Berkay Akar

WEB APİ

BERKAY AKAR

İçindekiler

[WEB APİ 3](#_Toc101797693)

[HTTP Methodları : 3](#_Toc101797694)

[ **GET :** 3](#_Toc101797695)

[ **POST:** 3](#_Toc101797696)

[ **PUT :** 3](#_Toc101797697)

[ **DELETE:** 3](#_Toc101797698)

[HTTP RESPONSE KODLARI 3](#_Toc101797699)

[ **201 :** 4](#_Toc101797700)

[ **204 :** 4](#_Toc101797701)

[ **200 :** 4](#_Toc101797702)

[ **400:** 4](#_Toc101797703)

[ **500 :** 4](#_Toc101797704)

[ **Not Found :** 4](#_Toc101797705)

[Rest Service : 4](#_Toc101797706)

[İLK APİ PROJESİNİ OLUŞTURALIM 4](#_Toc101797707)

[APİ CONTROLLERİ 7](#_Toc101797708)

[BİR APİ PROJESİNİN KONFİGÜRASYONU 8](#_Toc101797709)

[ENDPOİNT YAPISI 10](#_Toc101797710)

[POST İLE KAYIT İŞLEMİ YAPMA 12](#_Toc101797711)

[PUT İLE UPDATE İŞLEMİ YAPMA 13](#_Toc101797712)

[DELETE İLE SILME İŞLEMI 15](#_Toc101797713)

# WEB APİ

Açılımı **application programming interface** (uygulama geliştirme arayüzü) olarak geçmektedir. Bir uygulama hem masaüstü, hem web ve hemde mobil için geliştirmeye kaltığımızda proje boyunca bizlere farklı platformda backend geliştirmemeyi sağlamakta ve tek merkezi bir uygulama üzerinden servis sağlamamıza yaramaktadır. Ayrı ayrı programlara hizmet etmek için geliştirilecek olan (hizmet edecek olan) bir uygulamadır. Genellikle Uygulamalar arası iletişimde **JSON** veya **XML** formatı kullanılmaktadır.

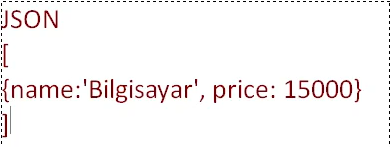


Figure 1 Tipik bir JSON formatı ve örneği

## HTTP Methodları :

* **GET :** Genellikle veriyi çekmek amaçlı kullanılan bir methoddur.
* **POST:** Bir veriyi eklemek maksatlı kullanılmaktadır.
* **PUT :** Birveriyi güncellemek için kullanılması gerekecektir.
* **DELETE:** Bir veriyi silmek için kullanılmaktadır.

Rest servisler de bu işlemler kullanılacaktır. Bundan ötürü ilk olarak HTTP methodları bilinmelidir.

metin, ekran, televizyon, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

İstekler sonucu görüldüğü üzere bir response objesi dönülerek işlemin statüsünü belirlememiz gerekmektedir. HTTP Kodlarını işleyerek sürece devam edelim.

## HTTP RESPONSE KODLARI

* **201 :** Bu bir veriyi post ettiysek bizlere dönmesi gereken koddur. Veri başarılı bir şekilde APİ üzerinden kaydedilmiştir. Response objesi ise bizlere bu kodu döndürmektedir.

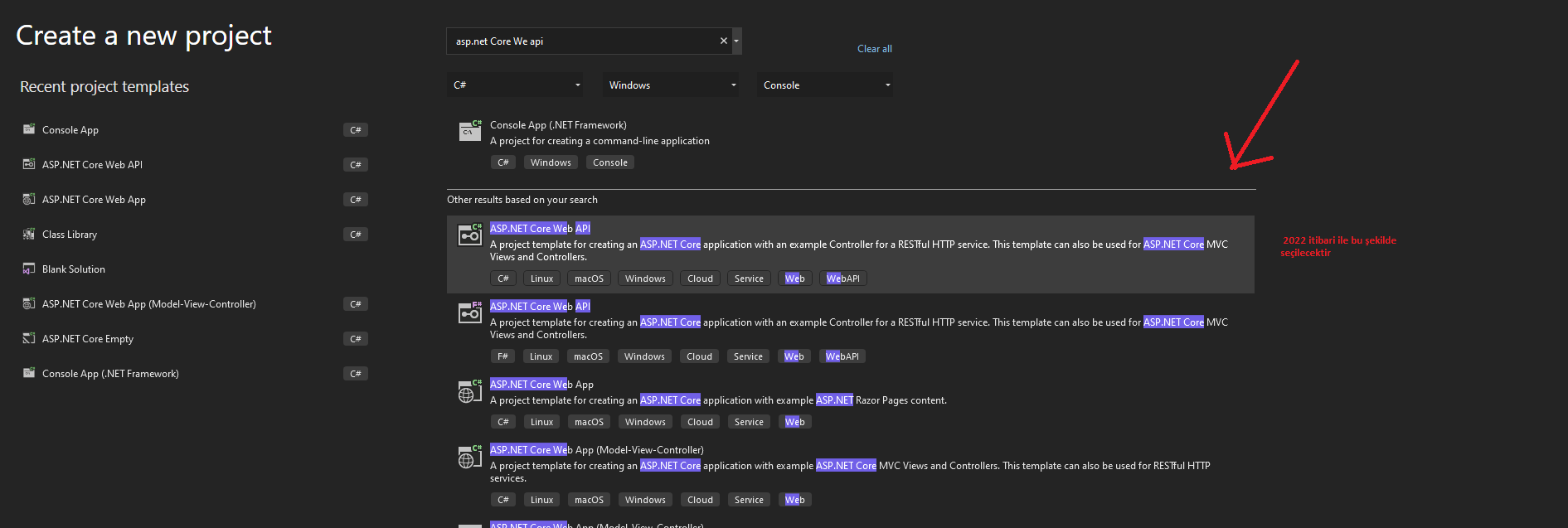
* **204 :** No Content dönecektir. Bu dönme 204 kodu ile gerçekleştirilmektedir.
* **200 :** Bir veriyi yada tüm verileri başarılı bir şekilde listelemeyi becerdiysek dönecek kod dur.
* **400:** Create işleminde validasyon hatasına takılırsak Bad request olarak dönmektedir.
* **500 :** Eklerken yüklerken veya getirirken bir hata ile karşılaşılması durumudur.
* **Not Found :** Güncellerken veya görüntüleme yaparken verinin bulunamamsı sonucu dönecek hata mesajıdır

## Rest Service :

Eğitim boyunca Restfull ve Rest servisler incelenecektir. Rest servislerden başlangıç aşamasında kısaca bahsetmek gerekirse MVC yapısındaki gibi bir controller ile endpointler vasıtası ile değilde HTTP methodları üzerinden etkileşime geçildiği yapıdır.

Normal şartlar altında MVC den hatırlanacağı üzere **controller/action** endpointi örneği üzerinden controller ve o controller içerisinden ne yapılacaksa onun action method çağırılıyorken Rest Servislerde ise HTTP Methodları üzerinden sadece **localhost:44735/api/controller** endpointi kullanılarak action ismi farketmeksizin sadece http protoklu üzerinden etkileşime geçilmesi gerekmektedir. Beraberinde Rest servislerde XML , JSON , HTML gibi veri tiplerini desteklediğinden SOAP mimarisine göre daha esneklik sağlamaktadır.

## İLK APİ PROJESİNİ OLUŞTURALIM



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Open API .NET 5.0 ile birlikte gelmekle birlikte 3.1 ve aşağısında bulunmamaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 2 BİR APİ NİN KLASÖR YAPISI (.Net 5.0)

Bir api projesi açıldığında WeatherForecast.cs isimli bir controller ve model dosyası belirmektedir. Bu Varsayılanda gelen bir controller olup random sayılar üreterek bizlere bilgi vermeyi sağlayan bir dosyadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü üzere içerisinde **ControllerBase** sınıfındankalıtım alan (Bu sınıf hem .net core mvc deki controller sınıfının atası olan bir abstract class, ve beraberinden belirtmek isterimki tüm api controller leri **ControllerBase** sınıfından kalıtım alırlar**.** )

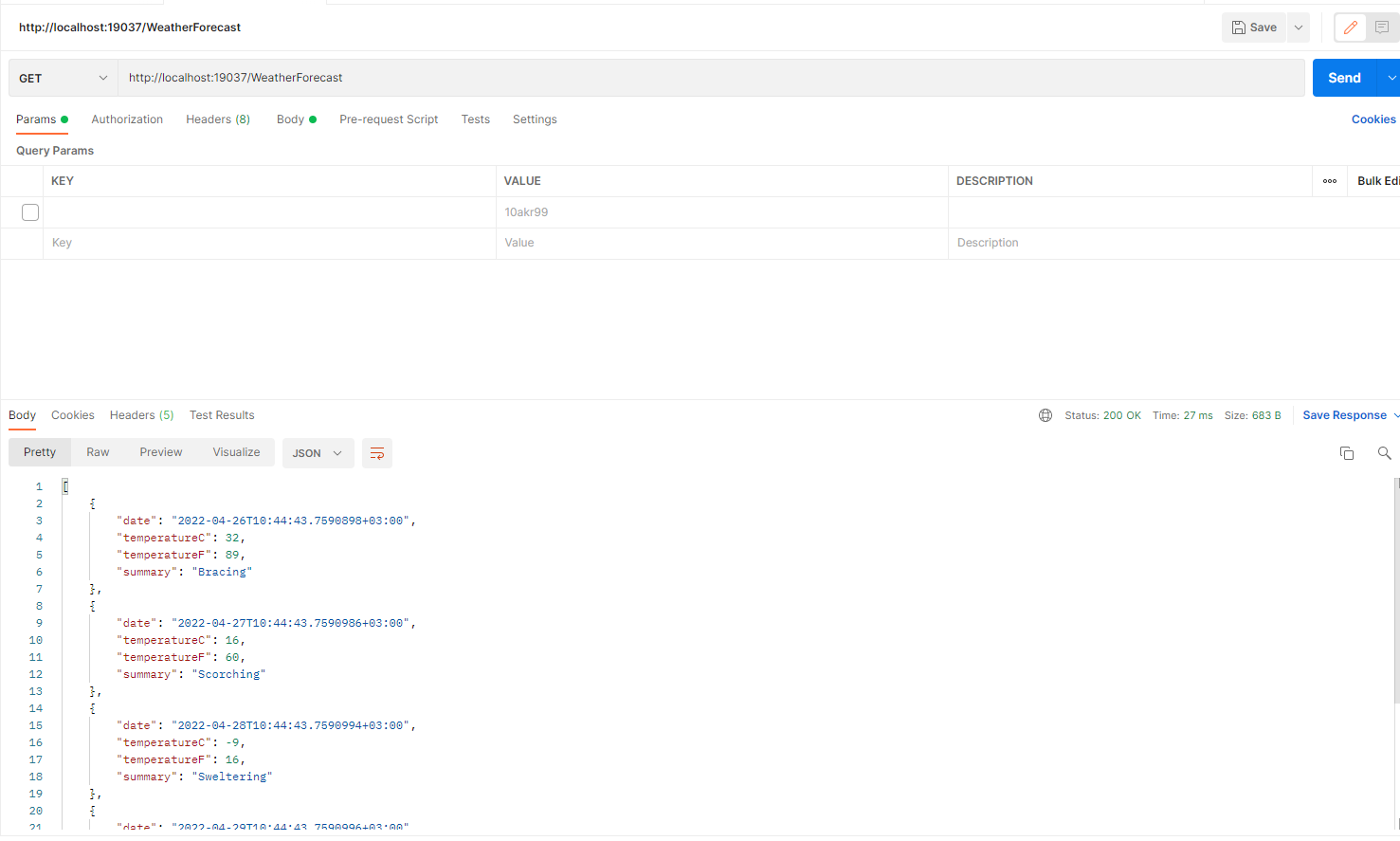
İçerisindeHTTPGET olarak sadece GET methodunun bulunduğu, random Enumerable döndüren bir sınıftır. İçerisinde sadece Get method vardır.

Bir APİ controller in api controller olması attribute verilerek sağlanır.

[ApiController]

[Route("[controller]")]

Bu attribute classın en üstüne konulur ve işlem bu şekilde gerçekleşir.



Şimdi yapıyı daha iyi analiz edebilmek için bir controller oluşturalım.

metin, ekran görüntüsü, ekran, televizyon içeren bir resim

Açıklama otomatik olarak oluşturuldu

# APİ CONTROLLERİ

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bir Api Controlleri için olması gerekenler:

* **ControllerBase** abstract sınıfındankalıtım alması,
* Kalıtım alan sınıfın en üstüne gidilerek **[APİCONTROLLER]** attribute ü verilmesi
* Her bir fonksiyonun başına olduğundan ötürü **HTTP METHODLARI (GET POST PUT DELETE )** methodlarının eklenmesi ve bu şekilde controller içerisinde istek atılması.
* **Route Configürasyonunun** verilmesi **[Route(Controller)]** (!!! Eğer bu routig yoksa gerçekleşecektir.)

Temel olarak bu şekilde söylenebilmektedir. Bu sayfada örnek olarak bir class oluşturulmuş ve get method içerisinde bir nesnesi oluşturularak ok result I içerisinde yollanması hedeflenmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 3 Karsilama Ekrani Fonksiyonu içerisinde OK ile nesne döndürme ve bunun Postman tarafindan tetiklenmesi.

URL e bakıldığında **Homecontroller** , http methoduna bakıldığında **get** ile içerisinde sadece bir adet get method bulunan **Karsilama\_Ekrani\_Fonksiyonu()** get http attribute üne sahip fonksiyonun tetiklenmesi gerçekleştirilmiştir.

# BİR APİ PROJESİNİN KONFİGÜRASYONU

metin, ekran görüntüsü, ekran içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 4 Program.cs Dosyası

Bir Web projesinden hatırlanacağı gibi **Program.cs** üzerinden **CreateHostBuilder()** Methodu vasıtası ile **Startup.cs** deki ayarlar build edilip çalıştırılacaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 5 Startup.cs

Configure service methodundaki farklılık ise bir web projesinde service.AddControllerWithViews eklerken artık sadece Add.Controller dememiz yeterli. Web projelerinde viewler bulunmamaktadır. Beraberinde MVC projelerinde olduğu gibi

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

Bu endpointi global bi şekilde düzenleyebiliriz. Bunun nezlinde bu varsayılan hali ile kullanmak istersek tüm controllerlerin başına attribute olarak controllers eklememiz gerekecektir.

# ENDPOİNT YAPISI

Daha öncelerinde Web projelerinden alışık olduğumuz routing şeması düşünüldüğünde Bir controller, bir action ve isteğe bağlı olarakta değişkenlik gösteren bir id parametresine sahip bir pattern verebilmekteydik. Bu yapı istenirse Api projelerinde de kullanılabilir. Fakat Rest servis amacı düşünüldüğünde bu yaklaşım Rest doğasına aykırı olacağından ötürü sadece controller, ve http method ile erişim daha doğru olacaktır.

Bu şekilde düşünüldüğünde aynı controller içerisinde action method belirtilmeden direk http fonksiyonları ile erişim sağlandığı zaman bir noktada 2 adet aynı http methodundan (isimleri farklı olsada parametreleri farklı olsada farketmeyecektir.) hata verecektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 6 Aynı controller içerisinde get olarak birbirlerinden garklı iki fonksiyon

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu sebeple bu hatanın önüne geçilebilmesi adına

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 7 Routing Örneği

Eğer dışarıdan bir id attribute’ü verilmez **(localhost:19037/Products)** ise tüm ürünleri getiren get ile çalışan **GetAll()** çalışacaktır. Son endpointe bakıldığında sadece Get method ile çalışan tek fonksiyon o olduğundan kaynaklanmaktadır.

Sisteme ayriyeten bir parametre **(localhost:19037/Products/1) gibi** daha verilirse daha geldiğinde burada yapılması gereken şey ise ilk olarak controller Route attribute ünü görerek ProductController ine gelecektir. Bir sonraki aşamada burada id parametresi ile bir get methodunun olduğunu görerek bu aşamada **GetById(int id)** isimli fonksiyonu tetikletecek ve çalıştıracaktır.

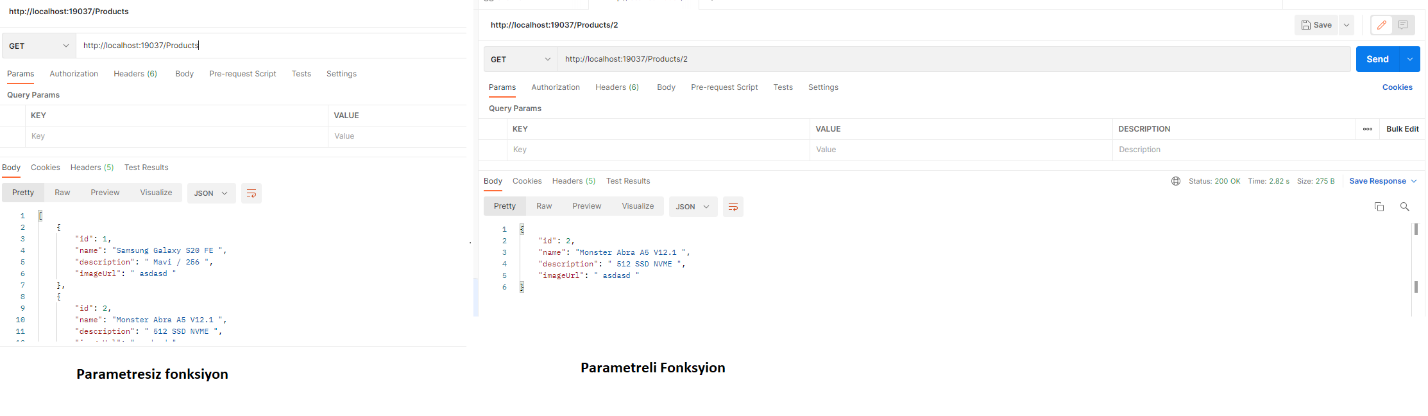


Figure 8 Parametreli Ve Parametresiz Yöntemin Ispatı

# POST İLE KAYIT İŞLEMİ YAPMA

Post işleminde sutun değişkenlerinin bir alttaki class resmindeki gibi olduğunu ve bunun entity framework yardımı ile bir database oluşturulduğunu düşünelim.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İşlem post olacağından ötürü **HTTPPOST** attribute ü kullanılarak fonksonumuz yazılması gerekmektedir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 9 Product Kaydetme

Görüldüğü üzere **HTTPPOST** olarak yukarıda gözüktüğü gibi product nesnesini parametre vererek süreci başlattık. Kayıt işleminin başarılı olması koşulunda bizlere **Created()** fonksiyonu vastıası ile **modelin kendisi** ve **201** kodu dönecektir. JSON format bir aşağıda verilmiştir.

{

    "id": 4,

    "name": "VESTEL BULAŞIK MAKİNESİ",

    "description": "A++ BULAŞIK MAKİNESİDİR. BEYAZ EŞYA",

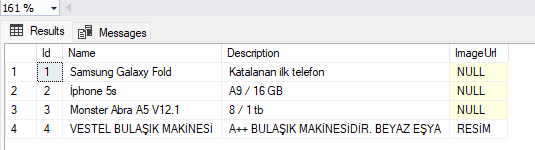
    "imageUrl": "RESİM"

}

Method **POST** olarak seçilmiş olup **BODY** parametresine **RAW** seçilerek **JSON** dosya tipinde bu işlem gereçekleştirilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu



# PUT İLE UPDATE İŞLEMİ YAPMA

HTTP fonksiyonlarından **PUT** kullanılması ile birlikte güncelleme işlemlerini gerçekleştirmiş olabilmekteyiz. Bu işlemi yaparken MVC den de hatırlanacagı üzere model komple parametre olarak yollanmalıdır. Örneğini bir alt kısımda mevcuttur.

Ekleme ve slime işlemlerinde response objesinin içerisinde herhangi bir model donmemiz gerekmemektedir. Bu sebeple işlemlerin gerçekleşmesi sonucunda **NoContent()** kullanılacaktır.

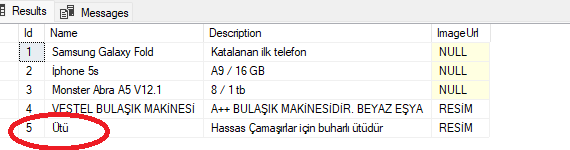


Figure 10 Güncelleme işlemi yapılacak olan model

Model numarası 5 olan bu model orjinalinde bu halde iken içerisinde çeşitli değişiklikler yapılacak ve bir aşağıdaki model e benzetilecektir. Update işlemi için **PUT** seçildikten sonra ürün ekler gibi body => Raw => JSON seçerek devam edecegiz. Model yollarken dikkat etmemiz gereken kıssas id parametresini yollamaktır. Kayıt ederken id ile yollamak sistemde problem çıkartabilmektedir. Fakat update te yollamamız zorunludur. İD si verilen ve geri kalan değişiklikler üzerinde yapıldıktan sonra ortaya çıkan JSON bir aşağıda verilmiştir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü üzere güncelleme işlemi gerçekleştirildi ve bizlere **204** kodunu döndürdü.

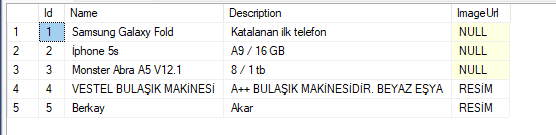
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 11 JSON u Put Ettikten sonraki Çıktı

# DELETE İLE SILME İŞLEMI

Veritabanından bir kayıt silmek için **DELETE** methodu kullanılması gerekmektedir. Bu neticede id işlemi baz alınarak bir silme gerçekleşeceğinden kaynaklı fonksiyon parametresine id vermemiz yeterlidir.



Veritabanındaki 5 numaralı kaydı silmek istediğimizi varsayalım.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Görüldüğü gibi bu aşamada bizlere **204** kodu sağlanmış olmaktadır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 12 Veri Tabanının Son Hali

# DOSYA GÖNRDERME İŞLEMLERİ

Dosya yüklemek için **HTTPPOST** methodunu kullanmamız gerekmektedir.