

Uygulama Programı Arayüzü



Öğr. Gör.
Furkan DURMUŞ

Bu Haftanın Ders Kazanımları

1

API'ye Giriş

2

Temel Tanımlar

3

HTTP Temelleri

4

HTTP Status Code'lar

API 'ye Giriş ➔



API Kavramı

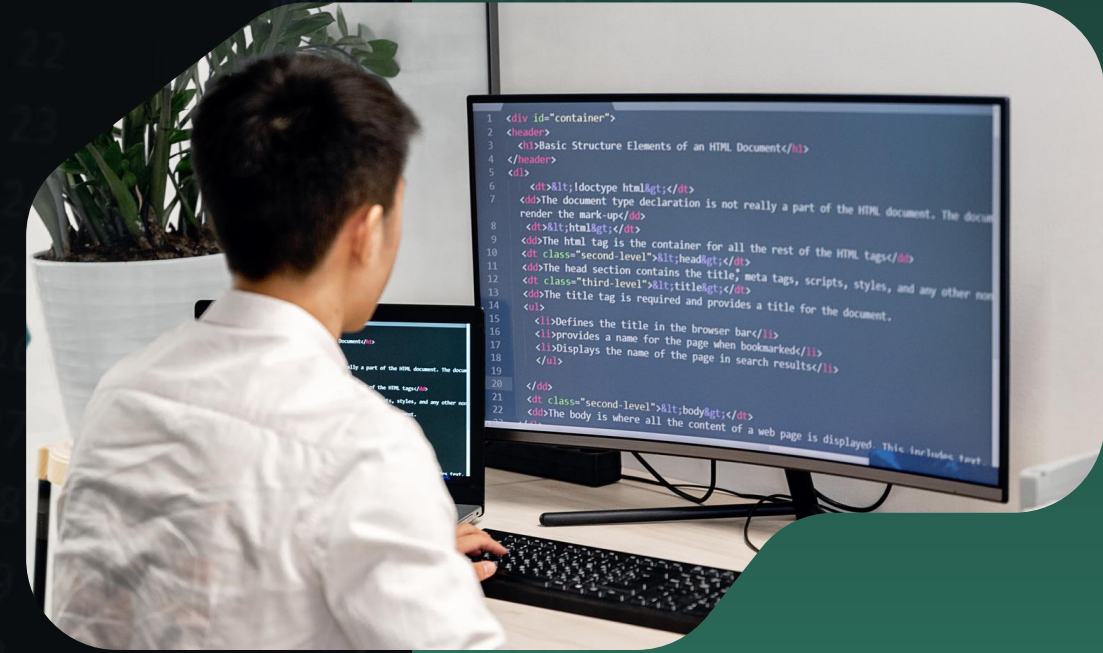
Günümüzde geliştirdiğimiz yazılımlar genellikle tek başına çalışmaz.

Bir uygulama, başka bir uygulamanın sunduğu bilgiye veya özelliğe ihtiyaç duyabilir. Örneğin bir hava durumu uygulaması geliştirdiğimizi düşünelim. Uygulamamızda şu bilgileri göstermek istiyoruz:

- Samsun'un hava sıcaklığı
- Ankara'da yağmur olup olmadığı
- İstanbul'daki rüzgâr hızı
- Önümüzdeki günlerin hava durumu

Burada bir problem ortaya çıkar:

Hava durumu bilgisini nereden bulacağız?



API Kavramı

Hava durumu bilgisini nereden bulacağız?

Hava durumu ölçümlerini kendimiz yapmak zorunda kalabiliriz. Bunun için meteorolojik sensörler, ölçüm istasyonları ve çok büyük bir veri altyapısı gerekir. Bu, basit bir uygulama geliştirmek isteyen yazılımcı için oldukça zahmetlidir.

Bunun yerine hava durumu verisini sağlayan bir servisin sunduğu **API'den** faydalanabiliriz. Bu durumda bizim uygulamamız hava durumu verisini sıfırdan üretmek zorunda kalmaz. İhtiyaç duyduğu bilgiyi, bu bilgiyi sağlayan **başka bir yazılımın sunduğu arayüz üzerinden** kullanır.



API Kavramı

Burada API'nin temel amacı ortaya çıkar:

API, bir yazılımın veya hizmetin sunduğu özelliklere ve verilere, belirlenmiş kurallar aracılığıyla erişmemizi sağlayan arayüzdür.



API Kavramı

API açılımı nedir?



API Kavramı

Application

Programming

Interface



API Kavramı

Application

Programming

Interface

Uygulama Programı(Programlama) Arayüzü



API Kavramı

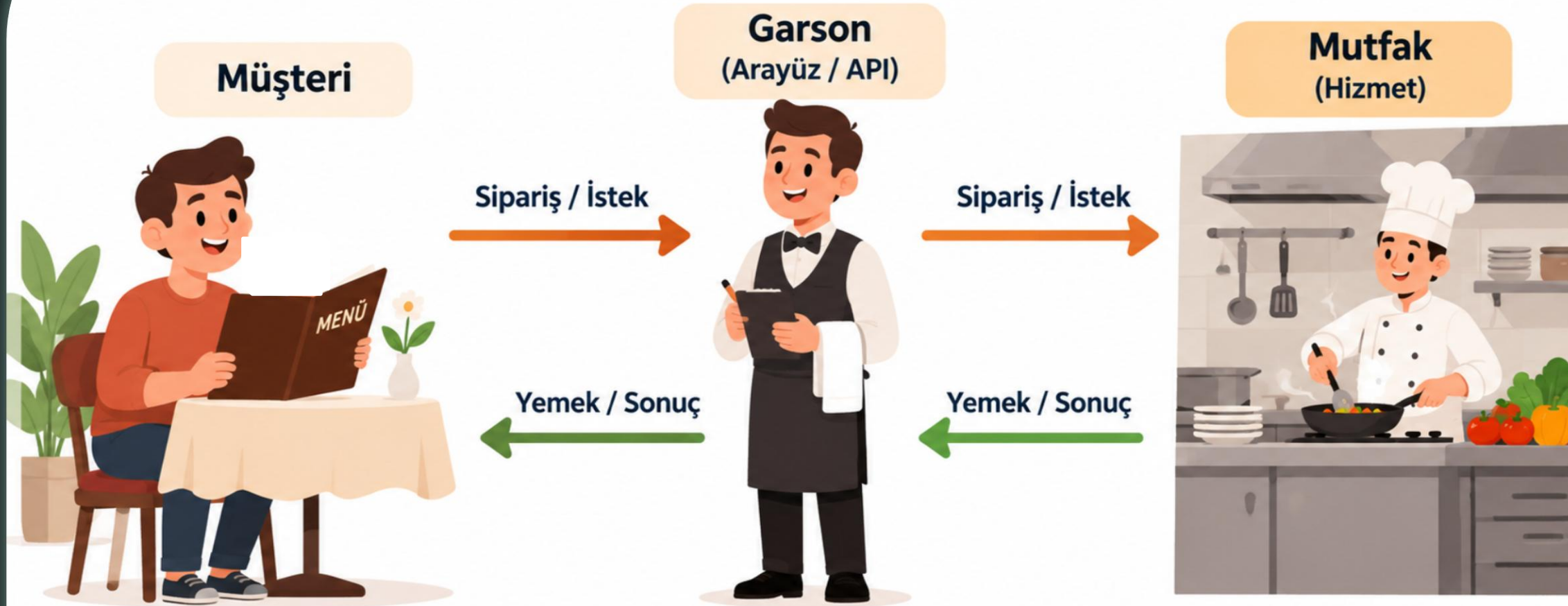
Buradaki arayüz kelimesi görsel bir ekran anlamına gelmez. Günlük kullanımda arayüz dediğimizde çoğunlukla butonları, menüleri ve ekranları düşünürüz. Ancak API'deki arayüz, genellikle yazılımların birbirleriyle etkileşim kurmasını sağlayan programlama arayüzüdür.

Yani API, yazılımlar arasında bir erişim noktası veya iletişim arayüzü olarak düşünülebilir.



API Kavramı

API'yi Anlamak: Restoran Benzetmesi



Müşteri mutfagın içini bilmeden hizmet alır.
Yazılımlarda da API benzer şekilde arayüz görevi görür.

API'nin Temel Amacı



API'nin temel amacı, bir yazılımın sahip olduğu işlevlerin veya verilerin başka yazılımlar tarafından kullanılabilmesini sağlamaktır.

Bir yazılımın içerisinde birçok işlem bulunabilir. Örneğin bir üniversite otomasyon sistemini ele alalım.

Bu sistemde:

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri

```
.box{
  position: absolute;
  transform: translate(-50%, -50%);
  width: 400px;
  padding: 40px;
  background: linear-gradient(to top right, #000 49%, #fff 49%, #fff 51%, #000 51%);
  box-sizing: border-box;
  box-shadow: 0 15px 25px #000;
  border-radius: 10px;
}
.box h2{
  margin: 0 0 30px;
  padding: 0;
  color: #fff;
  text-align: center;
}
.box h3{
  margin: 0 0 10px;
  padding: 0;
  color: #fff;
  text-align: center;
}
.box .input{
```

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri

```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 400px;
  padding: 40px;
  background: #000000;
  box-sizing: border-box;
  box-shadow: 0 15px 25px #000000;
  border-radius: 10px;
}
.box h2{
  margin: 0 0 30px;
  padding: 0;
  color: #ffff;
  text-align: center;
}
.box h3{
  margin: 0 0 10px;
  padding: 0;
  color: #ffff;
  text-align: center;
}
.box input{
```

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri



Web Sitesi

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri



Web Sitesi

Mobil Uygulama

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri

Web Sitesi

Mobil Uygulama

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri

Web Sitesi

Mobil Uygulama



API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri



Web
Sitesi



API

Mobil
Uygulama

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri



SamUlaş
App

API

Web
Sitesi

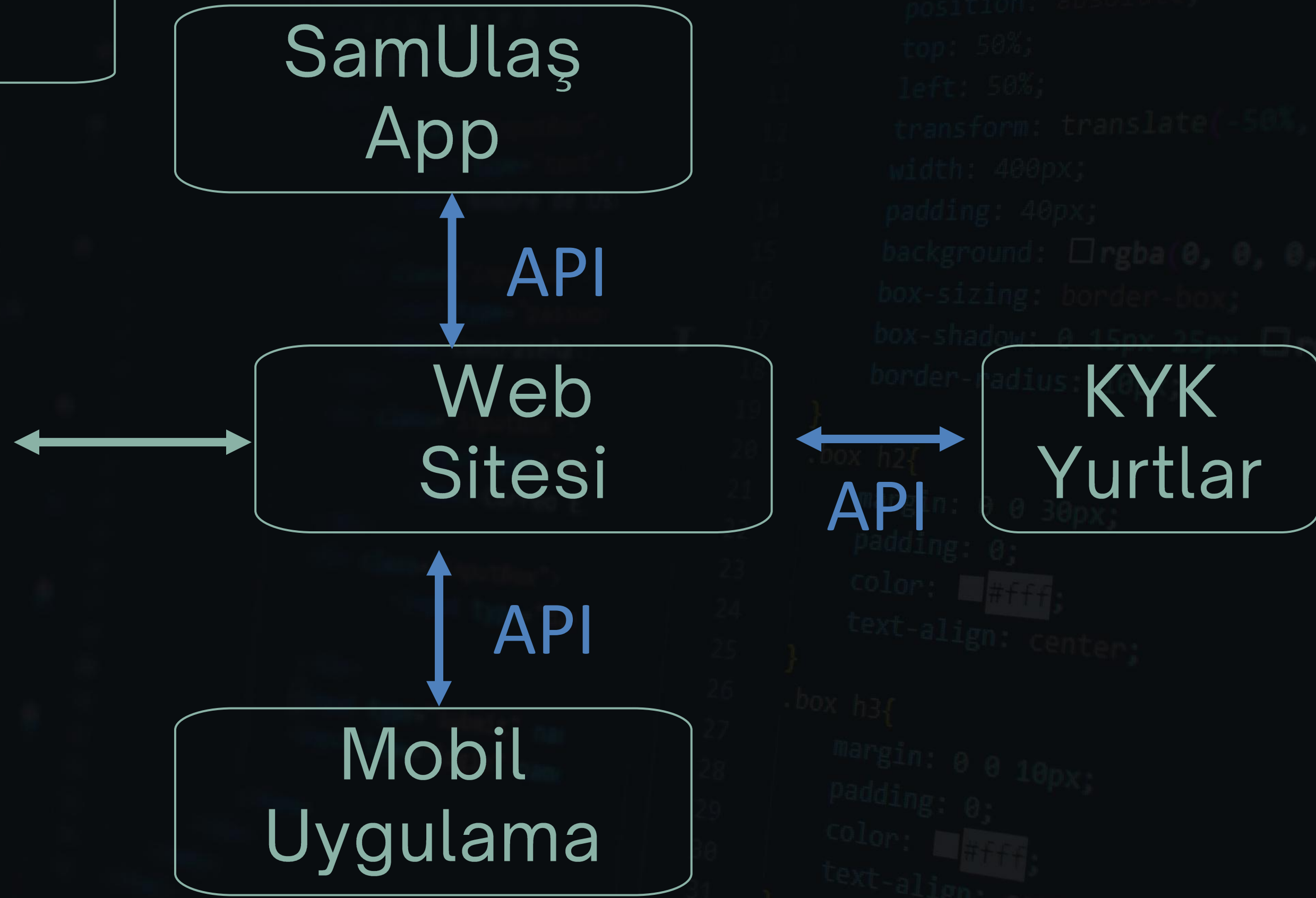
API

Mobil
Uygulama

API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri



API'nin Temel Amacı

DataBase

- Öğrenci bilgileri
- Ders bilgileri
- Notlar
- Devamsızlık bilgileri
- Ders kayıtları
- Sınav tarihleri

SamUlaş
App

API

Web
Sitesi

API

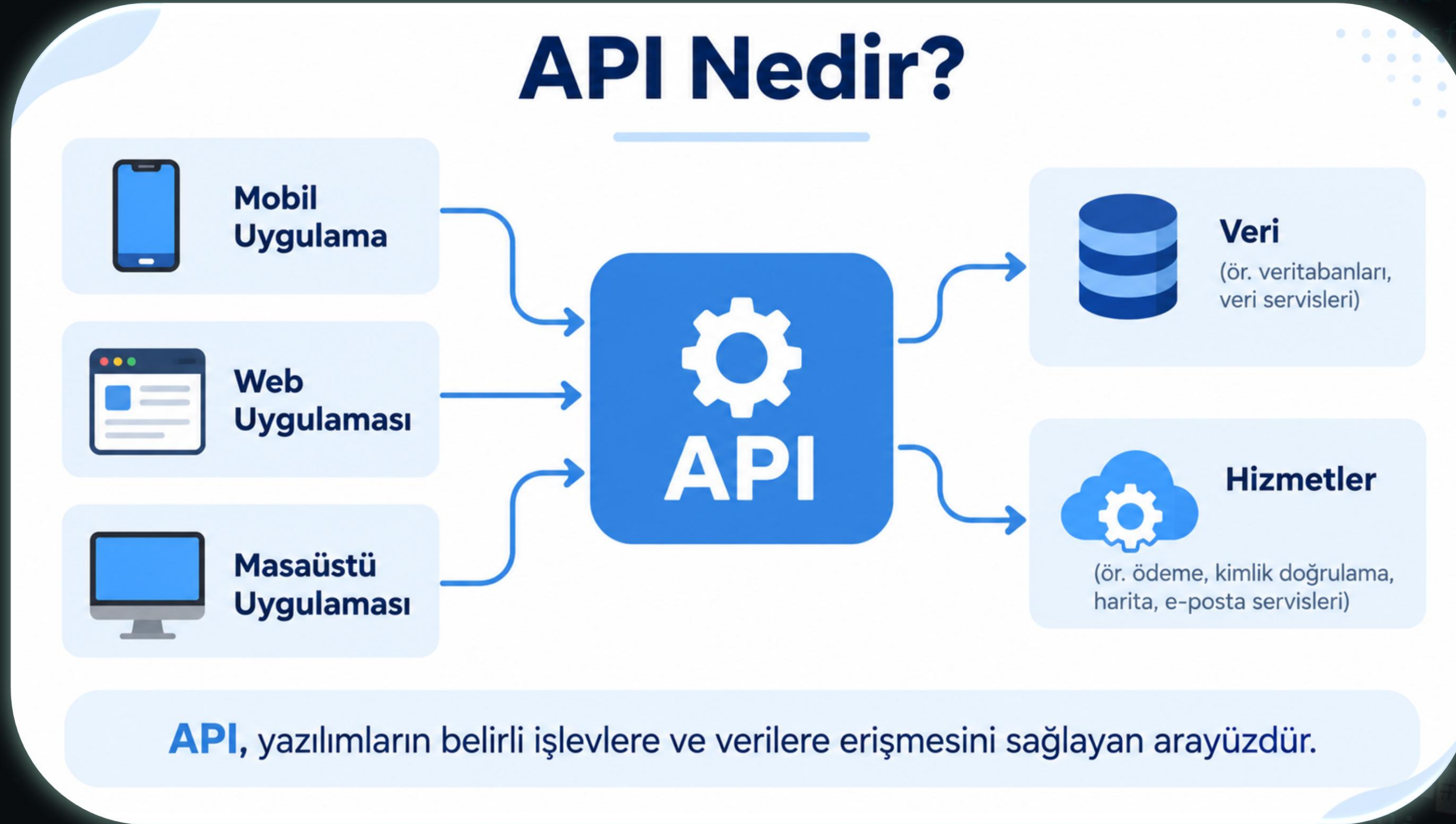
Mobil
Uygulama

KYK
Yurtlar

API

API kullanımında iki yazılımın aynı sistemin parçası olması gerekmez. Farklı ekipler tarafından, farklı teknolojilerle veya farklı zamanlarda geliştirilmiş yazılımlar da API aracılığıyla birbirleriyle çalışabilir.

API'nin Temel Amacı



API Kavramı

Bir API, sunduğu özelliklere bağlı olarak farklı işlemlerin gerçekleştirilmesine olanak tanıyabilir.

Örneğin bir kütüphane sistemi için API üzerinden:

- Kitap bilgilerini görüntülemek
- Bir kitabın mevcut olup olmadığını öğrenmek
- Kitap kaydı oluşturmak
- Bir kitabın bilgilerini güncellemek
- Bir kitap kaydını silmek

gibi işlemler sunulabilir. Burada dikkat edilmesi gereken nokta şudur: **Bir API'nin hangi işlemleri desteklediği, API'yi geliştiren sistemin tasarımına ve izinlerine bağlıdır.**



API Kavramı

Her API bütün işlemleri gerçekleştiremez. Bir API yalnızca veri görüntüleme amacıyla hazırlanmış olabilir. Başka bir API ise veri görüntülemenin yanında kayıt oluşturma veya güncelleme gibi işlemleri de destekleyebilir.

Ayrıca bir API'nin bir işlemi desteklemesi, herkesin bu işlemi sınırsız şekilde yapabileceği anlamına gelmez. Erişim yetkileri ve kullanım kuralları API'nin tasarımına göre belirlenebilir.



API Avantajları

Hazır hizmetlerden faydalanmak

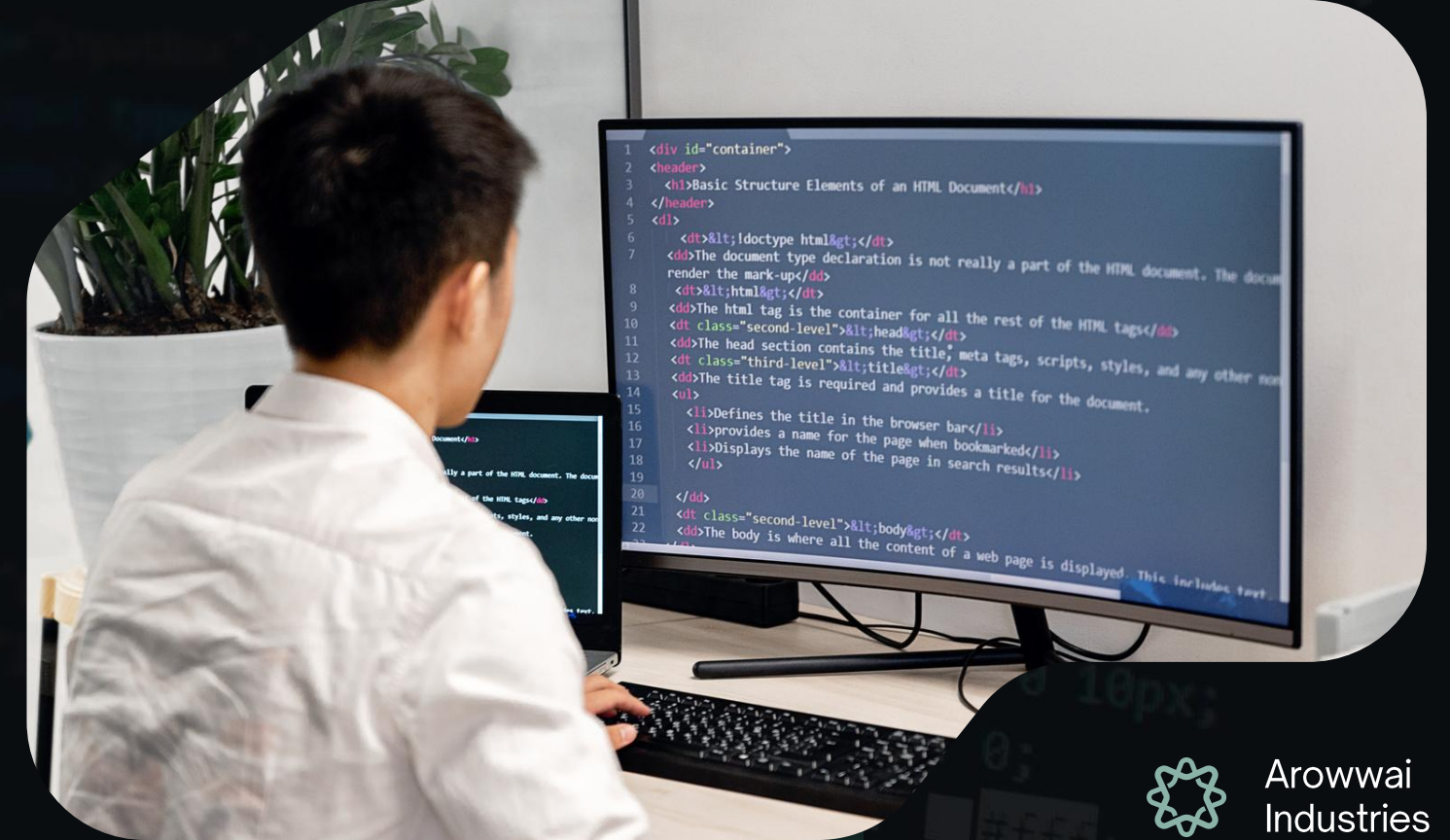
Her yazılımı sıfırdan geliştirmek zorunda kalmayız.

Yazılımların birbirleriyle çalışmasını sağlamak

Bir kurumda birbirinden farklı yazılımlar bulunabilir.

Kod tekrarını azaltmak

Aynı hizmeti birden fazla uygulamada kullanmak



Arowwai
Industries

API Avantajları



Her yazılımı sıfırdan geliştirmek zorunda kalmayız. Örneğin bir e-ticaret uygulaması geliştiriyoruz. Uygulamamızda ödeme alınması gerekiyor.

Ödeme işlemini gerçekleştirmek için:

- Bankacılık sistemleriyle iletişim kurmak
- Kart bilgilerinin işlenmesini sağlamak
- Güvenlik kontrolleri yapmak
- Ödeme sonucunu takip etmek
- Hatalı işlemleri yönetmek

gibi birçok işlem gerekebilir.

Bir ödeme kuruluşu gerekli hizmetleri API üzerinden sunuyorsa e-ticaret uygulaması bu hizmetten yararlanabilir.

Bu, ödeme sisteminin bütün iç yapısını kendimizin geliştirmemiz gerekmediği anlamına gelir.

Hazır hizmetlerden faydalanmak

API Avantajları



Bir kurumda birbirinden farklı yazılımlar bulunabilir.

Örneğin:

- İnsan kaynakları yazılımı
- Muhasebe yazılımı
- Stok takip yazılımı
- E-ticaret sistemi
- Mobil uygulama
- Web uygulaması

Bu sistemlerin tamamı aynı programlama diliyle veya aynı teknolojiyle geliştirilmiş olmayabilir. Bir sistem Python ile, başka bir sistem C# ile, diğer bir sistem Java ile geliştirilmiş olabilir.

API, bu sistemlerin belirli hizmetleri paylaşmasına yardımcı olabilir. Böylece yazılımlar, birbirlerinin bütün kaynak kodunu bilmeden veya aynı programlama dilini kullanmadan etkileşim kurabilir.

Yazılımların birbirleriyle çalışmasını sağlamak

API Avantajları



Bir kurumun öğrenci bilgilerini sağlayan bir sistemi olduğunu düşünelim.

Bu bilgiler farklı uygulamalarda kullanılabilir:

- Web sitesi
- Android uygulaması
- iOS uygulaması
- Öğrenci bilgi ekranı
- Yönetim paneli

Her uygulamanın öğrenci bilgilerini kendi içerisinde ayrı ayrı tutması ve yönetmesi gerekmeyebilir.

Bunun yerine ortak bir API üzerinden aynı hizmete erişim sağlanabilir. Böylece uygulamalar farklı ekranlara ve kullanıcı deneyimlerine sahip olsa bile aynı temel hizmetten faydalanabilir.

Aynı hizmeti birden fazla uygulamada kullanmak

API Avantajları



Bir işlevin birden fazla uygulamada kullanılacağını düşünelim. Örneğin bir kurumun müşteri bilgilerini kontrol eden bir sistemi olsun. Bu kontrolün farklı uygulamalarda tekrar tekrar yazılması yerine, ortak bir hizmet olarak sunulması tercih edilebilir.

Böylece aynı iş mantığı farklı uygulamalarda yeniden yazılmak yerine ortak bir arayüz üzerinden kullanılabilir.

Kod tekrarını azaltmak

API Kullanım Alanları

API Nereelerde Kullanılır?

Daha bağlantılı, daha güçlü uygulamalar



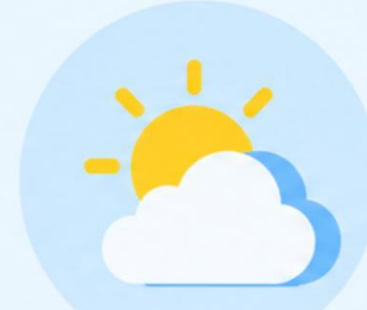
Ödeme Sistemleri

Ödeme işlemleri ve entegrasyon



Harita ve Konum

Konum ve harita hizmetleri



Hava Durumu

Sıcaklık ve tahmin bilgileri

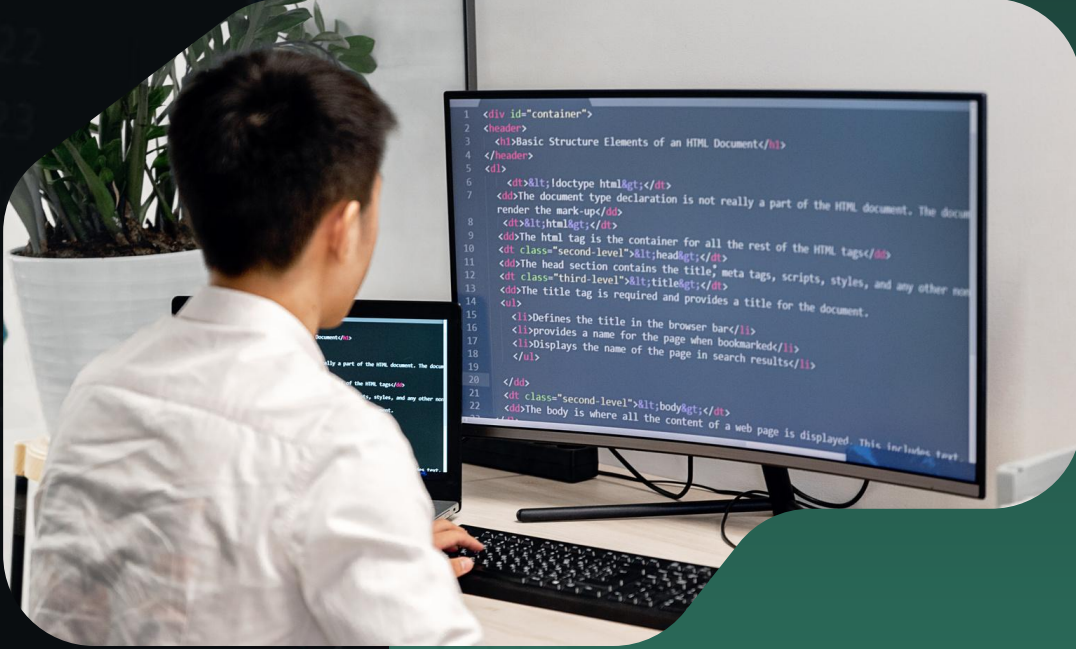


Kurumsal Yazılımlar

Farklı sistemler arası veri paylaşımı

API ile daha fazlası mümkün.

Entegre et • Geliştir • Geleceği inşa et



Günümüzde modern yazılım sistemlerinin neredeyse tamamı API mimarisi üzerine inşa edilmektedir.



API Temel Soru – Cevap



API kullanırken iç sistemin tamamını bilmek gerekir mi?



API Temel Soru – Cevap

API kullanırken iç sistemin tamamını bilmek gerekir mi?

Genellikle hayır. Bir API'nin en önemli faydalarından biri, kullanıcının veya başka bir yazılımın iç sistemin bütün ayrıntılarını bilmek zorunda kalmadan belirli bir hizmeti kullanabilmesidir.



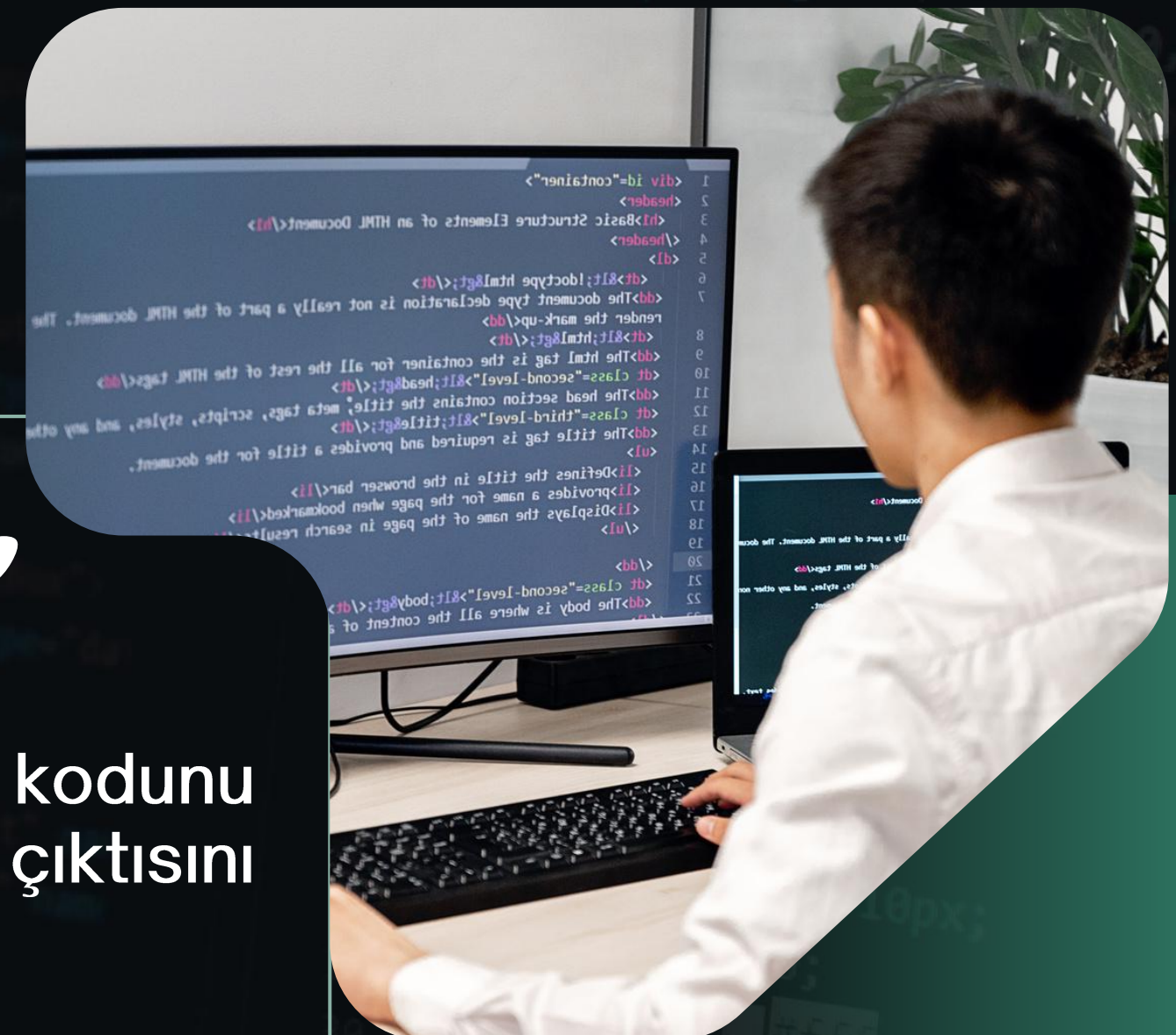
API Temel Soru – Cevap



API, kaynak kodunu paylaşmak anlamına mı gelir?



Hayır. Bir yazılımın API sunması, o yazılımın bütün kaynak kodunu başkalarına açtığı anlamına gelmez. API kaynak kodun çıktısını paylaşır.





API Temel Soru – Cevap



Her API herkese açık midir?

API Temel Soru – Cevap



Her API herkese açık mıdır?

Hayır. API'lerin erişim şekilleri farklı olabilir. Genel olarak API'leri erişim açısından üç grupta düşünebiliriz.

- Herkese açık API'ler
- Kurum içi API'ler
- Özel veya sınırlı erişimli API'ler



API Temel Soru – Cevap



API her zaman ücretsiz midir?

API Temel Soru – Cevap



API her zaman ücretsiz midir?

Hayır. API'lerin kullanım ücretleri, hizmeti sunan kuruluşa ve kullanım koşullarına göre değişebilir.

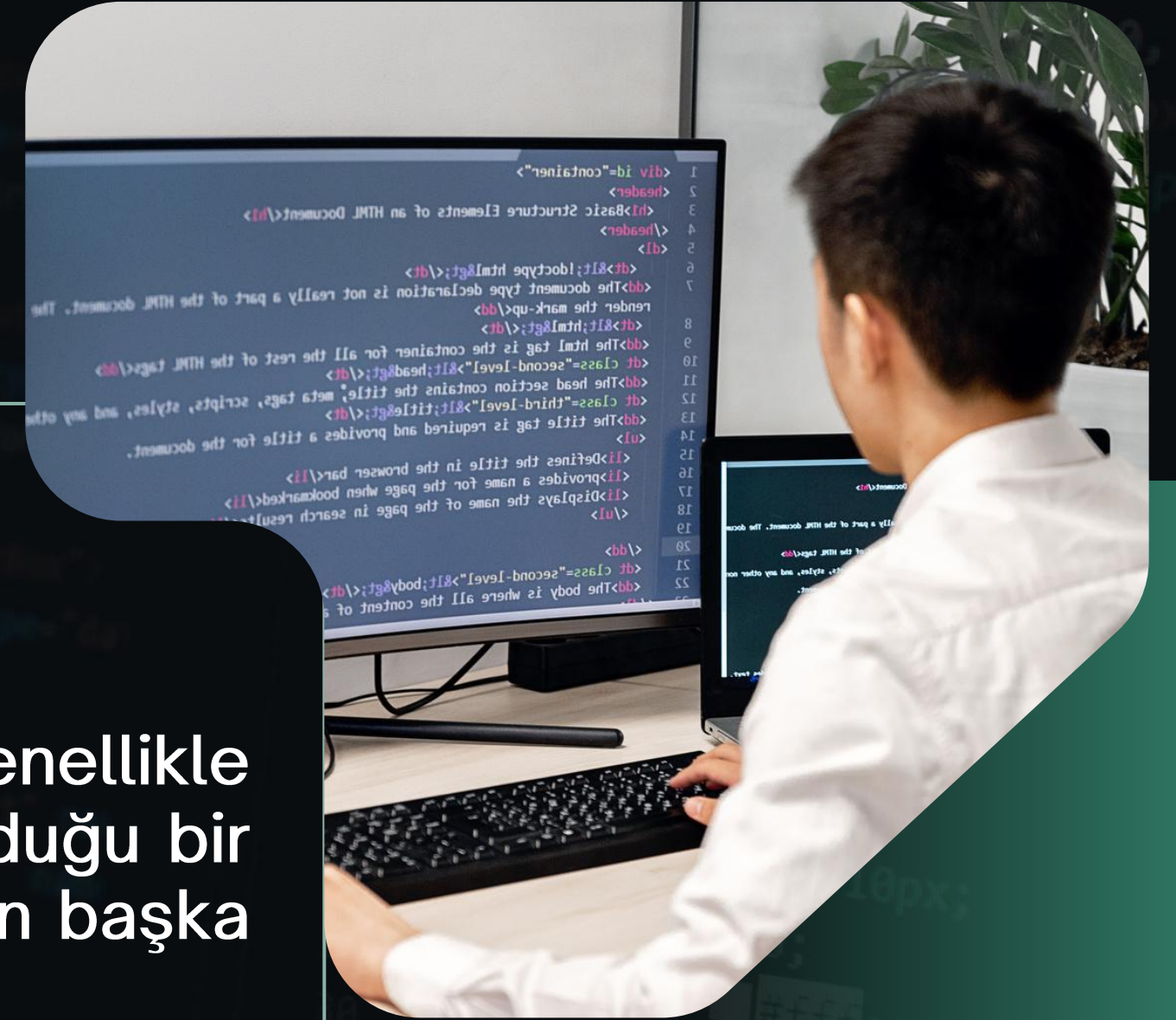


API Temel Soru – Cevap



API ile web sitesi aynı şey midir?

API Temel Soru – Cevap



API ile web sitesi aynı şey midir?

API ile web sitesi aynı şey değildir. Bir web sitesi genellikle insanların tarayıcı üzerinden görüntülediği ve etkileşim kurduğu bir kullanıcı arayüzüne sahiptir. API ise bir hizmetin veya verinin başka yazılımlar tarafından kullanılmasını sağlayan arayüzdür.



***Bu ders boyunca hem API kullanan
hem de API geliştiren bir yazılımcı
perspektifi kazanacağız.***



Temel Kavramlar



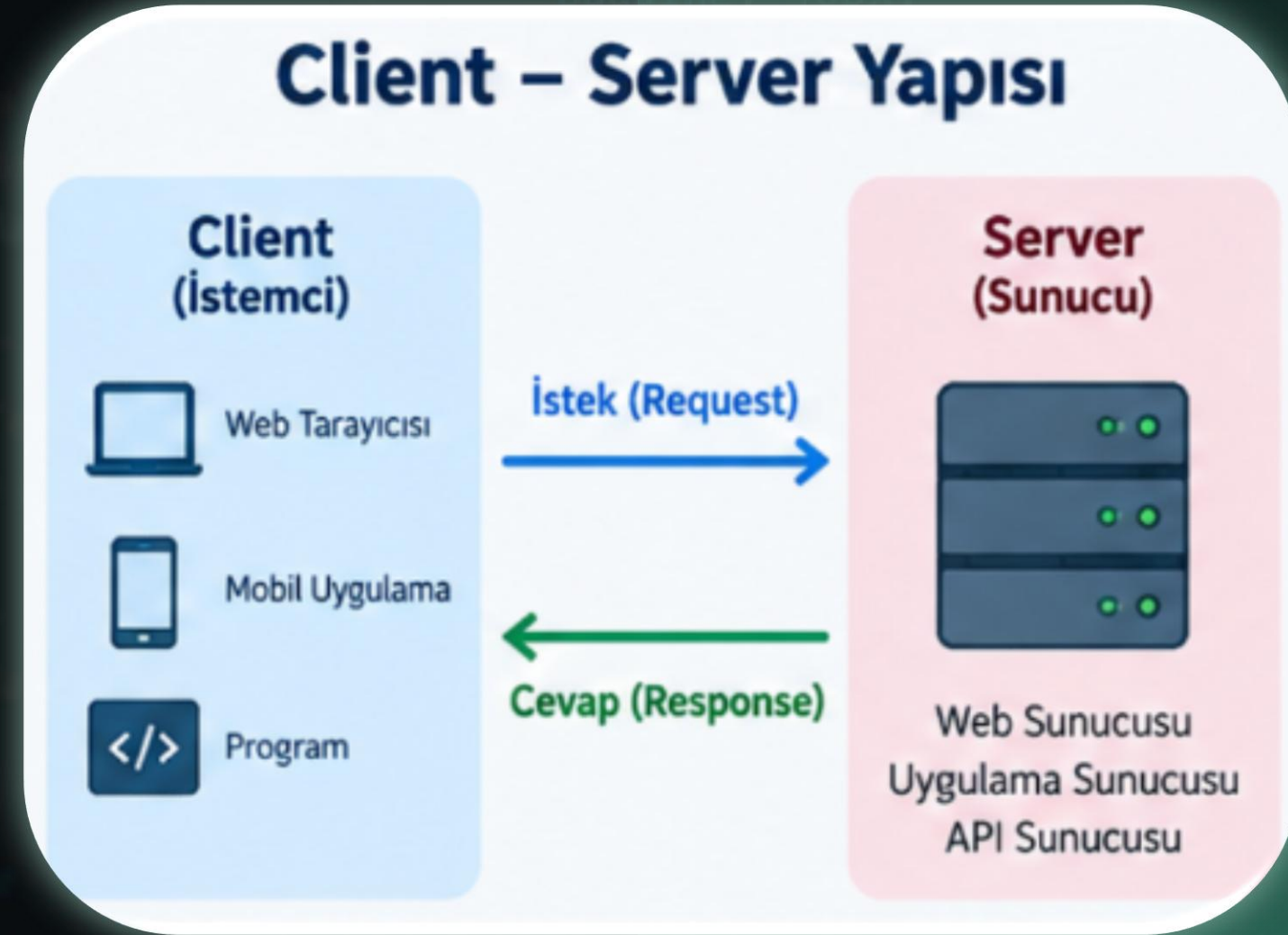
Client-Server Yapısı

İnternet üzerindeki birçok uygulama Client-Server, yani İstemci-Sunucu mantığıyla çalışır.

Burada iki temel taraf vardır:

- Client → İstemci
- Server → Sunucu

En basit haliyle: Client bir hizmet veya bilgi isteyen taraftır. Server ise bu hizmeti veya bilgiyi sağlayan taraftır.



Client-Server Yapısı

Örneğin tarayıcıdan bir web sitesini açtığımızı düşünelim.

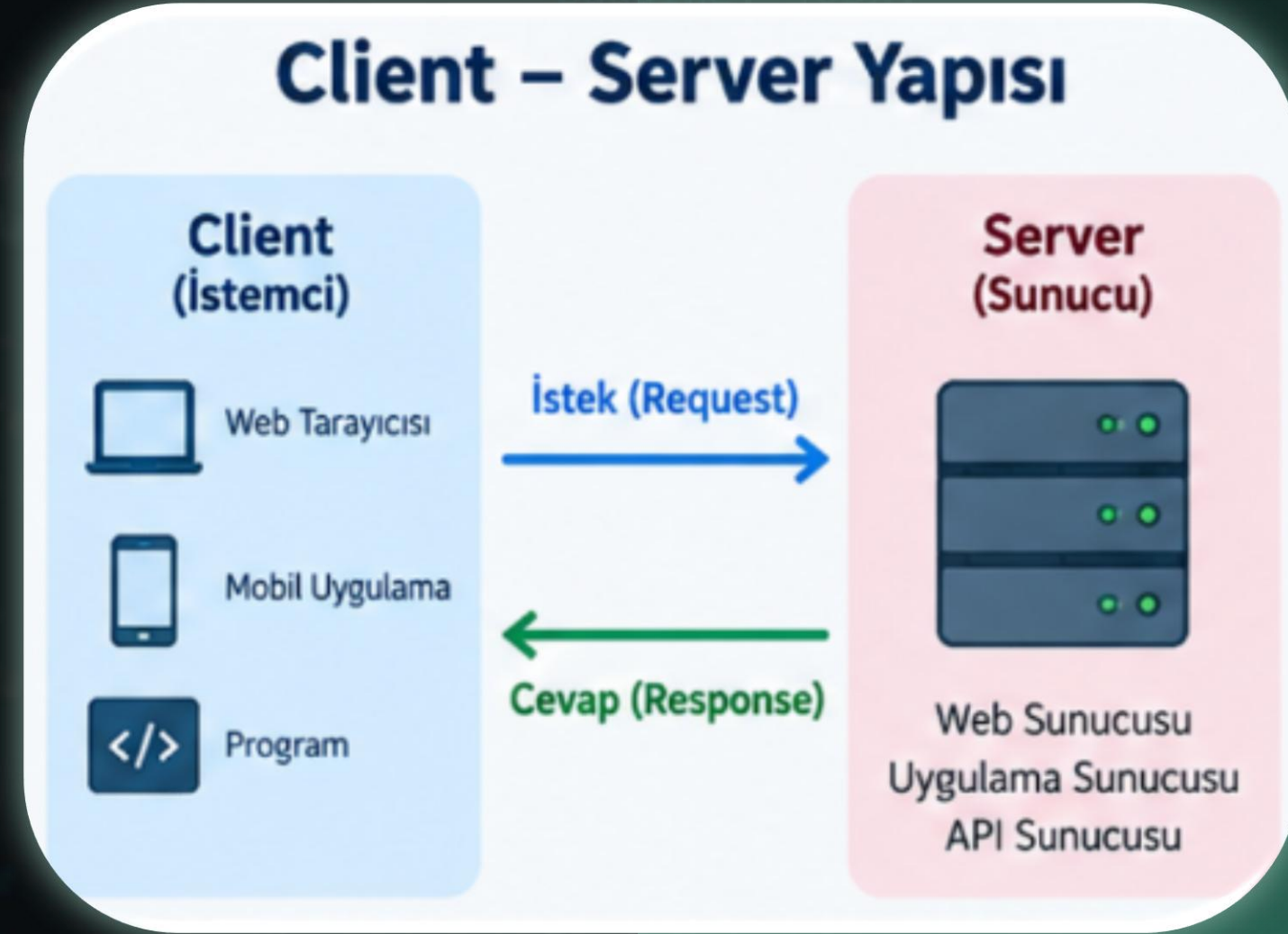
Chrome'a: <https://www.google.com>

yazdığımızda tarayıcımız bir şey istemektedir.

Bu durumda:

- Chrome → Client
- Google'ın sistemi → Server

şeklinde düşünülebilir. Client, server'a ulaşır ve belirli bir istekte bulunur. Server bu isteği değerlendirir ve uygun bir sonuç döndürür.

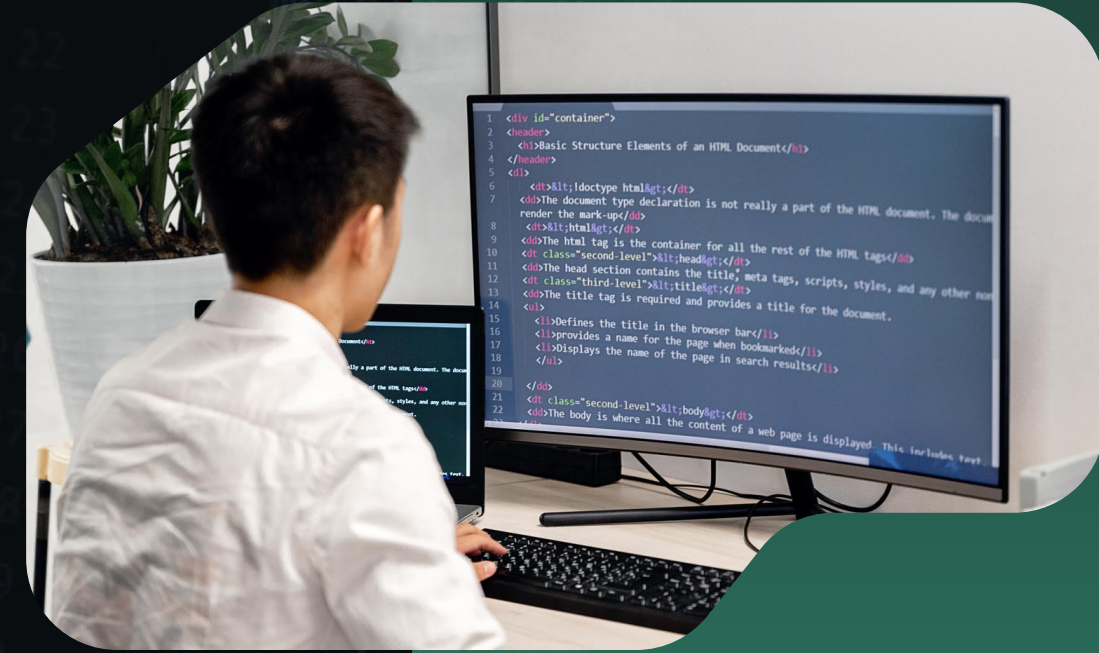


Client (İstemci)

İstemci, başka bir sistemden bir hizmet isteyen yazılımdır. Client mutlaka bir bilgisayar olmak zorunda değildir.

Örneğin client şunlardan biri olabilir:

- Web tarayıcısı
- Mobil uygulama
- Masaüstü uygulaması
- Başka bir sunucu
- Python program
- JavaScript uygulaması



Server (Sunucu)

Sunucu, client tarafından gönderilen talepleri karşılayan sistemdir. Server'ın görevi uygulamaya göre değişebilir.

Bir server:

- Web sayfası gönderebilir.
- Kullanıcı bilgisi gönderebilir.
- Ürün bilgisi gönderebilir.
- Hesaplama yapabilir.
- Dosya gönderebilir.
- Veritabanından bilgi okuyabilir.
- Bir işlemi gerçekleştirebilir.

```
.box{  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  transform: translate(-50%,  
  width: 400px;  
  padding: 40px;  
  background: rgba(0, 0, 0,  
  box-sizing: border-box;  
  box-shadow: 0 15px 35px
```

```
# Use the json module to parse the JSON data  
theJSON = json.loads(response)  
  
# now we can access the contents of the JSON  
if "title" in theJSON["metadata"]:  
  print(theJSON["metadata"] ["title"])  
  
# output the number of events, plus the number of events  
count = theJSON["metadata"] ["count"]  
print(str(count) + " events recorded")  
  
for each event, print the place where it occurred  
for i in theJSON["features"]:  
  print(i["properties"] ["place"])  
  print("\n")
```

```
text-  
  es":  
  ag": >= 4.0;  
  properties"] ["mag", if  
  where at least 1 person reported feeling  
  felt:")  
  es":  
  erties"] ["felt"]  
  if "properties"] ["mag", if  
  (feltReports) + " +
```

Client - Server



Client ve Server kavramlarını yalnızca fiziksel bilgisayar olarak düşünmemeliyiz. Bunlar çoğunlukla bir rolü ifade eder.

Örneğin bir yazılım aynı anda bir yerde server, başka bir yerde client olabilir.

Request (İstek)

Client'in server'a gönderdiği talebe Request denir. Client serverdan ne istediğini request aracılığıyla belirtir.



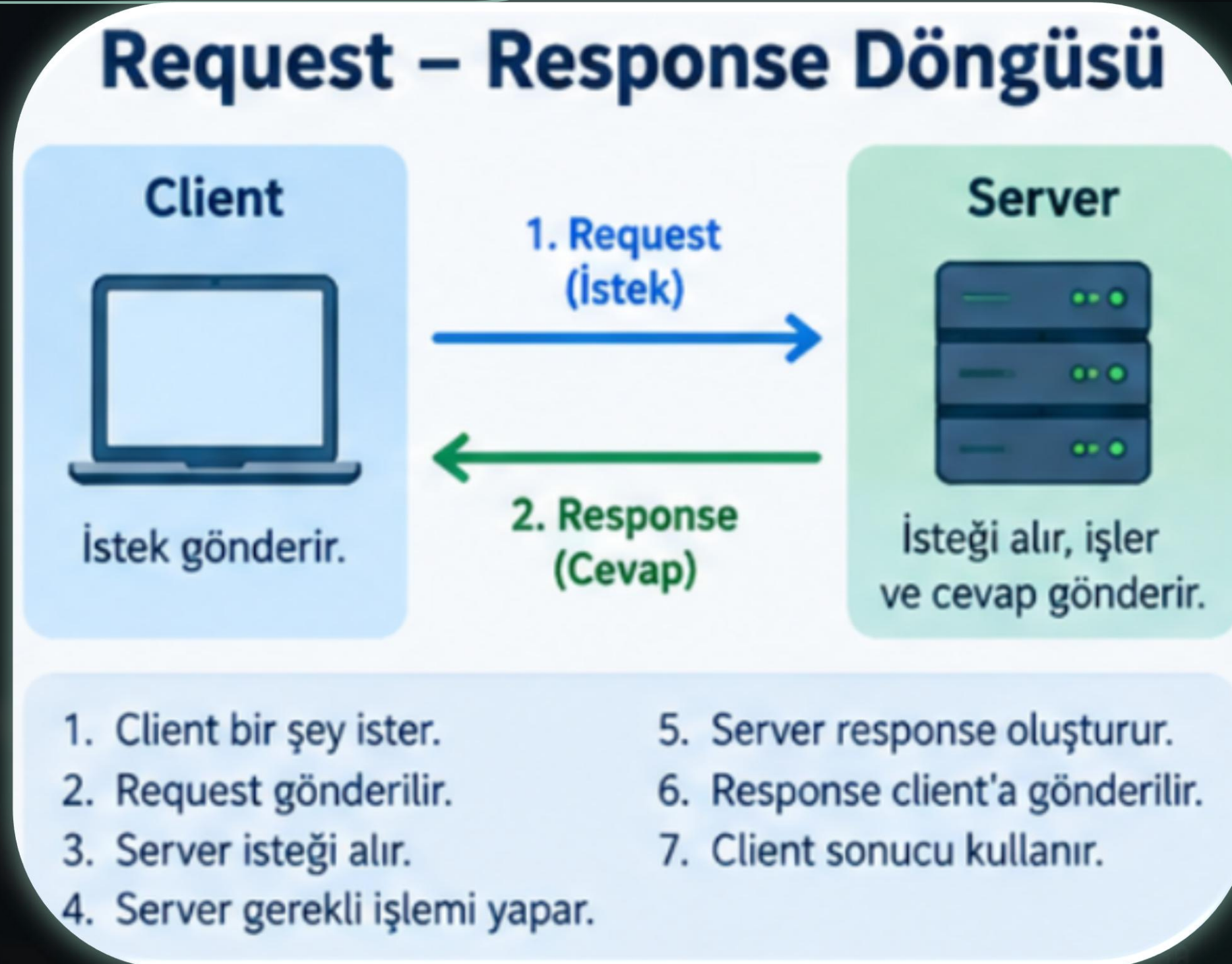
Response (Cevap)



Server'ın client'ın gönderdiği isteğe karşılık verdiği sonuca Response denir.

```
.box{  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%,  
    width: 400px;  
    padding: 40px;  
    background: rgba(0, 0, 0,  
    box-sizing: border-box;  
    box-shadow: 0 15px 25px rgba(  
    border-radius: 10px;  
}  
  
    margin: 0 0 30px;  
    padding: 0;  
    color: #fff;  
    text-align: center;  
}  
  
.box h3{  
    margin: 0 0 10px;  
    padding: 0;  
    color: #fff;  
    text-align: center;  
}  
  
.box input{
```

Request – Response Döngüsü



```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0, 0.1);
  box-sizing: border-box;
  box-shadow: 0 15px 25px rgba(0, 0, 0, 0.1);
  border-radius: 10px;
}

.box h2{
  margin: 0 0 30px;
  padding: 0;
  color: #fff;
  text-align: center;
}

.box h3{
  margin: 0 0 10px;
  padding: 0;
  color: #fff;
  text-align: center;
}

.box input{
```

Request – Response Döngüsü



Client server'a nasıl ulaşır?

Burada yeni bir problem ortaya çıkar. Client'ın bir server'a ulaşabilmesi için server'ın nerede olduğunu bilmesi gerekir.

Gerçek hayatta bir yere gitmek için adres bilmemiz gerekir. İnternette de benzer şekilde bir kaynağa ulaşmak için bir adres kullanılır.

Bu noktada kavramı ortaya çıkar.



Client server'a nasıl ulaşır?

Burada yeni bir problem ortaya çıkar. Client'ın bir server'a ulaşabilmesi için server'ın nerede olduğunu bilmesi gerekir.

Gerçek hayatta bir yere gitmek için adres bilmemiz gerekir. İnternette de benzer şekilde bir kaynağa ulaşmak için bir adres kullanılır.

Bu noktada **URL** kavramı ortaya çıkar.



URL

URL'nin açılımı:

Uniform Resource Locator

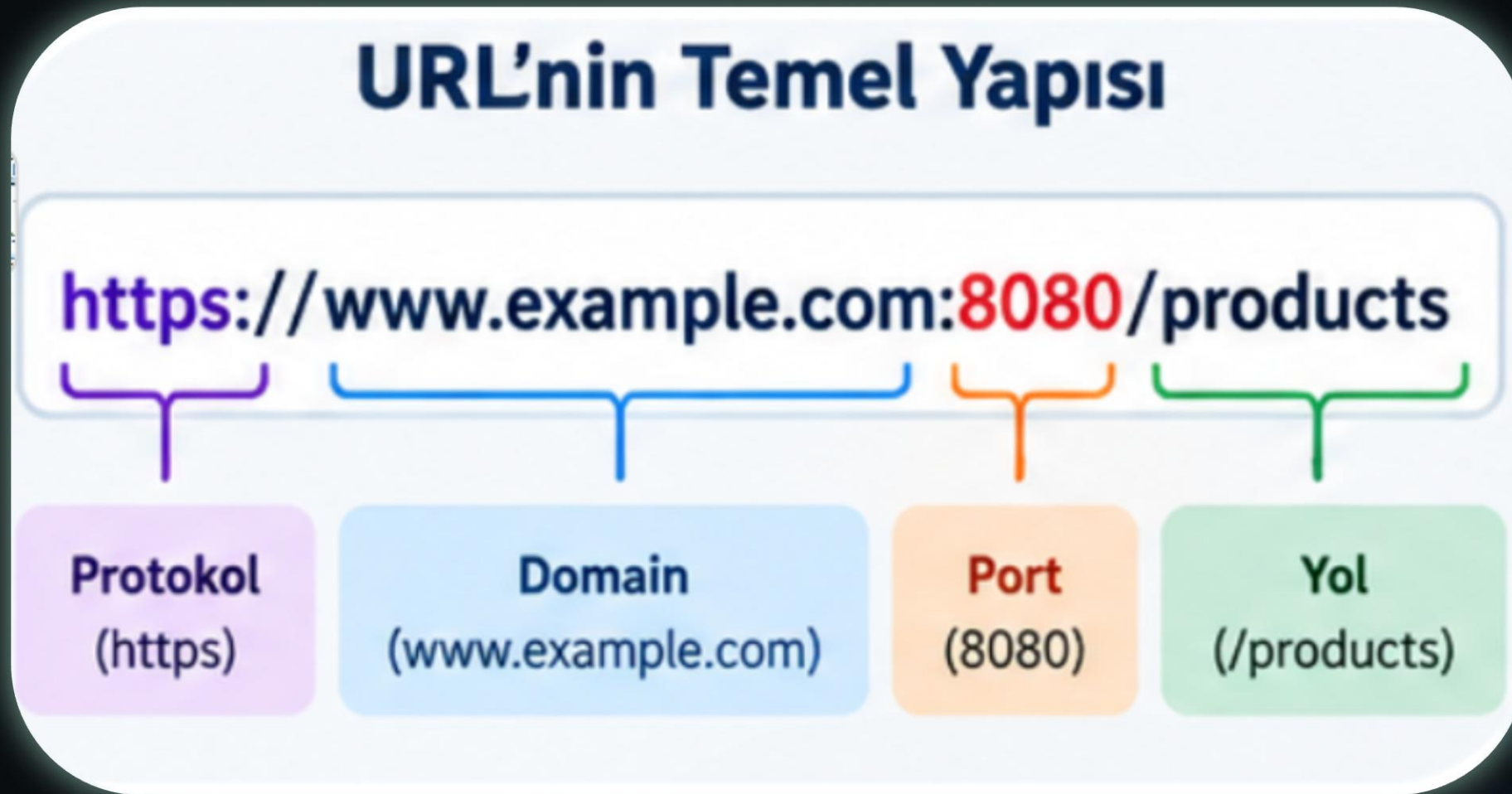
Türkçede genellikle:

Tekdüzen Kaynak Bulucu şeklinde çevrilir.

URL, internetteki bir kaynağın adresidir.
Bu adres bize internette hangi kaynağa ulaşmak istediğimizi belirtir.



Url'nin Temel Yapısı



URL sadece ana adresi değil, belirli bir kaynağın tam adresini ifade eder.



Domain

Domain, Türkçede **alan adı** olarak ifade edilir. Domain internette bir sunucuya ulaşmayı kolaylaştıran okunabilir isimdir.

google.com

youtube.com

samsun.edu.tr

İnternette sistemler birbirlerini sayısal adresler üzerinden bulabilir. Ancak insanların bu tür adresleri ezberlemesi zordur.

Örneğin **192.0.2.10** hatırlamak zor olabilir, bunun yerine **example.com** gibi bir isim kullanmak çok daha kolaydır.

```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%,
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0,
  box-sizing: border-box;
  box-shadow: 0 15px 25px
```

```
# Use the json module to load the JSON data
theJSON = json.loads(data)

# now we can access the contents of the JSON
if "title" in theJSON["metadata"]:
    print(theJSON["metadata"]["title"])

# output the number of events, plus the metadata
count = theJSON["metadata"]
print(str(count) + " events recorded")

for each event, print the place where it occurred
for i in theJSON["features"]:
    print(i["properties"])
    print("-----\n")

# only have a magnitude greater than 4.0
es":
  "mag": 4.0,
  "properties": {"mag": 4.0, "place": "Hawaii"}

# where at least 1 person reported feeling it
for i in theJSON["features"]:
    if i["properties"]["mag"] > 4.0:
        print(i["properties"])
        print("-----\n")
```

Domain

Domain, internet üzerindeki sunucuların, web sitelerinin veya API servislerinin sayısal IP adreslerinin (örn. 192.168.1.1) kolayca hatırlayıp yazabilmesi için kullanılan metin tabanlı isimlendirmedir.

Domain'in IP adresine çevrilmesini sağlayan sisteme DNS ise denir. Ancak DNS ayrı ve daha geniş bir konudur. Bu dersin kapsamında yalnızca domain'in okunabilir alan adı olduğunu bilmemiz yeterlidir.

```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%,
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0,
  box-sizing: border-box;
  box-shadow: 0 15px 25px
```

```
# Use the json module to load the JSON data
theJSON = json.load(open('data.json'))

# now we can access the data
if "title" in theJSON["metadata"]:
  print(theJSON["metadata"]["title"])

# output the number of events, plus the metadata
count = theJSON["metadata"]
print(str(count) + " events recorded")

# for each event, print the place where it occurred
for i in theJSON["features"]:
  print(i["properties"] ["place"])
  print("\n")
```

```
text-
  es":
  ag"] >= 4.0:
  properties"] ["mag", if
  where at least 1 person reported feeling
  felt:")
  es":
  rties"] ["felt"]
  if "properties"] ["mag", if
  (reports) + " +
```

```
.box h3{
  margin
  padding
  color:
  text-align: center;
  .box .input
```

Domain - URL

Domain ve URL aynı şey midir ?



```
.box{
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 400px;
    padding: 40px;
    background: rgba(0, 0, 0, 0.5);
    box-sizing: border-box;
    box-shadow: 0 15px 25px rgba(0, 0, 0, 0.5);
    border-radius: 10px;
}
.box h2{
    margin: 0 0 30px;
    padding: 0;
    color: #fff;
    text-align: center;
}
.box h3{
    margin: 0 0 10px;
    padding: 0;
    color: #fff;
    text-align: center;
}
.box input{
```

Domain - URL

Domain ve URL aynı şey midir ?

Domain URL'nin bir parçası olabilir, ancak URL ile domain aynı şey değildir.



Domain

5. Domain Nedir?

Domain, sunucuya ulaşmayı kolaylaştıran okunabilir bir isimdir.

google.com
youtube.com
samsun.edu.tr
api.example.com

İnsanların sayısal IP adresleri yerine kolay hatırlanabilen isimler kullanmasını sağlar.

~~192.0.2.10~~

Zor hatırlanır



example.com

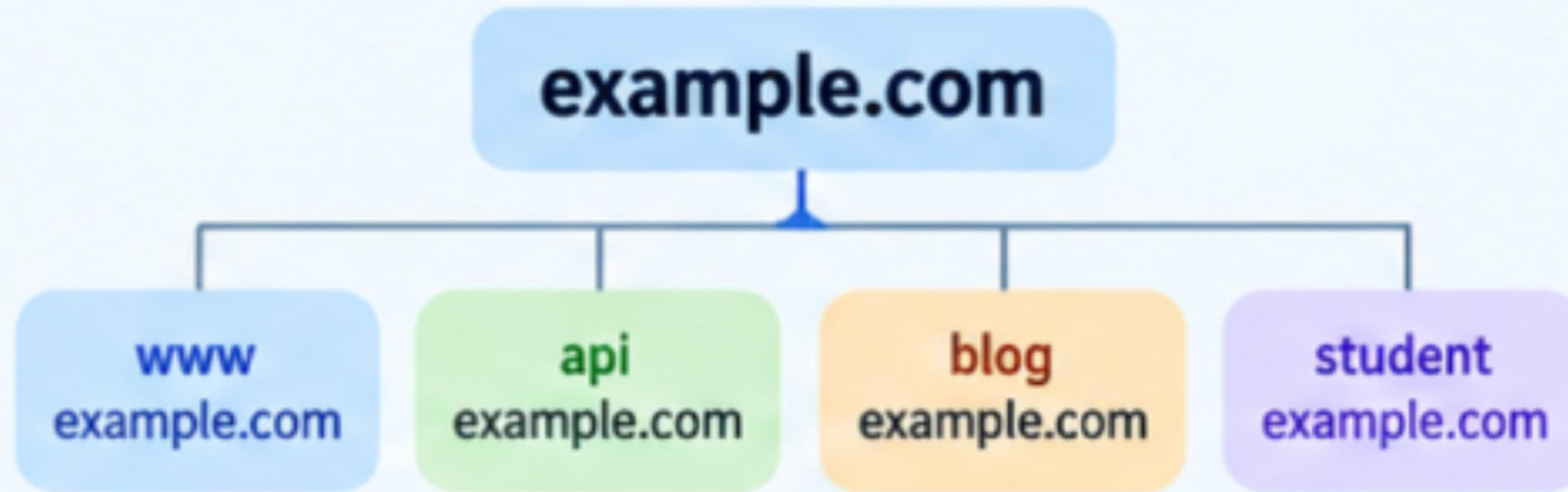


Kolay hatırlanır

Subdomain (Alt Alan Adı)

6. Subdomain Örnekleri

Aynı domain altında farklı alt alan adları olabilir.



Buradaki “www”, “api”, “blog” gibi kısımlar subdomain (alt alan adı) olabilir.

Port

Server üzerinde aynı anda birden fazla uygulama veya hizmet çalışıyor olabilir. Bu durumda hangi hizmetle iletişim kurulacağını belirtmek gerekir. Burada Port kavramı kullanılır.

Port'u basitçe:

Bir cihaz üzerindeki belirli bir hizmete ulaşmayı sağlayan mantıksal kapı numarası olarak düşünebiliriz.



Port

7. Port Nedir?

Port, bir cihaz üzerindeki belirli bir hizmete ulaşmayı sağlayan mantıksal kapı numarasıdır.



Aynı sunucuda birden fazla hizmet çalışabilir. Port numaraları hangi hizmete ulaşılacağını belirtmeye yardımcı olur.

Port

8. Yaygın Port Örnekleri

Port	Yaygın kullanım
80	HTTP
443	HTTPS
3306	MySQL
5432	PostgreSQL
8000	Geliştirme sunucularında sık kullanılır
8080	Alternatif web/geliştirme portu



Bazı protokollerin varsayılan portları vardır. Örneğin
HTTP → 80
HTTPS → 443
Bu yüzden çoğu URL'de port görünmez.



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>

Protokol





Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>

Subdomain



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>

domain



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>



Temel Kavramlar



https://api.furkandurmus.com:8000/products/30

port



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>

Kaynak yolu



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>



Temel Kavramlar



<https://api.furkandurmus.com:8000/products/30>

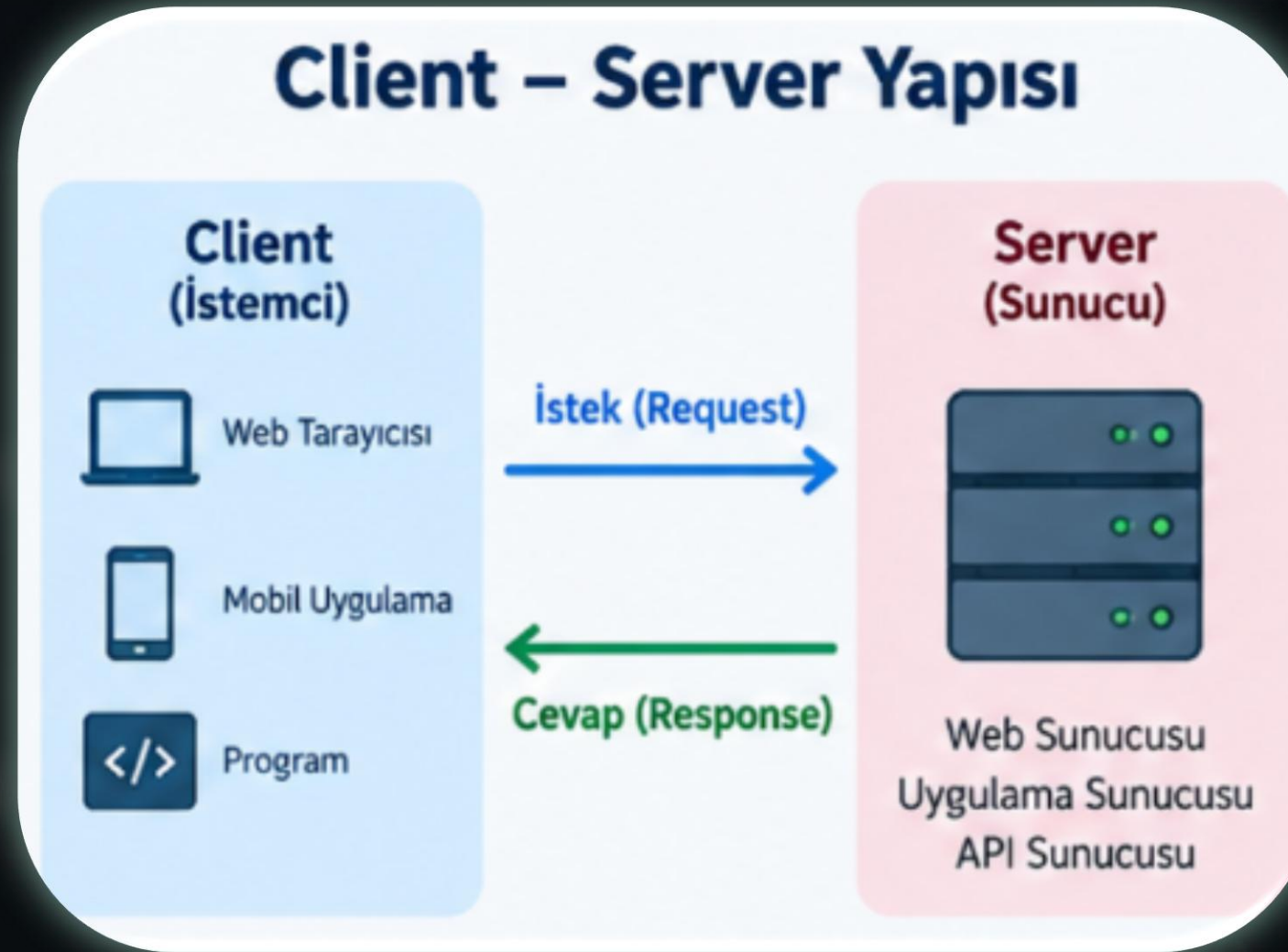
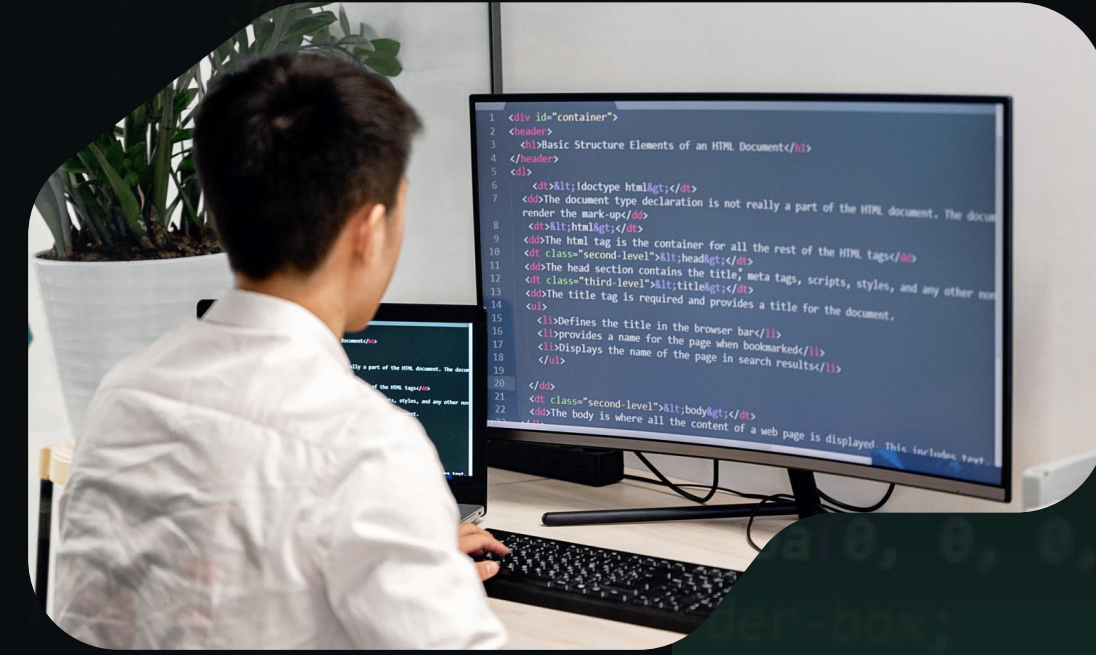
Url

HTTP Temelleri ➔



Giriş

Bir önceki derste şu yapıyı görmüştük:



Şimdi bu yapının içindeki **Request** kısmına biraz daha yakından bakacağız.

Giriş

Bir client server'a istek gönderirken sadece:

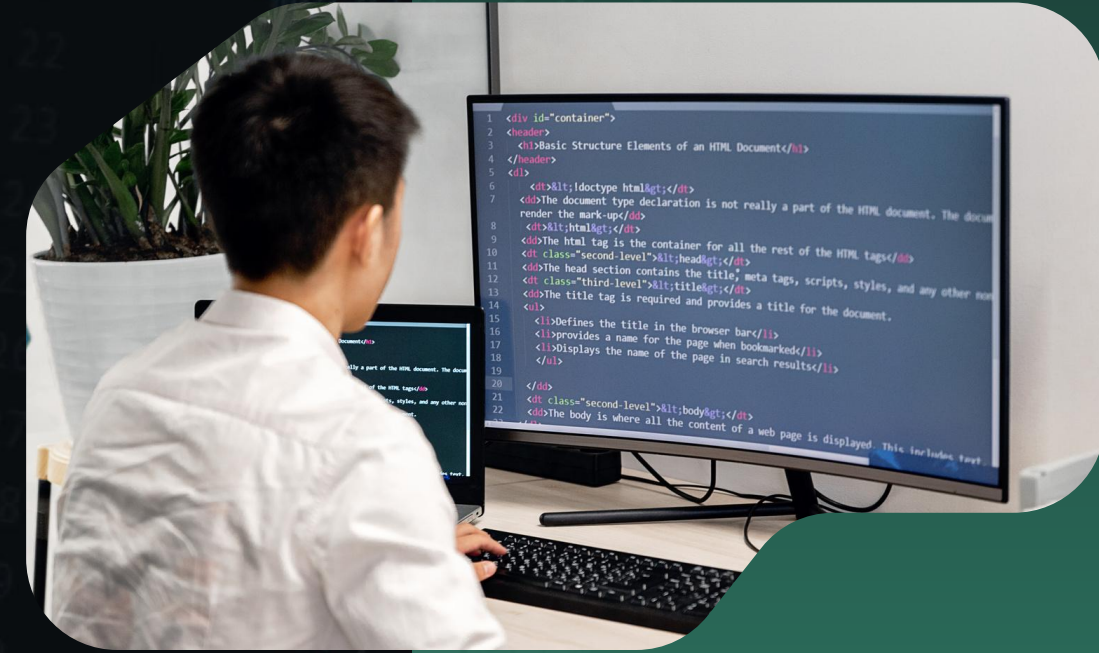
“Bir şey istiyorum.” demez.

Aynı zamanda çoğu durumda:

“Ben bu kaynak üzerinde ne yapmak istiyorum?” bilgisini de belirtir.

İşte HTTP metotları burada devreye girer.

Bildiğiniz HTTP Metotları nelerdir ?



HTTP nedir?

HTTP'nin açılımı:

HyperText Transfer Protocol

Türkçede kabaca:

Hiper Metin Aktarım Protokolü olarak ifade edilir.

HTTP, client ile server arasındaki iletişimde kullanılan kurallardan biridir.

HTTP request ve response iletişiminin nasıl gerçekleşeceğini belirleyen kurallar bütünü olarak düşünülebilir.

```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%,
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0,
  box-sizing: border-box;
  box-shadow: 0 15px 25px
```

```
# Use the json module to load the JSON data
theJSON = json.loads(response)

# now we can access the contents of the JSON
if "title" in theJSON["metadata"]:
    print(theJSON["metadata"]["title"])

# output the number of events, plus the metadata
count = theJSON["metadata"]
print(str(count) + " events recorded")

for each event, print the place where it occurred
for i in theJSON["features"]:
    print(i["properties"])
    print(i["properties"]["place"])

# only have a magnitude greater than 4.0
es":
  "mag": 4.0
  "properties": ["mag", "place", "felt"]

# where at least 1 person reported feeling the
  "properties": ["mag", "place", "felt"]
  "reports": 1
}
```

HTTP nedir?

Bir web sayfası açtığımızda temelde ne gibi amaçlarımız olabilir ?

Temel HTTP Metotları

Client server'a bir request gönderdiğinde, bu request'in bir amacı vardır.

Örneğin client:

- veri almak isteyebilir,
 - yeni veri oluşturmak isteyebilir,
 - mevcut veriyi değiştirmek isteyebilir,
 - verinin bir kısmını değiştirmek isteyebilir,
 - veriyi silmek isteyebilir.
-
- Bu işlemlerin hangi amaçla yapıldığını ifade etmek için HTTP metotları kullanılır.



Temel HTTP Metotları

HTTP Metodu	Temel amaç
GET	Veri almak
POST	Yeni veri oluşturmak
PUT	Mevcut veriyi tamamen güncellemek
PATCH	Mevcut veriyi kısmen güncellemek
DELETE	Veri silmek



GET Metodu

GET kelimesi İngilizce'de almak, elde etmek anlamındadır.

HTTP'de de temel mantığı buna yakındır.

GET metodu server'dan veri almak için kullanılır.

Örneğin:

GET /students

-> Öğrencilerin listesini getir.

GET /students/15

-> 15 numaralı öğrencinin bilgisini getir.

```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%,
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0,
  box-sizing: border-box;
  box-shadow: 0 15px 25px
```

```
# Use the json module to parse the JSON data
theJSON = json.loads(response)

# now we can access the contents of the JSON
if "title" in theJSON["metadata"]:
    print(theJSON["metadata"]["title"])

# output the number of events, plus the metadata
count = theJSON["metadata"]["count"]
print(str(count) + " events recorded")

# for each event, print the place where it occurred
for i in theJSON["features"]:
    print(i["properties"]["place"])
    print("\n")

# only have a magnitude greater than 4.0
es":
  "mag": 4.0,
  "properties": {"mag": 4.0, "place": "Hawaii", "time": 1500000000, "type": "earthquake"}
}

# where at least 1 person reported feeling it
if "properties":
  "felt": 1,
  "properties": {"mag": 4.0, "place": "Hawaii", "time": 1500000000, "type": "earthquake"}
}
```

GET Metodu

GET'in temel amacı veriyi okumaktır.

Yani kavramsal olarak:

GET = Oku / Getir / Görüntüle

şeklinde düşünülebilir.

Tarayıcılardaki tüm standart linkler varsayılan olarak GET isteği atar.

POST Metodu

POST yeni bir kaynak oluşturmak için kullanılır.

Örneğin sisteme yeni öğrenci eklemek istiyoruz.

POST /students

Yeni öğrenci oluştur anlamına gelir. POST metodu yeni bir veri oluşturacağı için, sunucunun yapısına ve beklentisine bağlı olarak bu veriye ait detaylar istek gövdesinde gönderilebilir.



GET - POST

Bir adres hem post hem get olabilir mi ?
Örneğin : POST /rapor ve GET /rapor



GET - POST

Bir adres hem post hem get olabilir mi ?

Örneğin : POST /rapor ve GET /rapor



Bunu bir 'Öneri – Şikâyet Kutusu' gibi düşünebilirsiniz. Adres aynı kutudur; fakat POST kutunun içine yeni bir şikâyet atmak, GET ise kutudaki şikâyetleri toplayıp okumak anlamına gelir.

PUT Metodu

PUT mevcut bir kaynağı **tam olarak güncellemek** için kullanılır.

Örneğin elimizde şu öğrenci olsun:

- id: 5
- name: Ahmet
- surname: Yılmaz
- department: Backend
- phone: 5551112233

Bu öğrencinin kaydını yeni bilgilerle **tamamen değiştirmek** istediğimizi düşünelim.

PUT /students/5 kullanılabilir.

Buradaki temel fikir PUT, mevcut kaynağın yeni haliyle değiştirilmesi mantığına dayanır.



PATCH Metodu

PATCH bir kaynağın yalnızca **belirli bir bölümünü** güncellemek için kullanılır.

Örneğin öğrencinin bilgileri:

- Ad: Ahmet
- Soyad: Yılmaz
- Bölüm: Backend
- Telefon: 5551112233

olsun. Sadece telefon numarasını değiştirmek istiyoruz.

Yeni telefon:5559998877

Bu durumda mantıksal olarak:

PATCH /students/5 kullanılabilir.

Burada bütün öğrenci kaydını değiştirmiyoruz. Yalnızca belirli bir alanı değiştiriyoruz.



DELETE metodu



DELETE var olan bir kaynağı silmek için kullanılır.

Örneğin:

DELETE /students/5

mantıksal olarak 5 numaralı öğrenciyi sil anlamına gelir.



CRUD İşlemleri

CRUD, veritabanı veya kalıcı veri depolayan sistemlerde yapılan 4 temel işlemin baş harflerinden oluşan bir kısaltmadır.





CRUD İşlemleri

HTTP Karşılığı

CREATE
READ
UPDATE
DELETE



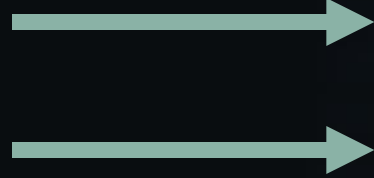


CRUD İşlemleri

HTTP Karşılığı

CREATE
READ
UPDATE
DELETE

POST





CRUD İşlemleri

HTTP Karşılığı

CREATE → **POST**
READ → **GET**
UPDATE →
DELETE





CRUD İşlemleri

HTTP Karşılığı

CREATE	→	POST
READ	→	GET
UPDATE	→	PUT / PATCH
DELETE	→	





CRUD İşlemleri

HTTP Karşılığı

CREATE	→	POST
READ	→	GET
UPDATE	→	PUT / PATCH
DELETE	→	DELETE





```
GET /students
POST /students
GET /students/{id}
PUT /students/{id}
PATCH /students/{id}
DELETE /students/{id}
```

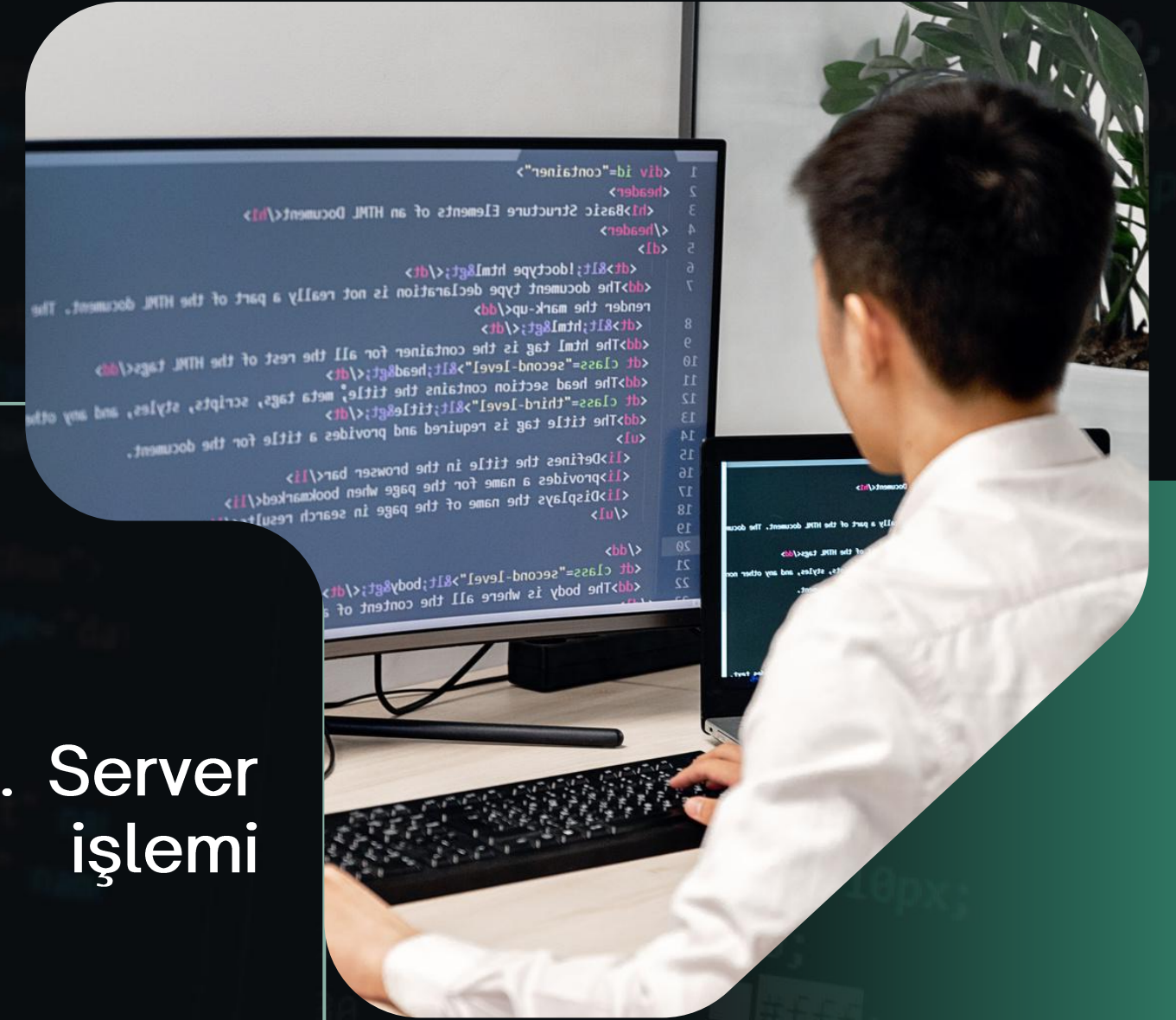




 Öğr. Gör.
Furkan DURMUŞ



HTTP Temel Soru – Cevap



HTTP metotları veritabanı komutları mıdır?

Hayır, bunlar HTTP request'lerinde kullanılan yöntemlerdir. Server gelen request'e göre arka planda veritabanı işlemi gerçekleştirebilir.





HTTP Temel Soru – Cevap



HTTP metodunu kim seçer?

API'yi kullanan client, API'nin belirlediği kurallara uygun şekilde HTTP metodunu kullanır. Ancak hangi işlem için hangi metodun kullanılacağını genellikle API tasarımcısı tarafından belirlenir.



 **Öğr. Gör.**
Furkan DURMUŞ



HTTP Temel Soru – Cevap



Her URL bütün HTTP metotlarını desteklemek zorunda mı?

Her kaynak bütün HTTP metotlarını kullanmak zorunda değildir. API hangi işlemleri destekliyorsa ona göre metotlar tanımlanır.



 Öğr. Gör.
Furkan DURMUŞ



DELETE metodunun anlamı, ilgili kaynağın silinmesini talep etmek şeklindedir.



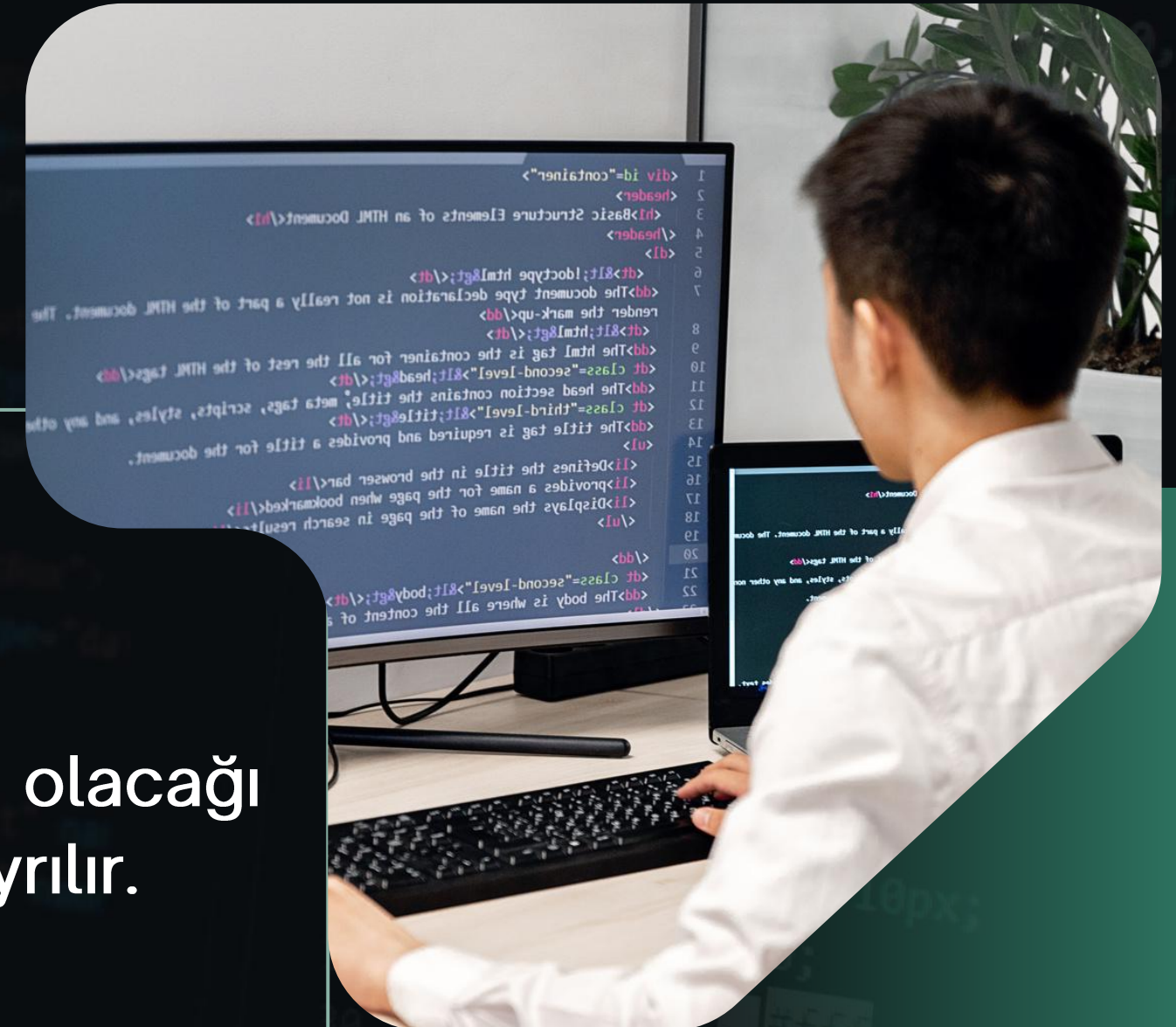
HTTP Temel Soru – Cevap



GET request'i her zaman başarılı olur mu?



HTTP metodunun doğru olması, işlemin mutlaka başarılı olacağı anlamına gelmez. Atılan isteğin sonucu **HTTP Kodları** ile ayrılır.



HTTP Durum Kodları



Giriş

Client server'a bir request gönderdiğinde server bir response döndürür. Bu response yalnızca veri içermez. Aynı zamanda işlemin sonucu hakkında da bilgi verir.

Örneğin server şunlardan birini söylemek isteyebilir:

- İşlem başarılı.
- Yeni kayıt oluşturuldu.
- Kayıt bulunamadı.
- Yetkin yok.
- Gönderdiğin veri hatalı.
- Sunucuda bir hata oluştu.

Bu bilgiyi standart bir şekilde ifade etmek için HTTP Status Code kullanılır.



Giriş

HTTP durum kodu (Status Code), istemcinin gönderdiği bir isteğin sunucu tarafından nasıl sonuçlandırıldığını bildiren 3 haneli standart bir sayısal yanıttır.



Giriş

Bir client server'a şöyle bir istek göndermiş olsun:

GET /students/5

Başarısız bir sonuç aldığımızı düşünelim.

Neden kaynaklı başarısız sonuç almış olabiliriz ?

Giriş

Bir client server'a şöyle bir istek göndermiş olsun:

GET /students/5

Başarısız bir sonuç aldığımızı düşünelim.

Neden kaynaklı başarısız sonuç almış olabiliriz ?

- Öğrenci Kaydı Bulunamamış Olabilir
- Yetkilendirme / Giriş Eksikliği
- Yetki Yetersizliği
- Hatalı Parametre Formatı
- Sunucu İçi Kod / Veritabanı Hatası
- Öğrenci Servisi Ayakta Olmayabilir
- Ağ / Geçit Zaman Aşımı

```
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%,
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0,
  box-sizing: border-box;
  box-shadow: 0 15px 25px
```

```
# Use the json module to parse the response
theJSON = json.loads(response)

# now we can access the contents of the JSON
if "title" in theJSON["metadata"]:
    print(theJSON["metadata"]["title"])

# output the number of events, plus the metadata
count = theJSON["metadata"]["count"]
print(str(count) + " events recorded")

for each event, print the place where it occurred
for i in theJSON["features"]:
    print(i["properties"]["place"])
    print("\n")
```

```
text-
  es":
  ag": >= 4.0;
  properties": ["mag", "place", "felt"]

only have a magnitude greater than 4.0
where at least 1 person reported feeling an
  properties": ["mag", "place", "felt"]
  properties": ["mag", "place", "felt"]
  properties": ["mag", "place", "felt"]
```

200 OK

En sık karşılaştığımız status code'lerden biri 200 kodudur.

Anlamı:
İstek başarılı şekilde işlendi.

Örneğin:
GET /students
isteği gönderildi. Server öğrencileri başarıyla buldu.

Response:200 OK dönecektir.

Yapmak istediğimiz http metodu gerçekleştiyse Status 200 döner.



201 Created

Yeni bir kaynak başarıyla oluşturuldu anlamındadır. Bu kod özellikle POST Metodu ile ilişkilidir.

Örneğin: POST /students
ile yeni öğrenci ekledik.

Server öğrenciyi **başarıyla oluşturduysa** 201 Created döner.



204 No Content

İşlem başarılı ama response içerisinde içerik gönderilmiyor.

Örneğin: DELETE /students/5

işlemi başarıyla gerçekleştirmiş olabilir. Server sadece **204 No Content** döndürebilir.

400 Bad Request

Request hatalı veya geçersiz anlamına gelir.

Örneğin server belirli bir bilgi bekliyor ama client uygun olmayan bir istek gönderiyor olabilir.

Mantıksal olarak "Bu isteği anlayamadım veya uygun bulmadım." şeklinde düşünülebilir.



401 Unauthorized

Kimlik doğrulaması gerekli veya kimlik geçersiz anlamına gelmektedir.

Yani server senin kim olduğunu doğrulayamıyorum demiştir.



403 Forbidden

Kimliğin biliniyor olabilir ama bu işlemi yapmaya yetkin yok manasına gelmektedir.

Bu kod 401 ile çok sık karıştırılır.

404 Not Found

Anlamı istenen kaynak bulunamadı.



500 Internal Server Error

Server request'i işlerken beklenmeyen bir hata oluştu anlamına gelir .

Örneğin server tarafında bir programlama hatası olabilir. Genelde server tarafında kod taraflı bir hata olduğunda bu durum kodunu alırız.



Tesekkürler



www.furkandurmus.com

