Uygulama Programı Arayüzü

Öğr. Gör.
Furkan DURMUŞ

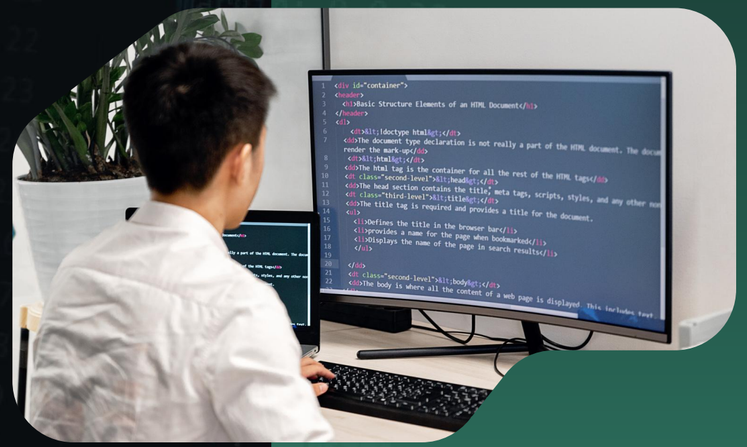
Bu Haftanın Ders Kazanımları

1. Kavram Tanımları
2. RESTful API
3. DEMO

Kavram Tanımları

Neleri Öğreneceğiz ?

REST nedir?
Resource nedir?
URI nedir?
Endpoint nedir?
Bu kavramların birbirinden farkı nedir?
HTTP metotları bu yapının neresinde durur?



REST Nedir ?

REST(Representational State Transfer - Temsili Durum Aktarımı), web tabanlı sistemlerde kaynakların (resource) doğrudan HTTP protokolünün temel prensiplerine göre yönetilmesini sağlayan mimari bir yaklaşımdır; veri alışverişi pratikte genellikle JSON formatıyla gerçekleştirilir.

REST Mimarisinin Farkı

Kural	RPC tarzı	REST tarzı	Neden?
İsim (Noun) kullanılmalı, fiil kullanılmamalı, adreslere metot ile erişilmeli	/kullaniciGetir?id=5 /kullaniciSil/5	GET /kullanicilar/5 DELETE /kullanicilar/5	Ne yapılacağını HTTP metodu söyler, URI sadece kaynağın adıdır.
Çoğul isimlendirme tercih edilir	/kullanici/5	/kullanicilar/5	Kaynak bir koleksiyonun parçası olarak düşünülür.
İlişkiler hiyerarşik modellenir	/kullanici5Siparisleri	/kullanicilar/5/siparisler	5 numaralı kullanıcının siparişleri mantıksal alt kaynaktır.
Küçük harf ve tire (-) kullanılır	/YeniKullanicilar /yeni_kullanicilar	/yeni-kullanicilar	URI'lar büyük/küçük harfe duyarlı olabilir; okunabilirlik için tire standarttır.

REST Çalışma İlkeleri

Temel Çalışma İlkeleri :

- **Kaynak Tabanlı** : Veriler "kaynak" (resource) olarak modellenir ve benzersiz bir URI ile tanımlanır (örneğin /api/v1/users/42).
- **HTTP Metotları** : Kaynaklar üzerindeki işlemler için HTTP eylemleri kullanılır.
- **Durumsuzluk** : Sunucu istemcinin geçmiş oturum bilgisini saklamaz. Her istek, sunucunun o işlemi gerçekleştirebilmesi için gereken tüm kimlik doğrulama ve veri yükünü kendisi taşır.
- **Önbelleklenebilir** : Yanıtlar, HTTP başlıkları üzerinden istemci veya ara sunucular tarafından önbelleğe alınabilir.
- **JSON Ağırlıklı Veri** : XML, HTML veya metin desteklense de endüstri standardı olarak neredeyse tamamen JSON kullanılır.

Resource Nedir ?

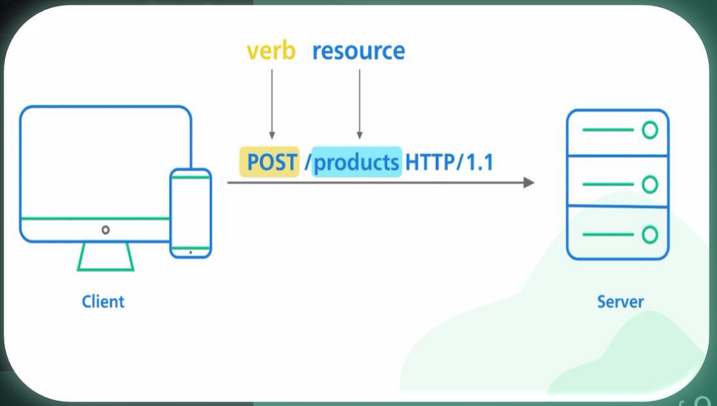
API üzerinden erişilebilen veya üzerinde işlem yapılabilen anlamlı bir varlıktır. REST dünyasında sunucu tarafında yönettiğin, üzerinde işlem yapılabilecek her bir mantıksal varlığa ya da bilgi parçasına "Resource" denir.

Resource

Veritabanında kaynak User tablosunda ikili (binary) formatta duruyor olabilir; ancak istemciye bu kaynak JSON, XML ya da HTML formatında sunulur.

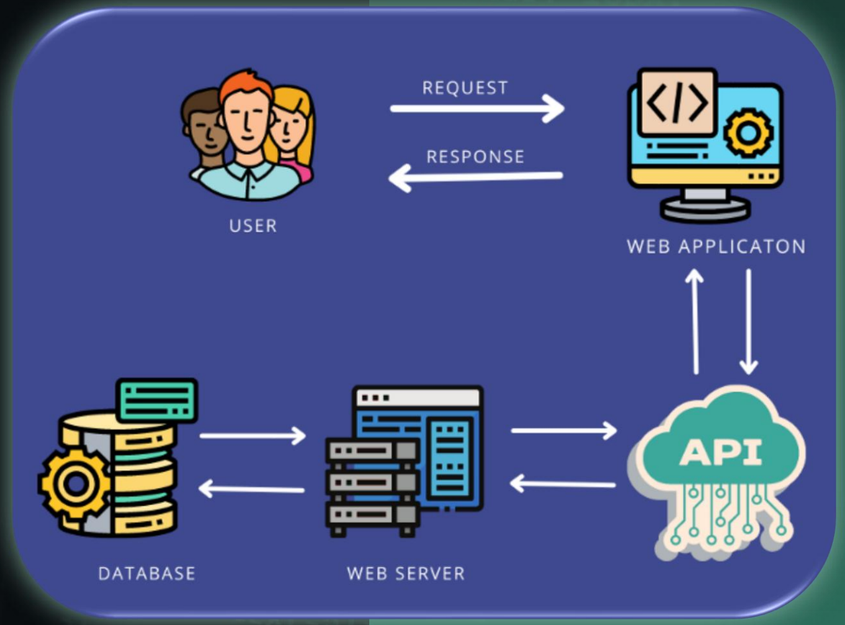
REST'teki Representational State Transfer adı buradan gelir: Kaynağın o anki durumunun bir temsilini (çoğunlukla JSON halini) transfer edersin.

Resource, fiziksel bir dosya veya JSON metni değildir. Resource, sistemdeki **soyut kavramın / varlığın kendisidir**.



Resource

Bir heykeltıraş düşünün;
Kafasındaki fikir veya gerçek insan -> Resource
Onun kilden yaptığı büstü veya kâğıda çizdiği resmi -> Representation (Temsili).



REST – Resource İlişkisi

Resource: Sunucuda bir kaynak var (örn. Kullanıcı Ahmet).

State (Durum): Bu Ahmet'in şu anki durumu nedir? (Bakiye: 100 TL, Durum: Aktif).

Representation (Temsil): Sunucu bu durumu alır, istemcinin anlayacağı bir temsile (örneğin JSON metnine) dönüştürür.

Transfer: Bu temsili ağ üzerinden HTTP ile istemciye aktarır.

Resource

Tekil veya Çoğul (Koleksiyon) olabilir:

/urunler →Ürünler koleksiyonu (Resource)

/urunler/42 →42 ID'li tekil ürün (Resource)

Resource isimleri neden önemli?

API kullanan kişi adrese baktığında neyle çalıştığını anlayabilmelidir.

URI Nedir ?

Bir kaynağa internet üzerinden erişebilmek için ona evrensel ve benzersiz bir kimlik/adres verilmesi gerekir. Bu adrese URI denir.

İnternetteki her kaynağın kendine ait bir URI (Uniform Resource Identifier - Tekdüzen Kaynak Tanımlayıcısı)'ı vardır.

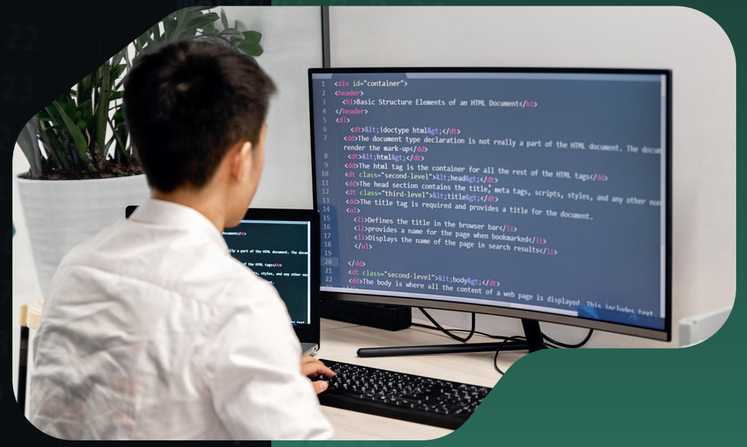
Kısaca özetlemek gerekirse: Resource "ulaşmak istediğin şey", URI ise "onun nerede olduğunu gösteren adres"tir.

URL – URI

Her URL bir URI'dir, ancak her URI bir URL değildir.

URI : Bir kaynağı (dosya, kişi, kitap, veri) internette veya sistemde benzersiz olarak tanımlayan (kimliğini belirten) genel terimdir. Kaynağın nerede olduğunu veya ona nasıl ulaşılacağını söylemek zorunda değildir; sadece onun kim olduğunu belirtmesi yeterlidir.

URL : URI'nin en yaygın alt türüdür. Bir kaynağı sadece tanımlamakla kalmaz, nerede olduğunu ve ona hangi protokolle (nasıl) erişileceğini de açıkça belirtir.



URI hangi ev olduğunu söylerken, url o eve nasıl gidileceğini ve nasıl girileceği söyler.

Kavram Tanımları

URL – URI

users/5 URI

https://data.com/users/5 URL

Endpoint Nedir ?

$$\text{Endpoint} = \text{HTTP Metodu} + \text{URI}$$

Bir API'nin belirli bir işlevine veya kaynağına erişmek için kullanılan erişim noktasıdır.

Basitçe söylemek gerekirse: Sunucunun dış dünyaya sunduğu dijital kapılardır. Her kapının (endpoint'in) arkasında farklı bir veri veya görev bekler.

Endpoint

Endpoint = HTTP Metodu + URI

HTTP Metodu	Yol (Path)	Bu Endpoint Ne Yapar?
GET	/users	Tüm kullanıcıların listesini getirir.
POST	/users	Sisteme yeni bir kullanıcı ekler.
GET	/users/42	42 ID'li kullanıcının detayını getirir.
PUT / PATCH	/users/42	42 ID'li kullanıcının bilgilerini günceller.
DELETE	/users/42	42 ID'li kullanıcıyı sistemden siler.

Kavram Tanımları

Resource – URI – Endpoint- URL

Student — Resource

/students/5 — URI

GET /students/5 — Endpoint

https://data.com/students/5 — URL

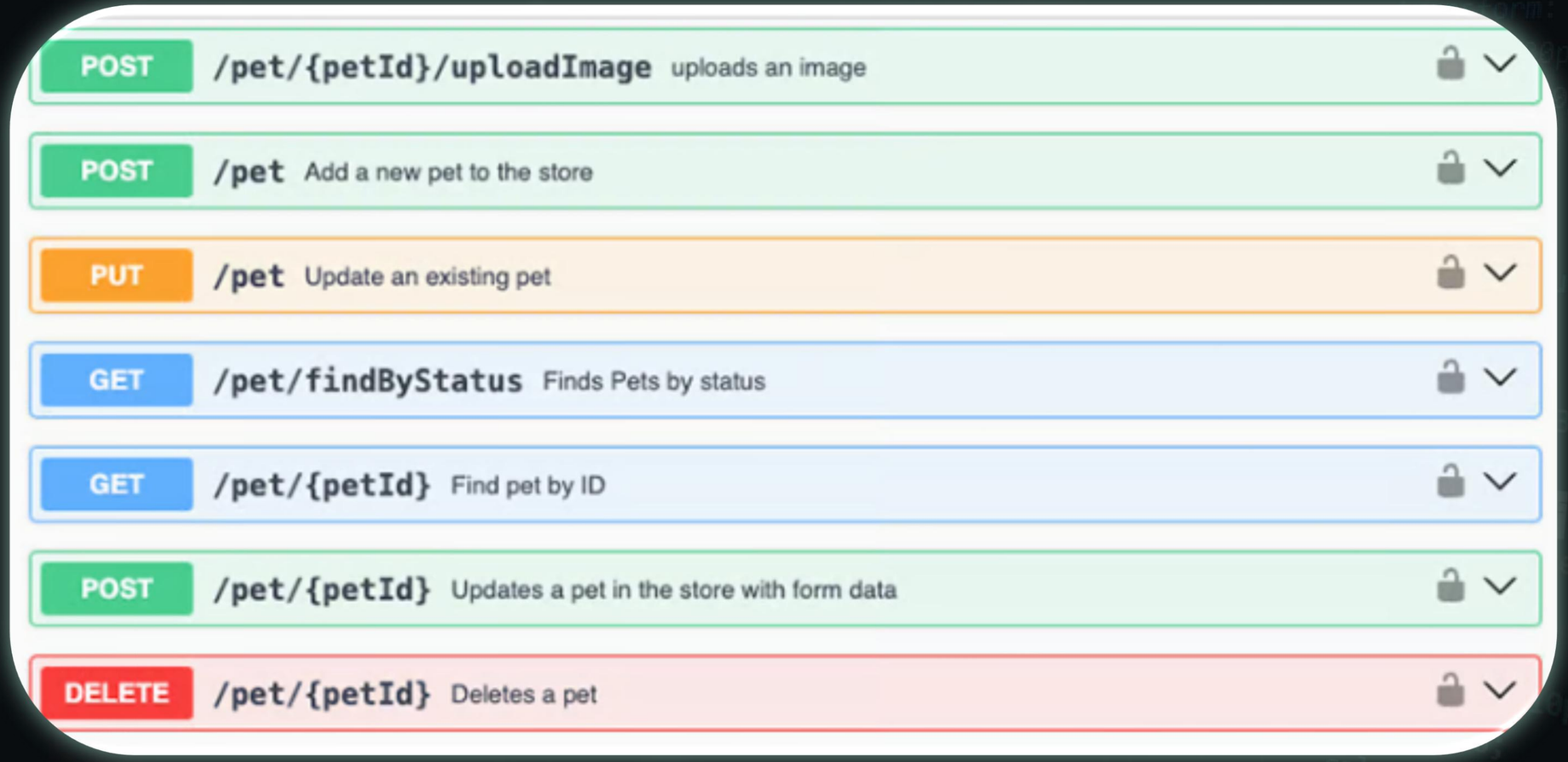
Base URL – URI – URL

URL

https://data.com/students/5

Base URL URI

API Tasarımı Örnek Görünüm

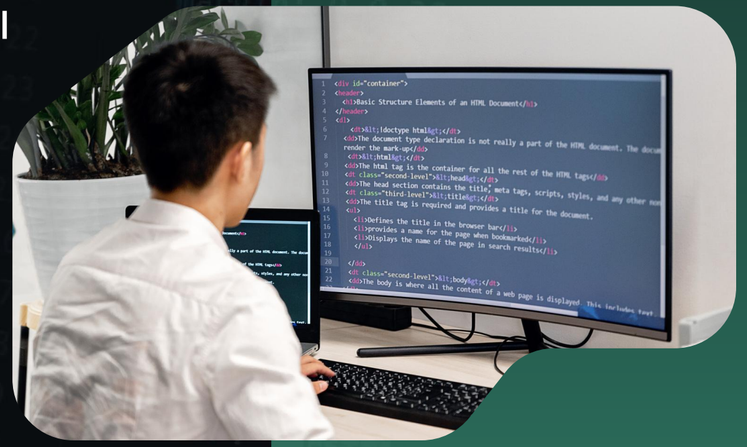


RESTful API

Giriş

REST prensiplerine uygun şekilde tasarlanmış API'lere RESTful API denir.

RESTful API Resource'ları anlamlı URI'lerle ifade eden ve HTTP'nin yapısını tutarlı biçimde kullanan API'dir.



REST ile RESTful API farkı

REST bir mimari kural setidir; RESTful ise bu kurallara tam olarak uyan sistemleri niteleyen sıfattır.

RESTful API Tasarımının Temel Amacı

RESTful tasarımın amacı API'yi:

- anlaşılır,
- tutarlı,
- tahmin edilebilir,
- geliştirilebilir,
- farklı client'lar tarafından kolay kullanılabilir

hale getirmektir.

RESTful API Tasarımının Temel Amacı

Örneğin bir API'nin dokümantasyonunu ilk defa gören kişi

/users
/users
/5/products
/products/10

gibi yapıları gördüğünde çalıştığı API'ı kolayca anlayabilmelidir.

API Tasarımı Neden Önemlidir?

/api1
/a12
/x
/processABC

İyi bir API tasarımında endpoint'ler mümkün olduğunca kendi anlamını anlatmalıdır.

Tasarım

RESTful api isimlerimdirmeleri resource odaklıdır.

/products
/users/2
/post/3/comments
/user/5/orders/44

.
.
.
.

Tasarım

RESTful API isimlendirmelerinde fiil kullanılmamalıdır.

URI'nin içinde mümkün olduğunca:

- Get
- Create
- Delete
- Update

gibi işlemleri ifade eden fiiller kullanılmaz. Çünkü işlemi zaten HTTP metodu ifade eder.

/getUsers
/createUser
/updateUser
/deleteUser

⟷ /users

istenmeyen isimlendirme örnekleridir.

Tasarım

RESTful tasarımda resource'lar genellikle isimlerle ifade edilir.

/users
/products
/orders
/students
/courses

Tasarım

Yaygın REST API tasarım pratiğinde resource isimleri çoğul kullanılır.

/users
/products
/orders
/students
/courses

Tasarım

Tutarlılık RESTful tasarımda çok önemlidir.

/users
/product
/allOrders
/student_list

Tasarım

Anlamlı olması gerekir ama gereksiz derecede uzun olmamalıdır.

/application/products/information/product-items/10

↕

/products/10

Tasarım

URl'lerde küçük harf kullanmak iyi bir pratiktir.

/users
/products
/orders
/students
/user-profiles

Tasarım

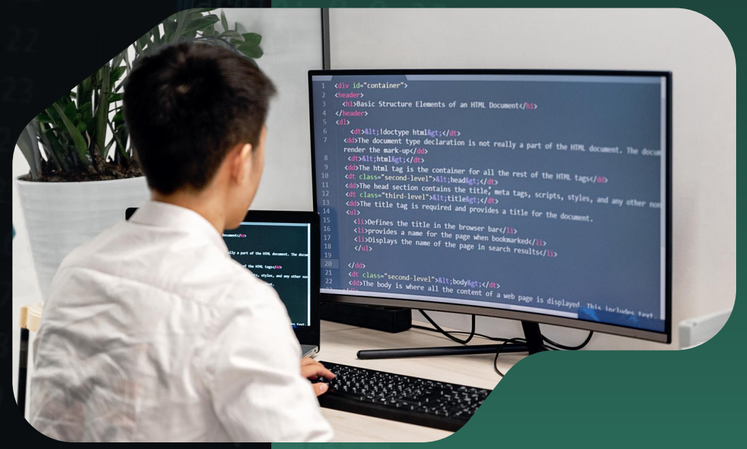
Resource iki kelimeden oluşuyorsa aralarına – koyulur. Snake_case de kullanılabilir fakat yaygın kullanım kebab-case kullanımıdır.

/order-items
/payment-methods
/user-profiles

RESTful Olmanın 6 Temel Kuralı

1 - İstemci - Sunucu Ayrımı :

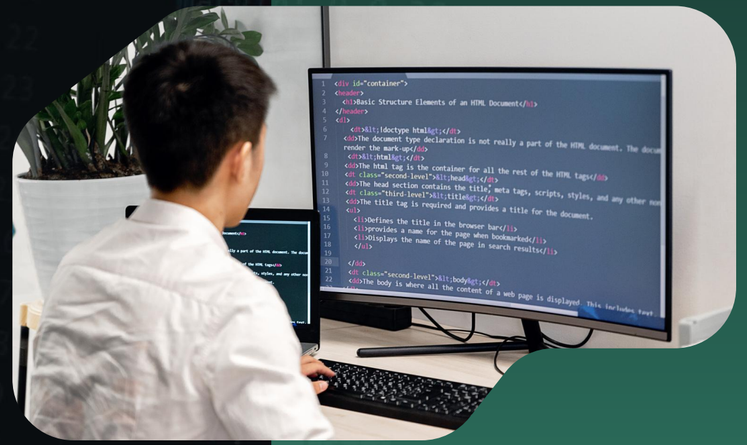
Ön yüz (mobil uygulama, web sitesi) ile arka yüz (veritabanı, iş mantığı) birbirinden tamamen bağımsız olmalıdır. Ön yüzün teknolojisi değiştiğinde arka yüz bundan etkilenmemelidir.



RESTful Olmanın 6 Temel Kuralı

2 - Durumsuzluk (Statelessness):

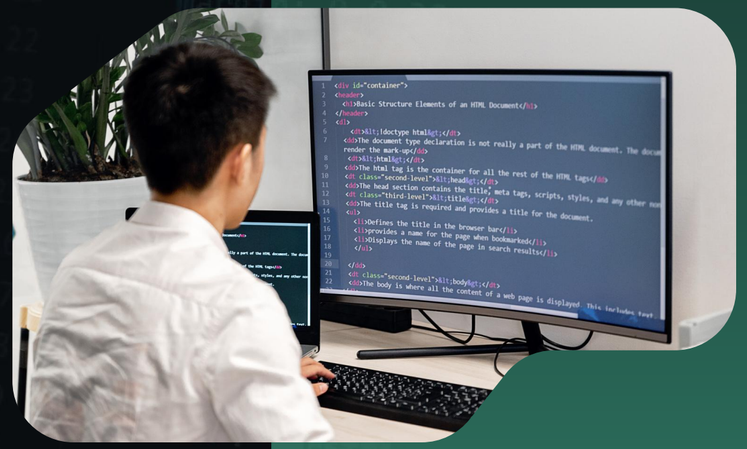
Sunucu istemcinin geçmiş oturum bilgisini belleğinde saklamaz. İstemciden gelen her istek, sunucunun o işlemi gerçekleştirebilmesi için gereken tüm yetki ve veri bilgilerini içinde taşımalıdır.



RESTful Olmanın 6 Temel Kuralı

3 - Önbelleklenebilirlik (Cacheability):

Sunucudan dönen yanıtlar açıkça önbelleğe alınıp alınamayacağını belirtmelidir. Böylece tarayıcı veya aracı sunucular aynı veriyi tekrar tekrar sunucudan istemez.



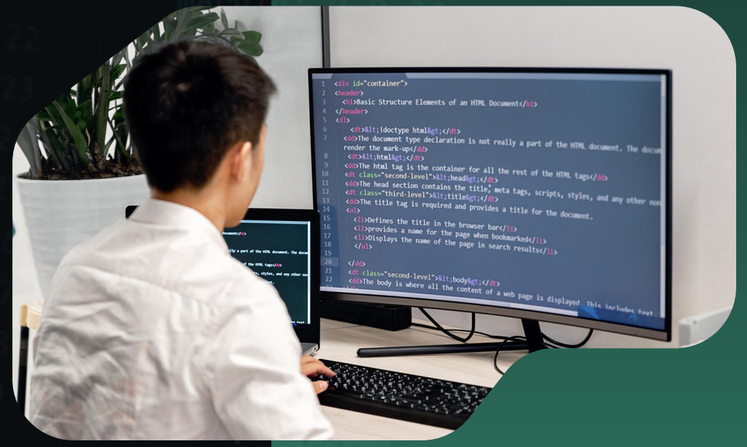
```
HTTP

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: max-age=3600, public
```

RESTful Olmanın 6 Temel Kuralı

4 - Uniform Interface (Tekdüze Arayüz):

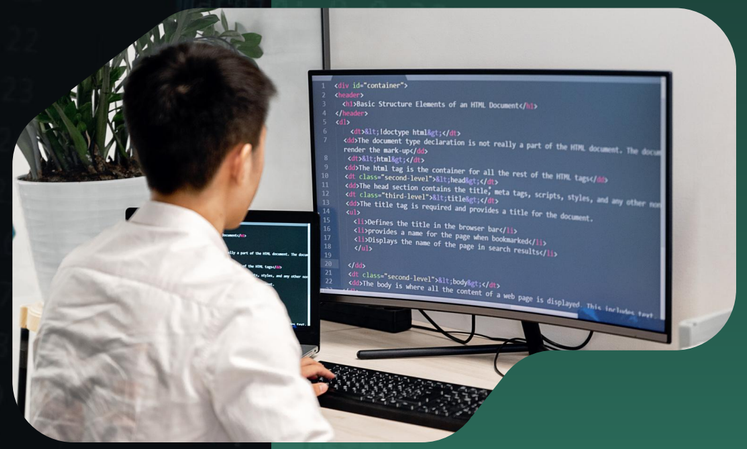
Kaynakların URI ile tanımlanması, temsillerle yönetilmesi ve standart arayüz kullanılması.



RESTful Olmanın 6 Temel Kuralı

5 - Katmanlı Sistem (Layered System):

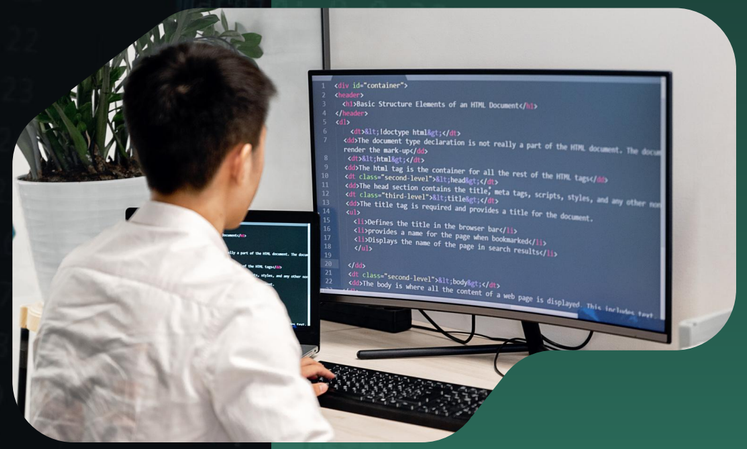
İstemci doğrudan veritabanı sunucusuna mı bağlandı, yoksa arada bir CDN, Load Balancer (yük dengeleyici) veya API Gateway mi var bilmek zorunda değildir. Sistem katmanlar halinde çalışabilmelidir.



RESTful Olmanın 6 Temel Kuralı

6 - İsteğe Bağlı Kod :

Sunucu gerektiğinde istemciye çalıştırılabilir kod (örneğin JavaScript komut dosyası) gönderebilir. REST kuralları içinde zorunlu olmayan tek kısıtlamadır.



Doğru Status Kod Kullanımı

İşlem	Endpoint	Başarılı Kod
Listele	GET /students	200
Tek kayıt getir	GET /students/5	200
Oluştur	POST /students	201
Güncelle	PUT /students/5	200
Kısmi güncelle	PATCH /students/5	200
Sil	DELETE /students/5	204
Bulunamadı	{Tüm Endpointler}	404

DEMO

Teşekkürler

www.furkandurmus.com