**RESTFUL API**

**REST (Representational State Transfer):**

REST, web tabanlı uygulamalar arasında iletişim kurmak için kullanılan bir mimari tarzdır.

REST, kaynakları temsil eden URI'lar (Uniform Resource Identifiers) üzerinden HTTP protokolü üzerinden erişim sağlar.

REST, stateless ve client-server mimarisine dayanır ve genellikle dört temel kısıtlama ile tanımlanır: client-server, statelessness, cacheability (ön belleğe alma) ve uniform interface (tek tip arayüz).

REST, CRUD (Create, Read, Update, Delete) operasyonlarını HTTP metodları (GET, POST, PUT, DELETE) üzerinden gerçekleştirir.

Ancak, REST, kesin bir uygulama veya protokol değildir, daha ziyade bir mimari tarzıdır.

RESTful:

"RESTful" terimi, REST prensiplerine uygun olan bir web servisini veya API'yi ifade eder.

Bir uygulama veya servis, REST mimarisini takip eden ve REST prensiplerine uygun olarak tasarlandığında "RESTful" olarak adlandırılır.

RESTful uygulamalar, stateless olmalı, HTTP metodlarını ve URI'leri doğru şekilde kullanmalı ve uniform bir arayüze sahip olmalıdır.

Genellikle, REST mimarisini takip eden bir web servisi veya API, "RESTful API" olarak adlandırılır.

Özetle, "REST" bir mimari tarzıdır ve web tabanlı uygulamaların iletişimini tanımlarken, "RESTful" terimi, REST prensiplerine uygun olan bir web servisini veya API'yi ifade eder.

Stateless, her işlem birbirinden bağımsızdır. Sistem veya protokol, bir önceki işlemin durumunu hatırlamaz. Her istek, önceki isteklerin sonuçlarına veya durumuna dayanmaz. HTTP, stateless bir protokoldür, yani her bir HTTP isteği birbirinden bağımsızdır ve önceki isteklerin durumunu hatırlamaz.

**Neden Restful Api kullanırız?**

-Yazılım alt yapısını dışarıya açmak için kullanılır. Dışarıya açmaktan kasıt o apiyi kullanacak olan yazılımın kullanılabilmesidir.

2 şekilde dışarıya açılır: Birinci api ikincisi hacking.

-Farklı uygulamaların birbiriyle ortak veri yapısı ile iletişimini sağlamak için.

-Farklı istemcilerin ihtiyaçlarına farklı endpointlerle cevap verebilmek için. XML, JSON, gRPC…

API: Application Program Interface

REST: REspresentational State Transfer

Resource: Apiden elde edilen bir veri parçasına denir.

Request: Bir apiye yapılan çağrıya denir.

Response: Apiden dönen sonuca denir.

Client – Internet – API (XML ya da JSON BİLGİSİ) – Web Server – Database

-------------------------------------------------------------------------------------------🡪

REQUEST

🡨------------------------------------------------------------------------------------------

RESPONSE

Bir request 4 şeyden oluşur:

Endpoint (Sorgu oluşturabilmek için kullanılan URL’lerdir. Diğer adı route’dur.)

HTTP method

HTTP header

Data

Endpointler, bir yazılım geliştiricinin API'ye erişmek ve API tarafından sağlanan hizmetlerden yararlanmak için kullandığı önemli bir kavramdır.

HTTPS, web sitesi ile kişi arasındaki veriler şifrelenir. Üçüncü kişiler erişemez.

SSL sertifikası vardır. SSL (Secure Sockets Layer) sertifikası, web siteleri ve diğer çevrimiçi hizmetlerde güvenli iletişim sağlamak için kullanılan dijital bir belgedir. SSL sertifikası, bir web sitesinin kimliğini doğrular ve kullanıcıların web sitesiyle güvenli bir şekilde etkileşim kurmasını sağlayan şifreleme teknolojisi sağlar.

SSL sertifikası, web tarayıcıları ve sunucular arasında güvenli bir bağlantı kurmak için kullanılan SSL/TLS protokolünün bir parçasıdır.

Root-Endpoint: API’nin base URL’ine root-endpoint denir.

Path: Endpoint’in içerisindeki bir URL parçasıdır.

Tam URL: <https://www.btkakademi.gov.tr/portal/course/player/docker>

Root Endpoint: <https://www.btkakademi.gov.tr/>

Path: [/portal/course/player/docker](https://www.btkakademi.gov.tr/portal/course/player/docker)

**Sorgu parametreleri,** REST mimarisinin bir parçası değildir. Requesti düzenlemek için çeşitli parametreler sunar. Soru işareti ile başlar.

Örn, ?query=value1&query2=value2

**HTTP Metot,** gerçekleştirilen http çağrısının tipini sunucuya bildirmek için kullanılır. Create, Read, Update, Delete gibi işlemler için…

**HTTP Metot Tipleri**

Get, Post, Put-Patch, Delete

Head, get ile aynıdır. Sadece header bölümlerini aktarır.

Connect, sunucuya bir tünel oluşturur.

Options, hedef kaynak için iletişim seçeneklerini açıklar.

Trace, hedef kaynağa giden yol boyunca message loop-back gerçekleştirir.

**API mimarisi her isteğin hangi http metodunu kullandığını tutar.**

**PUT VE PATCH FARKI**

PUT yöntemi, belirtilen kaynağın tamamen güncellenmesi için kullanılır.

PATCH yöntemi, belirtilen kaynağın kısmi olarak güncellenmesi için kullanılır.

Bir PATCH isteği, sadece istemcinin gönderdiği belirli alanları veya alanları günceller, diğer alanları dokunmaz.

Bu yöntem, bir kaynağın yalnızca belirli alanlarının güncellenmesi gerektiği durumlarda kullanılır. Bu, sunucuda bulunan kaynağın yalnızca belirli alanlarının bilindiği ve güncellenmesi gerektiği durumları içerebilir.

Örneğin, bir kullanıcı profili sadece ad veya e-posta adresi gibi belirli alanlarla güncellenecekse, PATCH yöntemi kullanılabilir.

Bir PUT isteği, kaynağın tamamının güncellenmesini sağlar. Yani, istemci tarafından gönderilen veriler, sunucuda bulunan kaynağın tamamıyla değiştirir.

Bu yöntem, genellikle bir kaynağın tüm alanlarının bilindiği ve güncellenmesi gerektiği durumlarda kullanılır.

Örneğin, bir kullanıcı profili tamamen yeni bir veriyle güncellenecekse, PUT yöntemi kullanılabilir.

**HTTP Header**

Http başlıkları, hem istemci hem de sunucuya bilgi sağlamak için kullanılır.

Örn, Content-Type: application/json

Host, User-Agent, Accept, Accept-Language, Accept-Encoding, Accept-Charset, Keep-Alive, Connection, Cookie, Pragma, Cache-Control…