ÖdeAl Test Case Çalışması:

Bu doküman yapılan çalışma hakkında bilgi vermek amaçlı hazırlanmıştır. Genel bilgiler sonrası sıra sıra anlatılacaktır.

Postman / Newman ile api Testi :

Postman, kullanıcıların hem basit hem de karmaşık HTTP isteklerini hızlı şekilde bir araya getirmelerini sağlayarak API’leri paylaşmak**, test etmek**, geliştirmek, belgelendirmek ve monitör etmek için kullanılan güçlü bir HTTP istemcisidir.

Postman’i özel kılan bir diğer özelliği ise otomasyon yeteneğidir. API’leri otomatize etmeyi kolaylaştıran tasarımı, hazır kod parçaları, anlaşılabilir arayüzü ve sağladığı entegrasyonları (Newman, Jenkins, Travis CI, Docker) sayesinde iyi bir otomasyon aracıdır.

Collection’lar Postman’in yapıtaşını oluşturmaktadır. Collection’lar içerisinde klasörlerimizi ve API’lerimizi saklayabiliriz. Ayrıca yazdığımız pre-request ve test scripts sayesinde test caselerimizi otomatize edebilir, raporlayabilir ve Jenkins’e entegre edebiliriz.

**1.** **Pre request script:** Bir request (istek), sunucuya gönderilmeden önce çalıştırılan kod parçalarından oluşur.

**2.** **Tests:** Bir request sunucuya gönderildikten ve response alındıktan sonra çalıştırılan kod parçalarından oluşur.

**3. Environment:** ortam değişkenleri oluşturabildiğimiz ve bu değişkenleri sakladığımız bir alandır. Environment altında bulunan değişkenleri, farklı request veya collection’larda çağırabiliriz. Bu değişkenleri pre-request ve test alanlarında set veya get edebiliriz.

**Test Cases:**

1. **Bir user create edilecek.**

<https://petstore.swagger.io/v2/user> --> atılan request ile ,

request Body içeriği 🡪

{

  "username": "*{{userName}}*",

  "firstName": "",

  "lastName": "*{{lastName}}*",

  "email": "*{{eMail}}*",

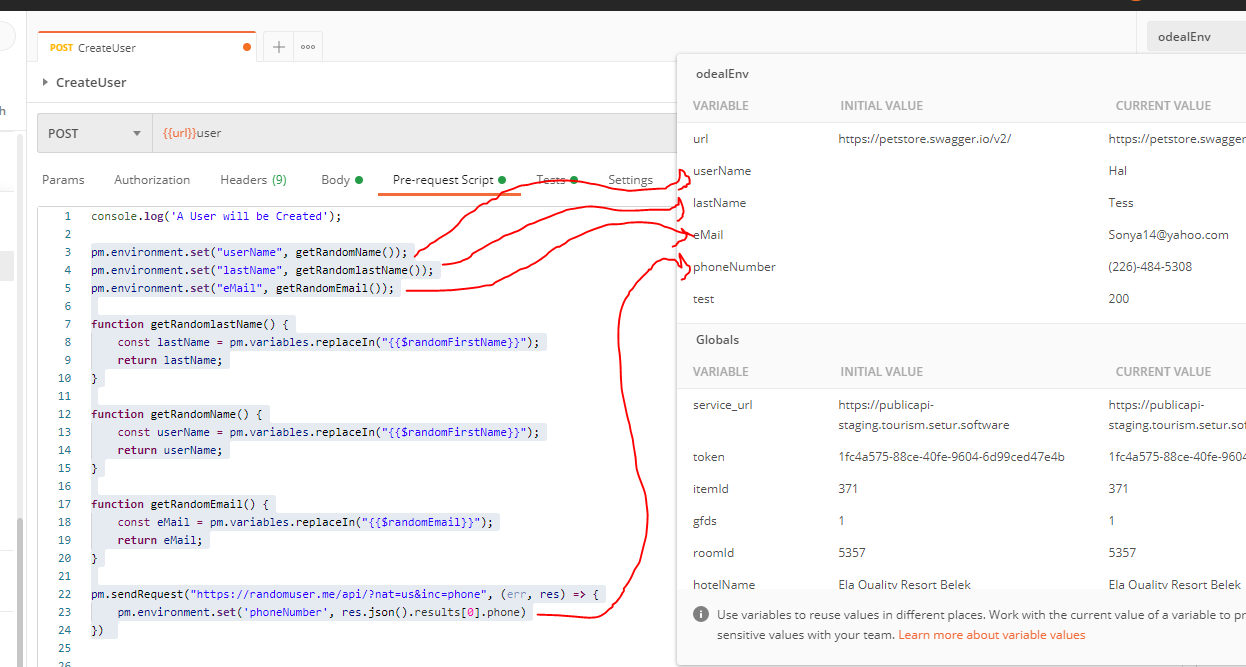
  "password": "11223344",

  "phone": "*{{phoneNumber}}*",

  "userStatus": 1

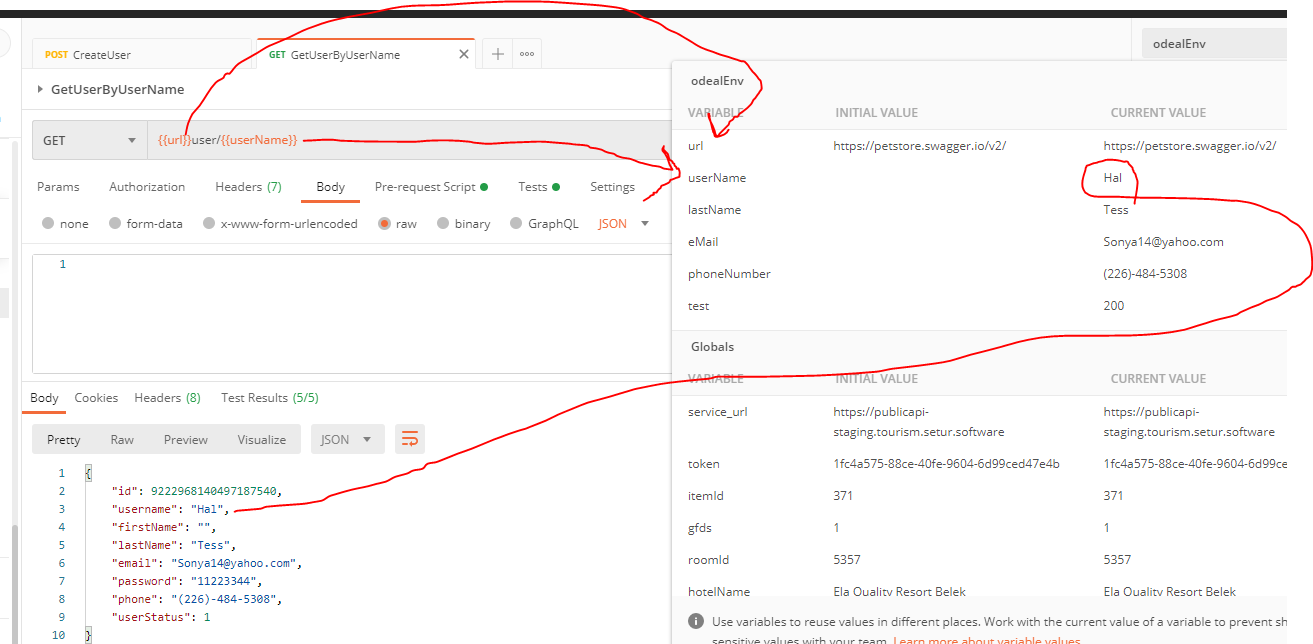
}

Burada gönderilen bilgiler random olarak üretilmiş env. Da tutularak requeste set edilmiştir.

İstek atılmadan testler gerçeklenmeden önce Pre-requestScript alanı çalıştığından bahsetmiştim. Burada randon mane, lastname, mail ,.. fonskiyonları yazılarak env set edildi ve request içerisinde çağırıldı. Böylece her seferinde farklı verilerle kayıtlar atılmış oldu.Ek olarak Pre-requestScript alanına en başa her istekten önce log atılarak case adı yazılmıştır

1. **Create edilen user’ın username ve password bilgileri kullanılarak kullanıcıbilgilerigetirilecek**

Create edilen user name bilgisini tutmuştuk env. da mevcut. Bunu parametrik olarak getirip isteğimizi atıyoruz.



Burada bir önceki istekte oluşturulan user mı getirildi doğru mu kontrolü yapılmıştır.

var jsonData **=** JSON.**parse**(responseBody)

pm.environment.**set**("userId", jsonData.id);

**if** (jsonData.username **==** pm.variables.**get**('userName')) {

        tests["Create edilen usr getirlidi Passed"] **=** jsonData **!==** **null**;

    }

**else**{

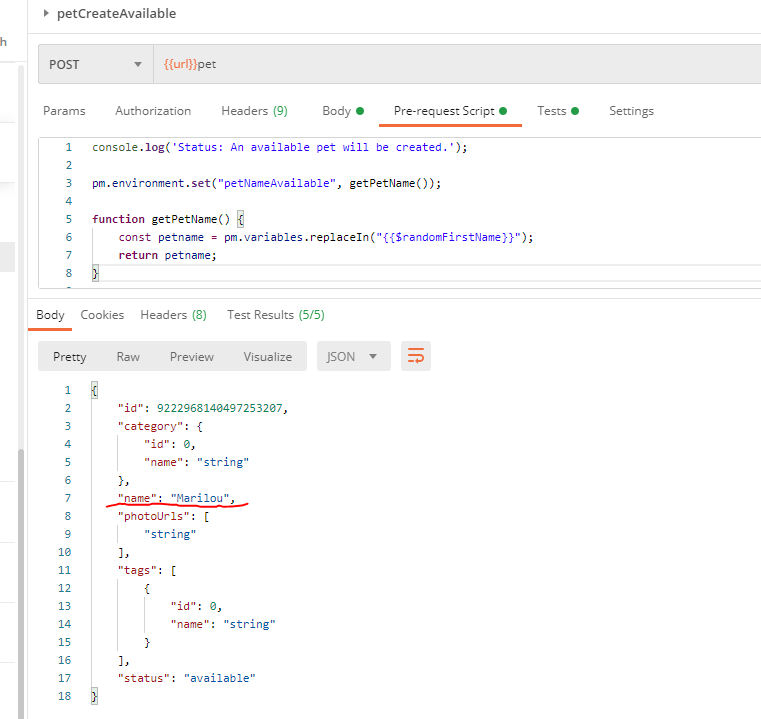
        tests["Create edilen User getirilemedi FAILED"] **=** jsonData **==** **null**;}

1. **Status : Available olan bir pet create edilecek.**

Statusa = available olar bir pet create edilmesi için atılan requeste Statusu available verilmiştir. Burada create edilen pet name her seferinde random verilmesi sağlanmıştır.

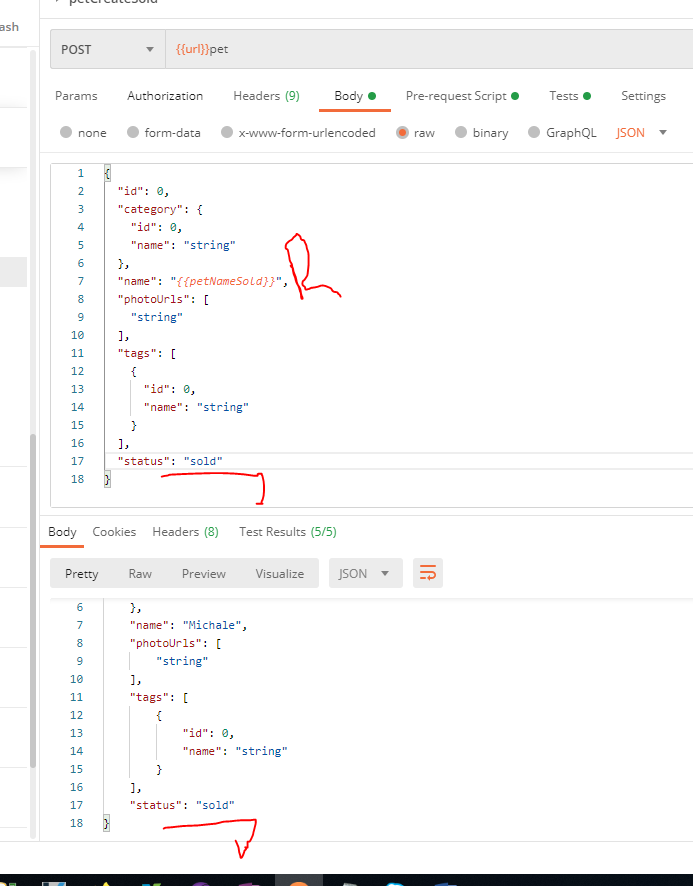
Burada dönen id son 4 rakamında yuvarlama yapılarak döndüğü görülmüş bu yüzden response cevabın içinden regex kullnılarak string set edilerek kullanılmıştır.

responseBody **=** responseBody.**replace**(/\"id"\s**?**:\s**?**([0-9]**\***)/**g**, `"id": "$1"`)

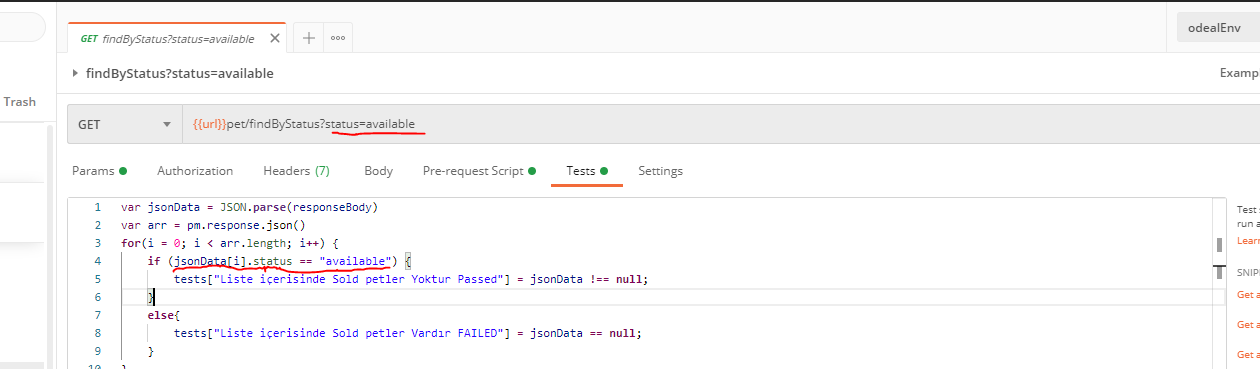


1. **Status : Sold olan bir pet daha create edilecek.**

Statusa = sold olar bir pet create edilmesi için atılan requeste Statusu sold verilmiştir. Burada create edilen pet name her seferinde random verilmesi sağlanmıştır.



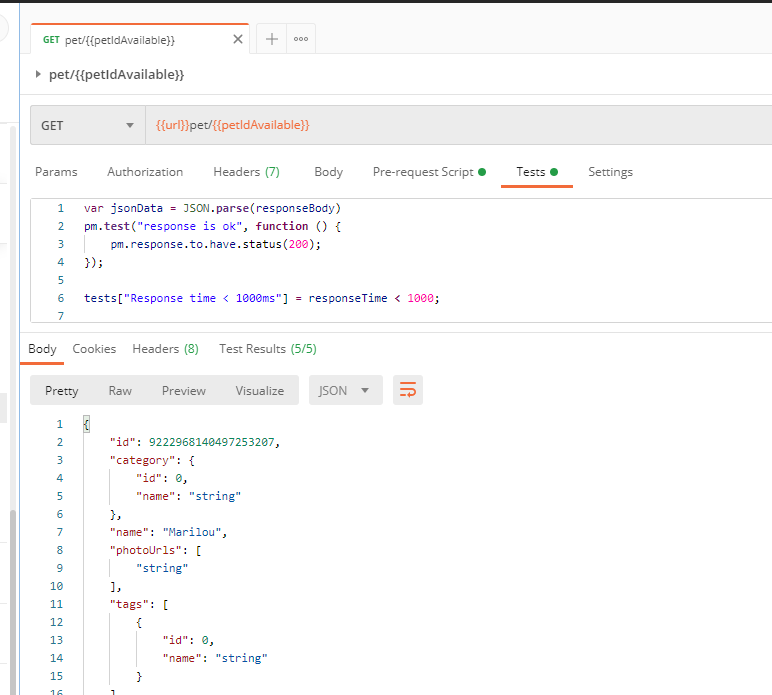
1. **Available olan petlerin listesi getirilecek ve bu liste içinde Sold olan petlerin listesi olmayacak.**



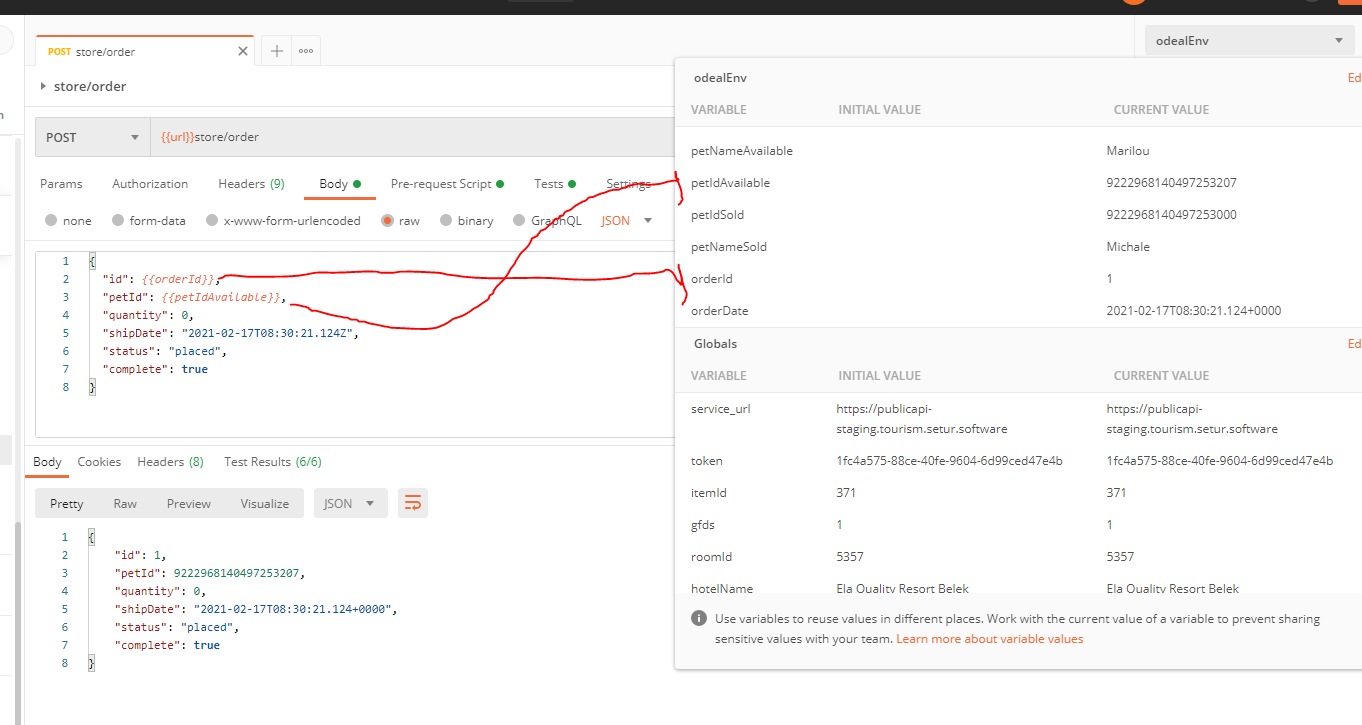
1. **Available olan pet’in id'si kullanılarak ilgili pet’in dataları getirilecek.**

Env tutulan en son create edilmiş pet verilerek pet’in dataları getirilir.

{{url}}pet/{{petIdAvailable}}



1. Aynı pet id kullanılarak 1 adet ve id’si 1 olan bir pet sipariş edilecek.



Ve kontrol edilir comlete == true ??

var jsonData **=** JSON.**parse**(responseBody)

pm.environment.**set**("orderDate", jsonData.shipDate);

**if** (jsonData.complete) {

        tests["pet sipariş edildi Passed"] **=** jsonData **!==** **null**;

    }

**else**{

        tests["pet sipariş edilmedi FAILED"] **=** jsonData **==** **null**;

    }

1. **Order id’sini 1 verdiğimiz Pet’in satın alma siparişi bulanacak ve doğru siparişin bulunup bulunmadığı kontrol edilecek.**

{

    "orderId"  : *{{orderId}}*

}

var jsonData **=** JSON.**parse**(responseBody)

**if** (jsonData.shipDate **==**pm.variables.**get**('orderDate') ) {

        tests["doğru sipariş bulundu Passed"] **=** jsonData **!==** **null**;

    }

**else**{

        tests["doğru sipariş bulunmadı FAILED"] **=** jsonData **==** **null**;

    }

1. **Kullanılan pet silinecek.**

Create edilen pet id’si env tutulmuştu, parametre olarak get ederek bu pet id silinir.

{{url}}pet/{{petIdAvailable}}

1. **Silinen Pet’i pet id kullanarak Get isteği atıp dönen response’dan Pet’in silinmiş olduğu doğrulanacak.**

{{url}}pet/{{petIdAvailable}} create edilen id tutulmuş ve id silinmişti tekrar aynı id ile GET edilir dönen response cevaba göre kontrol edilir.

var jsonData **=** JSON.**parse**(responseBody)

**if** (jsonData.message **==**"Pet not found" ) {

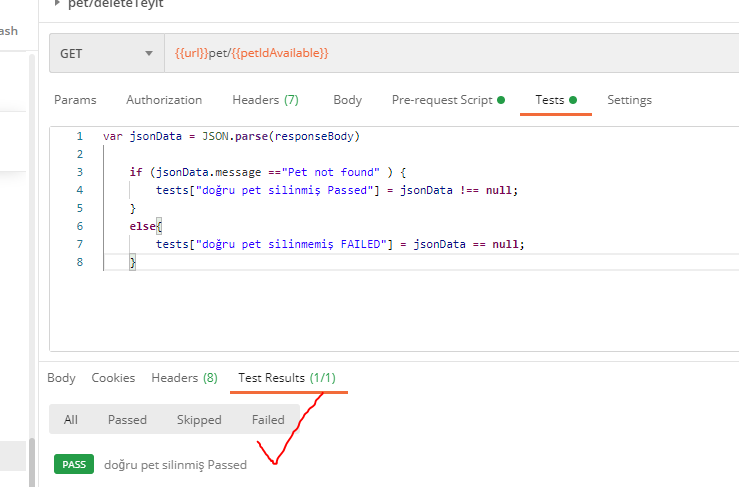
        tests["doğru pet silinmiş Passed"] **=** jsonData **!==** **null**;

    }

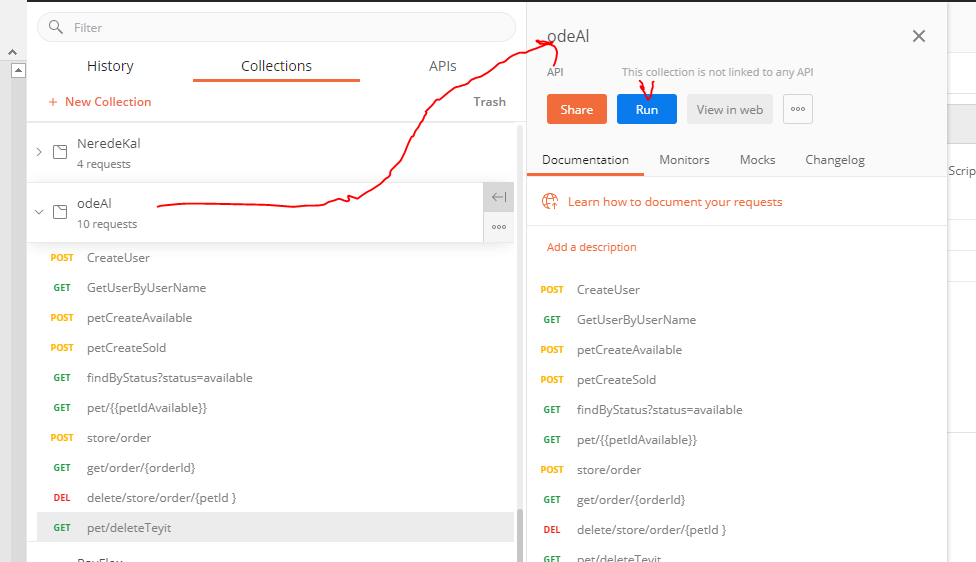
**else**{

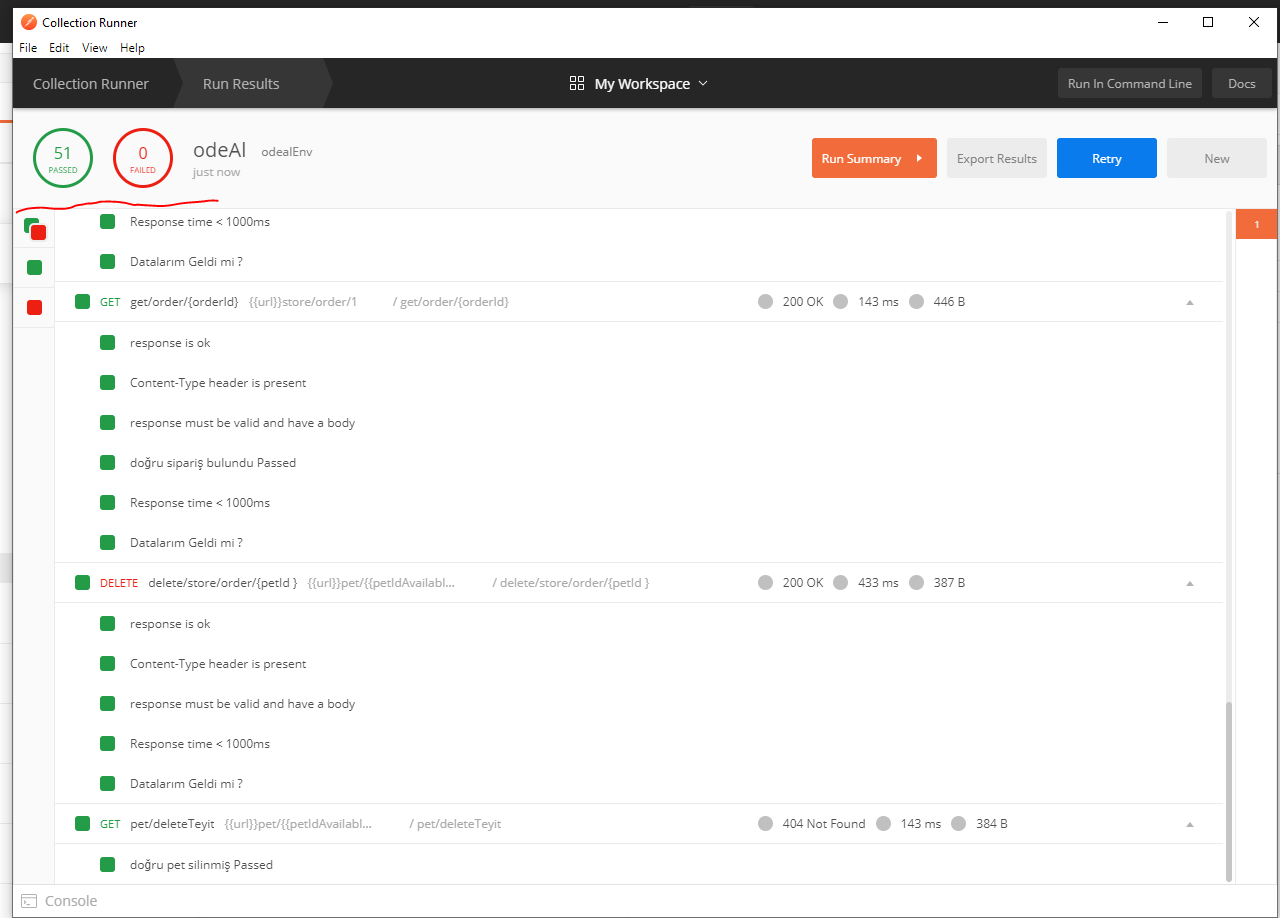
        tests["doğru pet silinmemiş FAILED"] **=** jsonData **==** **null**;

    }



Postman üzerinden Collection Run edilebilir.





Newman Entegrasyonu ve Html Reporter

**Newman:** Postman collection’larımızı Collection Runner üzerinden değil doğrudan komut satırı üzerinden koşmamızı ve test etmemizi sağlayan bir Npm (Node Package Manager) teknolojisidir. Npm javascript betik dili için geliştirilmiş olan ve Node.js’in standart olarak kabul ettiği bir paket yönetim sistemidir. Npm komut satırından çalıştırılır. Bu sayede CI (Continuous Integration) sistem ve server’lara kolayca kurabiliriz (Jenkins, Travis CI, Gitlab,Bamboo). Burada Newman diğer tool’lara ulaşmak için kullanılan bir gateway gibi hayal edilebilir.

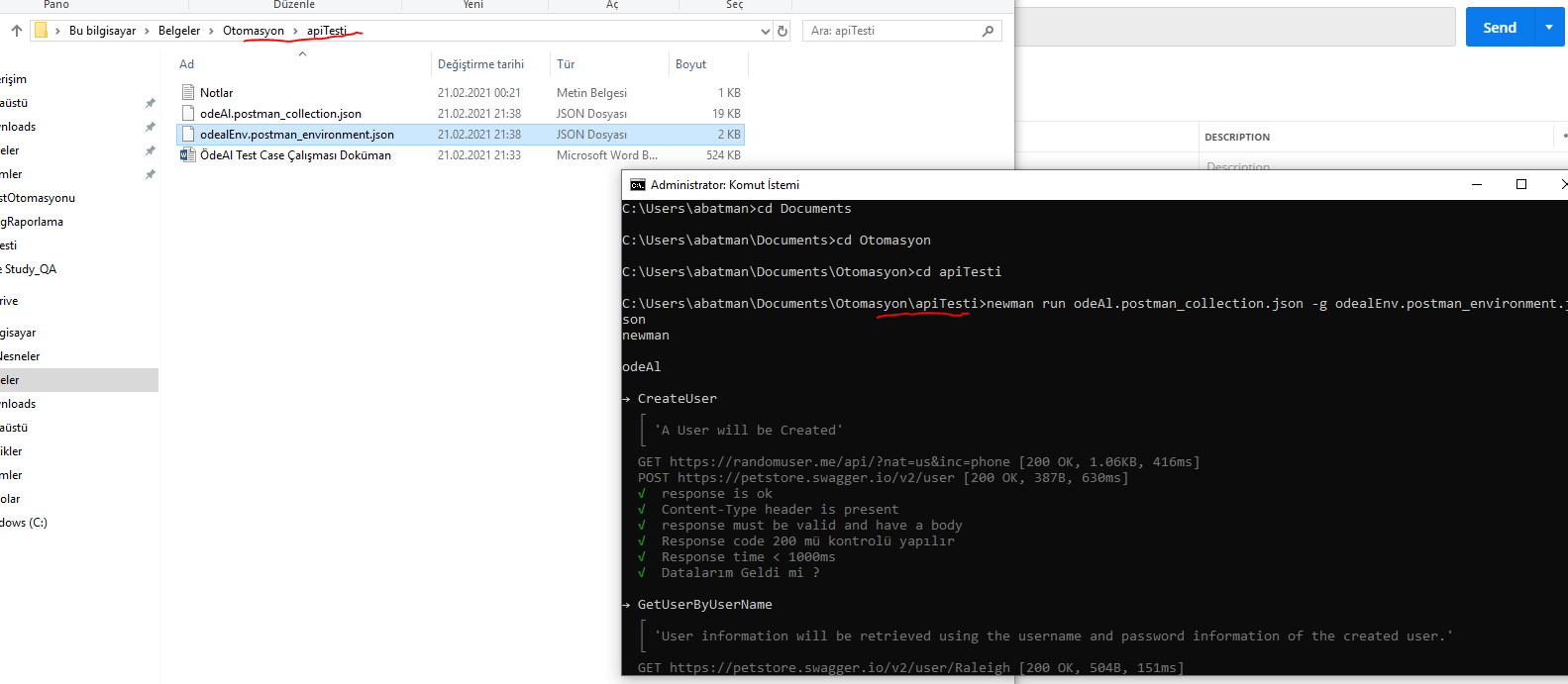
1. Postman Collection’ları Newman ile komut satırından çalıştırabilmemiz için öncelikle bilgisayarımıza node.js indirmemiz ve kurmamız gerekmektedir. https://nodejs.org/en/ sitesi üzerinden node.js indirin ve kurunuz.
2. Node.js yükledikten sonra komut satırını (cmd) açınız. Newman’ı aşağıda yer alan komut ile kurunuz.

npm install –g newman

Daha sonra çalıştırmaya hazırdır kaydedilen collection ve environment json tipinde yoluna gidilerek koşturabilirsiniz, Şimdi yapılan test için koşumu paylaşacağım.

Scriptler : ek olarak da göndereceğim. 

Komut : newman run odeAl.postman\_collection.json –g odealEnv.postman\_environment.json



Newman üzerinden run edilen komut olarak farklı özelliklerde barındırmaktadır,

n : collection’ı kaç kere koşacağımızı belirlediğimiz özelliktir. Örneğin 10 kere koşmak için; newman run collection\_name.json –n 10

e : Collection Runner’da olduğu gibi collection’da kullanılan ve environment dosyasında tanımlanan değişkenleri kullanmamıza olanak sağlar. Örn; newman run collection\_name.json –e environment.json

g : Collection’da kullanılan ve global dosyasında tanımlanan değişkenleri kullanmamıza olanak sağlar.Örn; newman run collection\_name.json –g MyWorkspace.postman\_globals.json