımlılıklarını dışarıdan alır. Bu, kodu daha daha esnek, kolay test edilebilir ve bakımı kolay hale getirir.

Yazılım Testleri ve Kalite Güvencesi

Unit, Integration ve End-to-End Testler Nedir?

Unit Test (Birim Testi):
Unit Test, yazılımın en küçük parçalarını (birimleri) test etmeyi amaçlar. Genellikle bir fonksiyon veya metodun doğru çalışıp çalışmadığını kontrol etmek için kullanılır. Bu test, bağımsız olarak çalışan küçük kod parçalarının doğru sonuç verip vermediğini doğrular.

Integration Test (Entegrasyon Testi):
Integration Test, birden fazla birimin bir arada çalışmasını test eder. Yani, farklı parçaların birbiriyle düzgün bir şekilde iletişim kurup kurmadığını kontrol eder. Bu test, birimlerin birleştiğinde sistemin beklenen şekilde çalışıp çalışmadığını inceler.

End-to-End Test (Uçtan Uca Testi):
End-to-End Test, yazılımın tüm işleyişini baştan sona kadar test eder. Kullanıcı gözünden tüm süreçlerin doğru şekilde çalışıp çalışmadığını kontrol eder. Sistem tüm bileşenleriyle birlikte test edilir ve gerçek dünya senaryolarında sistemin nasıl tepki verdiği gözlemlenir.

✨Hayal Ederek Anlama

1. Unit Test:
Düşün ki bir puzzle yapıyorsun. Unit Test, puzzle'ın her bir parçasını tek tek kontrol etmek gibidir. Her parçayı tek başına denersin ve doğru şekilde oturup oturmadığını görürsün. Eğer bir parça yanlış yerleştirilmişse, sadece o parçayı düzeltiriz. Ama sadece tek bir parça üzerinde odaklanılır.

2. Integration Test:
Şimdi puzzle'ın tamamını kurduğunda, parçaların birbirine nasıl uyduğuna bakman gerek. Integration Test, puzzle'ın tüm parçalarının doğru şekilde birleşip birleşmediğini test eder. Her parça birbiriyle uyumlu çalışmalı. Yani, bir parça doğru olmalı, diğerleriyle birleştiğinde puzzle tamamlanmalıdır.

3. End-to-End Test:
End-to-End Test, puzzle'ın tamamlandığında gerçekten yerleştirilen yerin doğru olup olmadığını ve tamamlanıp tamamlanmadığını kontrol etmektir. Bu, puzzle'ı tamamen tamamlayıp ne kadar iyi bir görüntü ortaya çıktığını görmek gibidir. Gerçekten puzzle'ın tüm süreci doğru şekilde tamamlanıp tamamlanmadığına bakılır.

Kısacası:
- **Unit Test:** Küçük, bağımsız birimleri test eder (puzzle'ın her parçasını tek tek kontrol etmek gibi).
- **Integration Test:** Bu birimlerin bir arada çalışıp çalışmadığını test eder (parçaların birbirine nasıl uyduğunu kontrol etmek gibi).
- **End-to-End Test:** Tüm sistemin, gerçek dünyadaki gibi doğru çalışıp çalışmadığını test eder (puzzle'ın tamamlanıp doğru şekilde yerleştirilip yerleştirilmediğini görmek gibi).

Bu testler, yazılımın her aşamasının doğru çalışıp çalışmadığını kontrol ederek hataların erkenden bulunmasına yardımcı olur ve daha güvenilir bir yazılım oluşturur.

Yazılım Testleri ve Kalite Güvencesi

Test Framework'leri (Jest, Mocha, JUnit, PyTest)

Test framework’leri, yazılım geliştirme sürecinde yazılımın doğru çalışıp çalışmadığını test etmek için kullanılan araçlardır. Bu framework'ler, testlerin yazılmasını, çalıştırılmasını ve raporlanmasını kolaylaştırır.

Jest:

Jest, JavaScript uygulamaları için geliştirilmiş bir test framework’üdür. Özellikle React gibi kütüphanelerle uyumlu çalışır. Jest, testleri otomatik olarak çalıştırır, beklenen sonuçları kontrol eder ve sonuçları raporlar.

Mocha:

Mocha, Node.js ve JavaScript için kullanılan başka bir test framework'üdür. Jest'ten farklı olarak daha esnek bir yapıya sahiptir ve testler yazarken farklı assertion (doğrulama) kütüphaneleri ile kullanılabilir.

JUnit:

JUnit, Java programlama dili için bir test framework’üdür. JUnit, Java uygulamalarındaki kodun doğruluğunu test etmek için yaygın olarak kullanılır ve testlerin kolayca yazılmasını sağlar.

PyTest:

PyTest, Python dili için kullanılan popüler bir test framework'üdür. Hem küçük birim testlerini hem de büyük entegrasyon testlerini yazmayı kolaylaştırır. PyTest, testlerin hızlı bir şekilde çalıştırılmasını sağlar ve Python projelerinde yaygın olarak kullanılır.

Yazılım Testleri ve Kalite Güvencesi

✨Hayal Ederek Anlama

1. Jest:
Düşün ki bir video oyunu yapıyorsun ve oyunun her bölümünün doğru çalışıp çalışmadığını görmek istiyorsun. Jest tam olarak buna benzer; oyununu test etmek için bir "oyun testi" aracı gibidir. Jest, oyun boyunca her seviyenin doğru işlediğini kontrol eder ve herhangi bir hata varsa hemen bildirir. Jest, özellikle React gibi oyun yapılarında kullanılır.

2. Mocha:
Mocha da bir test aracı ama daha esnek ve özelleştirilebilir. Mocha, sanki bir şirket çalışanı gibidir. Çalışan, farklı test araçlarıyla çalışabilir ve istediği gibi testler yapabilir. Mocha, her testin nasıl çalışacağını sana bırakır ama her şeyi organize eder ve düzenli bir şekilde test eder.

3. JUnit:
JUnit, Java dilinde yazılmış oyunları test etmek için kullanılan bir araçtır. Diyelim ki bir yazılım ya da oyun yapıyorsun ve her bir kod parçasının doğru çalıştığından emin olmak istiyorsun. JUnit, her bir parçayı tek tek kontrol eder ve bir hata varsa hemen bildirir. Java oyunlarında genellikle bu aracı kullanırız.

4. PyTest:
PyTest de, tıpkı bir ev temizleme aracı gibi düşün. Evde her şeyin düzgün çalışıp çalışmadığını görmek için her odada testler yapar ve eksik ya da yanlış bir şey varsa hemen gösterir. PyTest, Python dilinde yazılan projeleri test eder ve kodun hatasız çalışmasını sağlar.

Kısacası:

- **Jest:** JavaScript ve React projeleri için otomatik test aracı.
- **Mocha:** JavaScript ve Node.js projelerinde esnek test aracı.
- **JUnit:** Java dilindeki yazılımları test etmek için kullanılan bir araç.
- **PyTest:** Python projeleri için test yazmayı kolaylaştıran araç.

Bu test framework'leri, yazılımın doğru çalışıp çalışmadığını kontrol etmek ve hataları erken tespit etmek için kullanılır, böylece daha sağlam ve güvenilir yazılımlar geliştirilebilir.

Yazılım Testleri ve Kalite Güvencesi

Mocking ve Dependency Injection Nedir?

Mocking

Mocking, yazılım testlerinde kullanılan bir tekniktir. Gerçek bir nesne veya bileşen yerine, testlerin çalışabilmesi için sahte (mock) nesneler oluşturmak anlamına gelir. Mocking, genellikle birim testlerinde, dış bağımlılıkların (örneğin bir veritabanı bağlantısı veya üçüncü parti bir API) yerine sahte nesneler kullanılarak yapılan testlerde kullanılır. Bu, testlerin daha hızlı ve güvenilir bir şekilde yapılmasını sağlar.

Dependency Injection (DI)

Dependency Injection (DI), yazılımda bir nesnenin ihtiyaç duyduğu diğer nesneleri dışarıdan sağlamak anlamına gelir. Yani, bir nesne veya sınıf, bağımlılıklarını (örneğin veritabanı bağlantısı, başka bir sınıf, vb.) kendisi yaratmaz, bunlar dışarıdan ona verilir. Bu, yazılımın daha esnek, test edilebilir ve sürdürülebilir olmasını sağlar.

✨Hayal Ederek Anlama

1. Mocking

Düşün ki, bir doktor oluyorsun ve hastaları iyileştirmek için medikal cihaz kullanman gerekiyor. Ama gerçek bir cihaz yok. Bunun yerine, sadece sahte bir cihaz (mock cihaz) kullanıyorsun. Bu sahte cihaz, gerçek cihazın yerine geçiyor ve senin testlerini yapabilmeni sağlıyor.

Mocking, yazılımda da gerçek nesneler yerine sahte nesneler kullanmak gibidir. Eğer bir test yapıyorsan ve dış bağımlılıkların (mesela bir veritabanı veya internet bağlantısı) olması gerekiyorsa, gerçeklerini kullanmak yerine sadece işini görebilecek sahte nesneler oluşturursun. Bu şekilde daha hızlı testler yapabilir ve gerçek sistemdeki hatalardan etkilenmeden yazılımını test edebilirsin.

2. Dependency Injection (DI)

Şimdi, bir şef olduğunu düşün ve yemek yapmak için malzemelere ihtiyacın var. Ama her seferinde malzemeleri almakla uğraşmak istemiyorsun. Bunun yerine, birisi sana her seferinde gerekli malzemeleri getiriyor. Böylece sen sadece yemeği hazırlamaya odaklanabilirsin.

Dependency Injection de yazılımda aynen böyle çalışır. Diyelim ki bir sınıfın, başka bir sınıfı kullanması gerekiyor. O sınıfı kendisi yaratmaz, başka bir yerden alır. Yani her sınıf, neye ihtiyacı varsa dışarıdan sağlanan bağımlılıkları kullanır.

Özetle:

- **Mocking:** Gerçek nesneler yerine sahte nesneler kullanarak test yapma tekniğidir (mesela sahte bir cihaz kullanarak testi yapmak gibi).
- **Dependency Injection:** Nesnelerin, ihtiyaç duydukları diğer nesneleri dışarıdan almasıdır (mesela şefin yemek yaparken malzemeleri dışarıdan alması gibi).

Her iki kavram da yazılım geliştirmede, testlerin daha kolay yapılmasını sağlar ve bağımlılıkları yönetmeyi kolaylaştırarak kodun daha esnek, güvenilir ve bakımı kolay hale gelmesini sağlar.

Gelişmiş Yazılım Pratikleri

Test-Driven Development (TDD) Nedir?

Test-Driven Development (TDD), yazılım geliştirme yaklaşımında, önce testlerin yazılması, sonra testleri geçecek kodun yazılması sürecini ifade eder. Bu yöntem, yazılım geliştirme sırasında sürekli olarak testler yazarak, kodun doğru çalışıp çalışmadığını anında kontrol etmeyi amaçlar.

TDD süreci şu şekilde işler:

1. **Test yaz:** Önce ne yapmak istediğine dair bir test yazılır.
2. **Kod yaz:** Testi geçecek kadar küçük bir kod yazılır.
3. **Testi çalıştır:** Testi çalıştırıp geçip geçmediğini kontrol et.
4. **Kodunu iyileştir:** Kod çalışıyorsa, kodu daha iyi hale getirebiliriz (refactor).
5. **Tekrar et:** Bu adımlar, her yeni özellik ekledikçe tekrar edilir.

✨Hayal Ederek Anlama

Test-Driven Development (TDD), yazılım geliştirmede önce testi yazmak, sonra kodu yazmak demektir.

Bir örnekle açıklayalım:

Düşün ki bir oyun yapıyorsun ve oyunda bir karakterin zıplamasını istiyorsun. Ama oyunun nasıl zıplaması gerektiğini bilmek için önce test yazman gerekiyor. Test, "Karakter zıpladığında yüksekliğinin belirli bir seviyeye çıkması gerekir" şeklinde olabilir.

Adım adım nasıl olur:

1. **Test yaz:** İlk önce "Karakter zıpladığında, yukarı doğru 10 birim yükselmeli" diye bir test yazıyorsun.
2. **Kod yaz:** Şimdi, bu testi geçecek bir zıplama hareketi kodluyorsun.
3. **Testi çalıştır:** Testi çalıştırıyorsun. Eğer test geçerse, yani karakter zıplarsa, kodun doğru çalıştığını anlıyorsun.
4. **Kodunu geliştir:** Eğer zıplama düzgünse, kodu daha verimli hale getirebilirsin. Eğer test geçmiyorsa, hatayı bulup düzeltirsin.

Ve sonra yeni özellikler eklerken de aynı adımları izlersin.

Kısacası:

Test-Driven Development (TDD), yazılım geliştirirken önce test yazmak, sonra testi geçecek kodu yazmak demektir. Bu şekilde yazdığın her kodun doğru çalıştığından emin olursun ve hataları daha hızlı bulursun.

TDD'nin amacı, yazılımın her parçasının doğru çalıştığını garantilemek ve geliştirilen yazılımın her zaman güvenilir ve hatasız olmasını sağlamaktır.

Gelişmiş Yazılım Pratikleri

Debugging

Debugging, yazılım geliştirme sürecinde karşılaşılan hataların (bug'ların) bulunması, çözülmesi ve düzeltilmesi işlemidir. Yazılım çalışırken bazı hatalar meydana gelebilir, ve debugging, bu hataları tespit edip ortadan kaldırmaya yönelik yapılan işlemlerdir. Genellikle, geliştirici hata mesajlarını ve kodu dikkatlice inceleyerek hatanın nedenini anlamaya çalışır ve ardından kodu düzelterek hatayı giderir.

✨Hayal Ederek Anlama

Debugging, yazılımdaki hataları bulma ve düzeltme sürecidir.
Düşün ki, büyük bir yapboz yapıyorsun ve sonunda tüm parçalar birleşiyor ama bir parça eksik gibi görünüyor. O zaman ne yaparsın? Eksik olan parçayı bulmaya çalışırsın. Debugging de yazılımda tam olarak böyle çalışır. Kodda bir şeylerin yanlış olduğunu fark ettiğinde, bu hatayı bulup düzeltmek için araştırma yaparsın.

Adım adım debugging şöyle olur:

1. **Hata bulma:** Yazılımını çalıştırıyorsun ve bazı şeyler doğru çalışmıyor. Mesela, oyun açılmıyor.
2. **Nedenini bulma:** Hata mesajlarına bakıyorsun ve yazdığın koda göz gezdiriyorsun. Hangi kod parçasının yanlış olduğunu buluyorsun.
3. **Düzeltme:** Yanlış olan yeri bulduktan sonra, oradaki hatayı düzeltiyorsun. Mesela, yanlış bir işaret veya yanlış yazılmış bir kod olabilir.
4. **Tekrar deneme:** Hata düzeldiyse, yazılımı tekrar çalıştırıp doğru çalışıp çalışmadığını kontrol ediyorsun.

Kısacası:
Debugging, yazılımda hataları bulmak ve düzeltmek demektir. Eğer bir şey yanlış çalışıyorsa, hata mesajları ve kodu inceleyerek sorunu çözersin ve tekrar doğru çalışmasını sağlarsın.

Debugging, yazılımcıların yazılımı daha güvenilir ve hatasız hale getirmesini sağlar, tıpkı bir yapbozda eksik parçayı bulup yerine yerleştirmek gibi.



Gelişmiş Yazılım Pratikleri

Design Patterns (Singleton, Factory, Observer)

Design Patterns (Tasarım Desenleri), yazılım geliştirme sürecinde karşılaşılan yaygın problemlere yönelik tekrarlanabilir çözümler sunan şablonlardır. Bu desenler, yazılımın daha düzenli, esnek ve sürdürülebilir olmasını sağlamak için kullanılır.

Singleton Pattern

Singleton, bir sınıfın yalnızca bir tane örneğinin (instance) oluşturulmasını sağlar. Yani, sınıfın sadece bir tane nesnesi vardır ve bu nesne her yerden aynı şekilde erişilebilir.

Factory Pattern

Factory, nesnelerin oluşturulmasını bir fabrika aracılığıyla yapar. Yani, nesneler doğrudan oluşturulmaz, bunun yerine bir fabrika sınıfı tarafından sağlanır. Bu, nesne oluşturmayı daha esnek hale getirir.

Observer Pattern

Observer, bir nesne değiştiğinde, bu değişikliği gözlemleyen (observer) diğer nesnelere bildirir. Yani, bir nesne bir şey değiştirdiğinde, ona bağlı diğer nesneler de otomatik olarak bilgilendirilir ve buna göre tepki verir.

✨Hayal Ederek Anlama

1. Singleton Pattern

Düşün ki, bir okul müdürü var ve okulda sadece bir tane müdür var. Herkes ona danışmak için aynı müdüre gider. Singleton, yazılımda da aynıdır: Bir sınıfın sadece bir tane nesnesi oluşturulur ve bu nesne tüm sistemde aynı kalır. Yani, okulda sadece bir müdür olduğu gibi, yazılımda da sadece bir tane nesne vardır.

Örnek: Bir yazılımda, sadece bir tane veritabanı bağlantısı olması gerektiğinde, Singleton deseni kullanılır.

2. Factory Pattern

Bir oyuncak fabrikası düşün. Bu fabrikada birçok farklı oyuncak üretiliyor ama oyuncakları sen direkt yapmıyorsun, fabrikaya gidip oradan alıyorsun. Factory, yazılımda da bir "fabrika" gibi çalışır: Farklı türde nesneleri doğrudan oluşturmazsın, bir fabrika sınıfı aracılığıyla alır ve kullanırsın.

Örnek: Bir oyun yapıyorsan, farklı karakterleri Factory kullanarak oluşturabilirsin (mesela bir zombi veya bir kahraman gibi). Her karakteri kendin değil, bir fabrika sınıfı yaratır.

Gelişmiş Yazılım Pratikleri

3. Observer Pattern

Düşün ki, bir haber ajansı var ve bu ajans, gazetecilere sürekli haber veriyor. Eğer ajans bir haber yaparsa, tüm gazeteciler bu haberi hemen öğrenir. Observer, yazılımda da böyle çalışır: Bir nesne bir değişiklik yaptığında, buna bağlı olan diğer nesneler de otomatik olarak bilgilendirilir.

Örnek: Bir oyun yapıyorsun ve oyuncunun canı azaldığında, diğer nesneler (mesela bir sağlık çubuğu) hemen can durumunu günceller. Burada Observer deseni devreye girer.

Kısacası:

- **Singleton:** Yazılımda sadece bir tane nesne olması gerektiğinde kullanılır (okulda sadece bir müdür gibi).
- **Factory:** Nesneleri doğrudan yaratmak yerine bir fabrika aracılığıyla yaratırız (oyuncakları fabrika gibi üretmek).
- **Observer:** Bir nesne değiştiğinde, bağlı nesnelerin de otomatik olarak bilgilendirildiği bir yapıdır (haber ajansından gazetecilere haber gitmesi gibi).

Tasarım desenleri, yazılım geliştirmeyi daha düzenli, esnek ve anlaşılır hale getiren çözümler sunar.

Gelişmiş Yazılım Pratikleri

Clean Code Nedir?

Clean Code (Temiz Kod), yazılım geliştirmede okunabilir, anlaşılır, bakımı kolay ve sürdürülebilir kod yazma pratiğidir. Temiz kod, hem diğer yazılımcılar tarafından hem de yazılımın gelecekteki bakımı için kolayca anlaşılabilir olmalıdır. Bu, gereksiz karmaşıklıklardan kaçınmak, anlaşılması zor kod parçalarından uzak durmak ve kodun amacına uygun, düzenli olmasını sağlamak anlamına gelir.

Temiz kodun bazı temel özellikleri:

- **Okunabilirlik:** Kodun anlamı kolayca anlaşılabilir.
- **Basitlik:** Gereksiz karmaşıklıklardan kaçınılır.
- **Bakım Kolaylığı:** Kod zamanla değişse de kolayca güncellenebilir.
- **Tekrar Kullanılabilirlik:** Kod yeniden kullanılabilir.

✨Hayal Ederek Anlama

Clean Code, yazılımda temiz, düzenli ve anlaşılır yazılmış koddur.
Düşün ki, bir odada oyuncaklar var ve oyuncakları düzgün bir şekilde yerleştirmişsin. Her oyuncak belli bir yere konmuş ve odanın her köşesi düzenli. Bu durumda odada bir şey kaybolursa, kolayca bulabilirsin ve odanın dağılmasını engellersin.

Temiz Kod da yazılımda aynıdır: Kod düzenli olmalı, her şey yerli yerinde olmalı, böylece yazılımda bir şey değiştiğinde veya hata olduğunda, hızlıca bulup düzeltebilirsin.



Gelişmiş Yazılım Pratikleri

Temiz Kodun Temel Özellikleri:

Okunabilirlik:

Temiz kod, başkalarının veya kendinin gelecekte kolayca anlayabileceği şekilde yazılır. Mesela, kodu yazarken anlamlı isimler verilir (değişkenlere veya fonksiyonlara). Bu, kodu okuyan kişinin ne olduğunu hemen anlamasını sağlar.

Örnek: Bir değişkenin ismi age yerine personAge olmak daha anlaşılırdır.

- Basitlik: Temiz kod, gereksiz karmaşıklıktan kaçınır. Kodu karmaşık hale getirecek fazla özellik eklemek yerine, basit bir çözüm tercih edilir.

Örnek: Bir oyun karakterinin zıplama fonksiyonu çok karmaşık yerine, sadece gerekli olan kodla yapılır.

- **Modülerlik:** Temiz kod, büyük işlevler yerine küçük, tek bir sorumluluğu olan fonksiyonlar ve sınıflar kullanır. Bu sayede, bir şey değiştirmek gerektiğinde yalnızca ilgili kısmı değiştirirsiniz ve tüm kod etkilenmez.

Örnek: Oyunda bir karakterin canını hesaplamak için ayrı bir fonksiyon, skorunu hesaplamak için ayrı bir fonksiyon yazmak.

- **Yorumlar ve Dokümantasyon:** Temiz kodda, karmaşık kısımlar yorumlarla açıklanır. Ancak, kodun kendisi açıklayıcı olmalı; yorumlar sadece gerekli olduğunda kullanılmalıdır.

Örnek: "Bu fonksiyon, karakterin canını artırmak için kullanılır" gibi açıklamalar.

Kısacası:

Clean Code yani Temiz Kod, yazılımda düzenli, basit, anlaşılır ve kolayca bakımı yapılabilir kod yazmak demektir. Kodun karmaşık olmaması, kolayca anlaşılabilir olması, hataların daha hızlı bulunabilmesi ve gelecekte de kolayca değiştirilebilmesi için çok önemlidir.

Temiz kod yazmak, yazılımcıların uzun vadede işlerini kolaylaştırır ve başkalarının da yazdıkları kodu anlamasını sağlar.

Teşekkürler!

Sevgili Okuyucu,

Bu e-kitabı okuma yolculuğunuzda bana eşlik ettiğiniz için teşekkür ederim. Web geliştirme dünyasına dair kavramları, sadece teknik terimler olarak değil, aynı zamanda hayal gücümüzle şekillenen birer anlatı olarak sunmayı amaçladım. Umarım bu yaklaşım, sizde merak uyandırmış ve öğrenme sürecinizi daha keyifli hale getirmiştir.

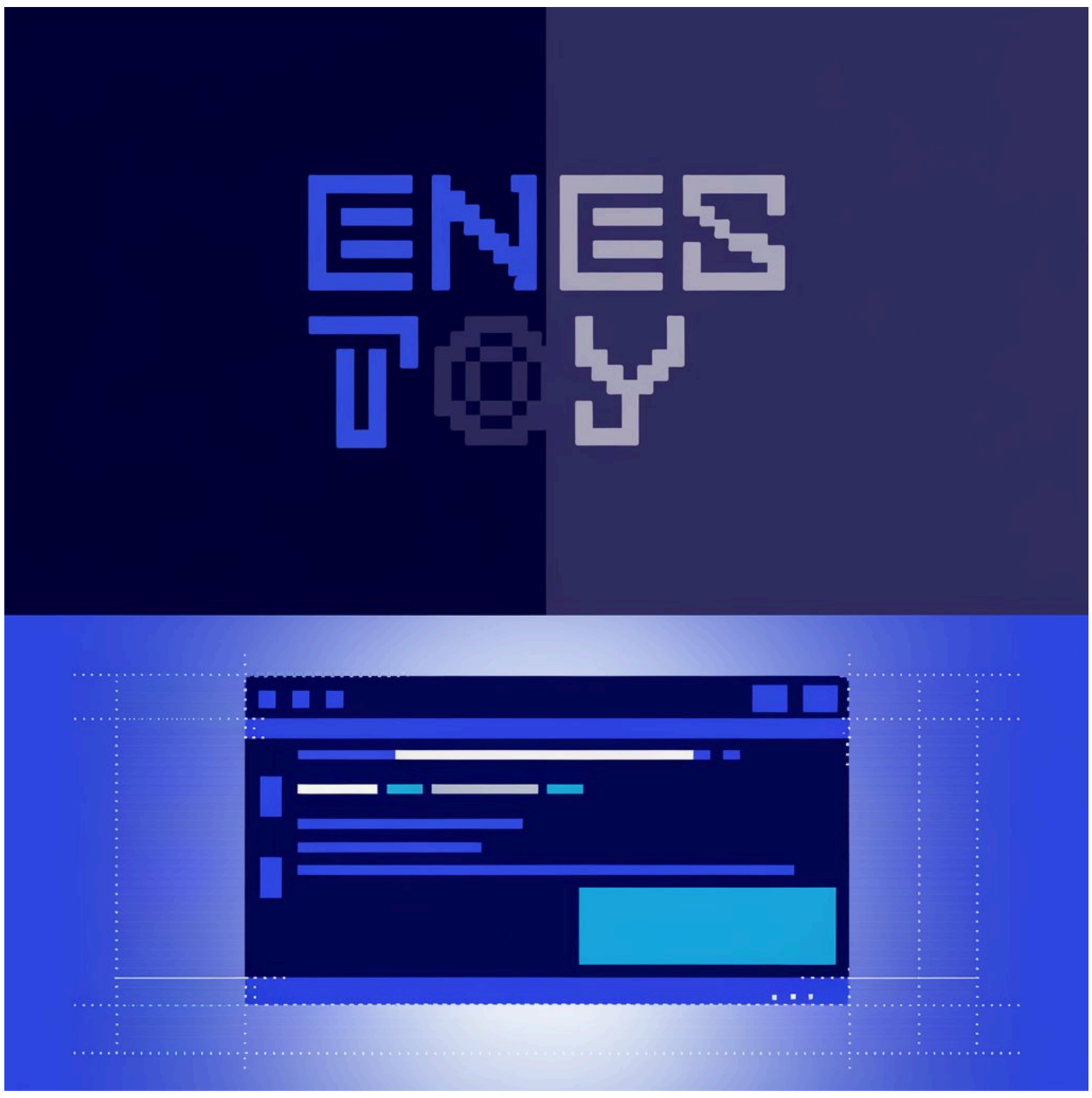
Bu kitabın daha da gelişebilmesi ve daha fazla kişiye ulaşabilmesi için sizden gelecek geri bildirimler benim için çok kıymetli. Görüşlerinizi, önerilerinizi veya aklınıza takılan her türlü soruyu benimle paylaşabilirsiniz. Ayrıca, web teknolojileri ve yazılım alanındaki diğer projelerimi incelemek isterseniz, aşağıdaki bağlantılardan bana ulaşabilirsiniz.

İletişim ve Bağlantılar:
📧 **E-posta:** enestoy@protonmail.com
🔗 **LinkedIn:** linkedin.com/in/enestoy
💻 **GitHub:** github.com/enestoy
🌐 **Web Site**: www.enestoy.com

Gelişmeye, üretmeye ve paylaşmaya devam etmeniz dileğiyle...

Keyifli projelerde tekrar buluşmak üzere!

Başarılar dilerim.



Telif Hakları ve Görsel Kullanım Bildirimi

Telif Hakkı Bilgilendirmesi

© 2025 Enes Toy – Tüm Hakları Saklıdır.
Bu e-kitabın tüm içeriği (metinler, grafikler, yapılar) **Enes Toy**'a aittir.

İçeriğin tamamı veya bir bölümü, yazarın yazılı izni olmadan hiçbir dijital veya basılı ortamda kopyalanamaz, dağıtılamaz, çoğaltılamaz, satılamaz.

Bu kitap, **5846 Sayılı Fikir ve Sanat Eserleri Kanunu** kapsamında korunmaktadır. Herhangi bir izinsiz kullanım durumunda, yasal işlem başlatma hakkı saklıdır.

Görsel Kullanım ve Lisanslar

Bu kitapta kullanılan:

- 30 adet görsel, lisanslı Canva Pro kütüphanesinden alınmıştır.
- Diğer görseller, telifsiz (CC0) kaynaklardan temin edilmiştir veya yazara aittir.
- Telifli olduğu tespit edilen görseller kaldırılmıştır.

Görsellerin yasal kullanımına özen gösterilmiştir. Tespit edilen bir hak ihlali durumunda yazar ile iletişime geçilmesi rica olunur.

ENES TOY