



# Mentoring





# Bugün ne yapacağız ?



- ▶ POSTMAN NEDİR ?
- ▶ POSTMAN'DE TANIMLAR
- ▶ API TEST CASE NASIL YAZILIR ?



# POSTMAN





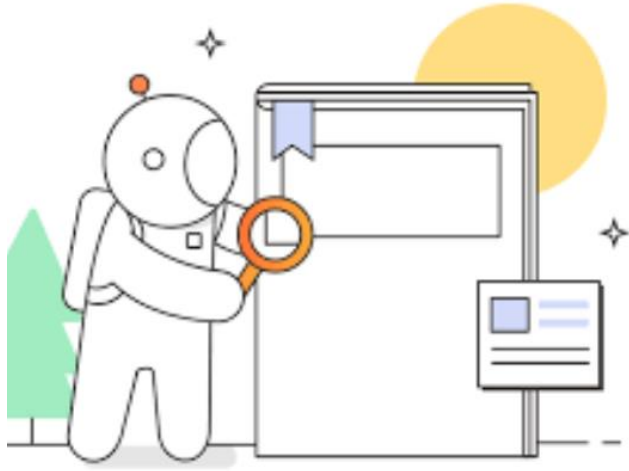
## Postman Nedir?

Postman, web servisleri arasında iletişim kurmak , API'ları test etmek, dokümente etmek, monitör etmek için milyonlarca yazılımcı tarafından kullanılan bir programdır.



Kullanım kolaylığından  
dolayı tercih edilmektedir.  
İşlevselliği, kullanıcıların API  
istekleri oluşturmalarına, bu  
istekleri özelleştirmesine ve  
ardından bu istekleri çalıştırarak  
API yanıtlarını incelemesine  
olanak tanır.





API(Application Programming Interface) farklı uygulama yazılımlarının birbirleri ile etkileşim sağlamasına olanak sağlar. Client (Android) ve Backend(java) yazılımlarının Restfull Api ile iletişim kurması buna örnek verilebilir.

Postman sayesinde uzun uzun kodlar yazmak yerine API'lerimizi kolayca test edebiliriz. Birçok özelliği sayesinde kolay bir şekilde istek hazırlayıp gelen cevap değerlerini kullanabiliriz



# Postman'de Test Yazmak Nedir?



Postman'de, test etmiş olduğumuz servisleri, klasörleyerek kaydedebiliyoruz. Bu klasörlere de “Collection” diyoruz. Kaydettiğimiz bu servisleri tek tuşla, dinamik davranışlar ekleyerek, dilediğimiz şekilde çalıştırıp sonrasında durum raporu alabiliyoruz.

Bunun bize bir kaç faydası bulunuyor. En temel olarak;



# 1. Zaman tasarrufu



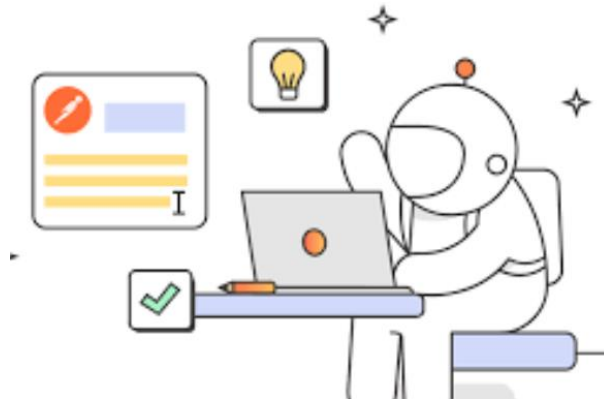
Tüm servis çağrılan ekranları gezerek her fonksiyonun çalıştığından emin olmak için ürünü gezdiğinizde uzun saatler alabiliyor. Postman'de zamanlayıcı kurarak testin her sabah çalışmasını ve size mail olarak sonucu atmasını sağlayabiliyorsunuz.





## 2. Yk testi yapabilme olanađı

Hibir Őekilde elle yapamayacađımız bu testi, Iteration sayısını 1.000'e, 10.000'e ıkararak rahatlıkla yapabiliriz.





### 3. Farklı Ortamlarda Test yapabilme

Sizlerin de test, pre-prod, prod gibi birden fazla ortamınız olduğunda, aynı testi ortam (Environment) değiştirerek rahatlıkla yapabiliyorsunuz.





Bu kullanımlar kolay ancak testi işimize yarayacak şekilde çalıştırabilmek için JavaScript kodunda scriptler yazmamız gerekiyor. Postman'ın web sitesinde ve postman içerisinde hazır scriptler bulunuyor.

### Writing tests

You can write test scripts for your Postman API requests in JavaScript. Tests allow you to ensure that your API is...

[learning.postman.com](https://learning.postman.com)





The screenshot shows the Postman interface with the 'Tests' tab selected. A red arrow points to the 'Tests' tab. The test script area contains the following code:

```
1 pm.globals.set("StateID",responseBody);
2
3 pm.test("Status code is 200", function () {
4 pm.response.to.have.status(200);
5 });
6
7
```

On the right side, a list of snippets is shown, enclosed in a red box:

- Test scripts are written in JavaScript, and are run after the response is received. [Learn more about tests scripts](#)
- SNIPPETS
- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Send a request
- Status code: Code is 200

**Javascript** bilmiyor olabilirsiniz bu bizi korkutmasın ; **kod okuyabildiğimiz** için internette bulduklarımızı kendi işimize yarayacak şekilde kullanabiliriz.



# *POSTMAN'DE TANIMLAR*

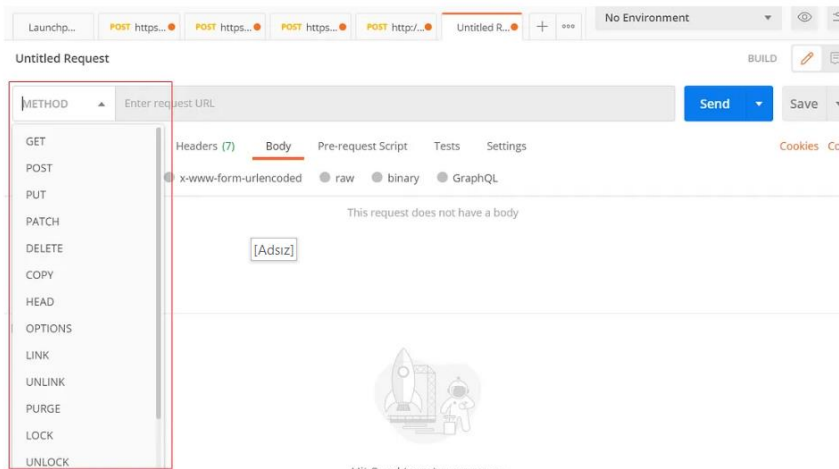




# POST- GET- PUT- DELETE

**CRUD**, Create (Oluştur), Read (Oku), Update (Güncelle), Delete (Sil) kelimelerinin baş harflerini temsil eder ve genellikle veritabanı işlemlerini açıklamak için kullanılır.

POSTMAN'de de bu işlemleri yapabilmek için bazı istek method türleri bulunmaktadır. Bunlardan en çok kullanılan **GET, POST, PUT, DELETE** metodlarıdır.





GET

1) **GET**: Sunucudan sadece veri çekmek(okuma) istiyorsak yani veri üzerinde herhangi bir değişiklik(ekleme, silme, modifiye) yapılmayacaksa GET metodunu kullanmamız tavsiye ediliyor.

- CRUD operasyonlarından Read'e karşılık geldiğini söyleyebiliriz.
- Ör: GET /students kullandığımızda bize öğrenciler listesini dönmesi.



GET http://localhost:20751/api/Category/getall Params Send

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (10) Test Results Status: 200 OK

Pretty Raw Preview JSON

```
1 [
2   {
3     "_name": "Beverages",
4     "_desc": "Soft drinks, coffees, teas, beers, and ales"
5   },
6   {
7     "_name": "Condiments",
8     "_desc": "Sweet and savory sauces, relishes, spreads, and seasonings"
9   },
10  {
11    "_name": "Confections",
12    "_desc": "Desserts, candies, and sweet breads"
13  },
14  {
15    "_name": "Dairy Products",
16    "_desc": "Cheeses"
17  },
18 ]
```





2) **POST**: Server Api'e body kısmını doldurarak ve veri üzerinde değişiklik yapmak istediğimizde kullanabiliriz

Değişiklik yapmak ile kastedilen CRUD operasyonlarından genel olarak Create kısmını kapsar.

**POST**

Ör: Post /createUser ile body kısmına kullanıcı bilgileri girip veritabanında bir kullanıcı oluşturulması istenmesi



POST `{{endpoint}}/sms?api-version=2021-03-07` Send

Params ☒ Authorization Headers (10) **Body** ☒ Pre-request Script Tests Settings ☒ Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   "from": "+1...",
3   "message": "Hello from ACS.",
4   "smsRecipients": [
5     {
6       "to": "+1..."
7     }
8   ]
9 }
```

Body Cookies Headers (8) Test Results Status: 202 Accepted Time: 824 ms Size: 561 B Save Response

Pretty Raw Preview Visualize **JSON** ≡

```
1 {
2   "value": [
3     {
4       "to": "+1...",
5       "messageId": "Outgoing 2021030900412857c6ffbb-f085-411e-...",
6       "statusCode": 202,
7       "successful": true
8     }
9   ]
10 }
```



### 3) **PUT:** Post isteğinin özelliklerine sahiptir.

**PUT**

Yani CRUD operasyonlarından Update operasyonlarını yapmak istediğimizde kullanıyoruz.

Post'dan ayrılan tarafı Put isteğinin idempotent ve not cacheable olarak tanımlanması



## RestfulBooker - Update Booking PUT

PUT

https://restful-booker.herokuapp.com/booking/10

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

```
"firstname": "James".
```

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

Visualize

JSON



1

2

3

4

5

6

7

8

9

10

11

```
"firstname": "James",  
"lastname": "Brown",  
"totalprice": 111,  
"depositpaid": true,  
"bookingdates": {  
  "checkin": "2018-01-01",  
  "checkout": "2019-01-01"  
},  
"additionalneeds": "Breakfast"
```

Response



**DELETE**

4) **DELETE**: CRUD operasyonlarından Delete'e karşılık gelir. Bir veriyi silmek istediğimizde kullanılması tavsiye ediliyor



File Edit View Help

New Import Runner

My Workspace Invite

No Environment

Filter

History Collections APIs

Save Responses Clear all

Today

- DEL http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- GET http://localhost:5000/api/tags/1006
- GET http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/users/1035
- GET http://localhost:5000/api/users/1035
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006
- PUT http://localhost:5000/api/tags/1006

Untitled Request

DELETE http://localhost:5000/api/tags/1006

Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   "tagId": 1006,
3   "tag": "red",
4   "postTags": []
5 }
```

Body Cookies Headers (4) Test Results

Status: 200 OK Time: 177 ms Size: 188 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "tagId": 1006,
3   "tag": "red",
4   "postTags": []
5 }
```

Find and Replace Console

Bootcamp Build Browse

Testing DELETE Request in Postman





## PARAMS-HEADER-BODY-PATH-SEND

Bir istek hazırladığımız zaman genellikle aktif olarak bu 5 alan kullanılıyor

- 1 numara ile gösterilen yer Api Url girdiğimiz yer.
- 2 numara ile gösterilen yer Params değerlerini tanımladığımız alan
- 3 numara ile gösterilen yer Headers değerlerini tanımladığımız alan
- 4 numara ile gösterilen yer Body kısmını dolduracağımız alan raw, binary gibi seçenekler mevcut
- 5 numara ile gösterilen yer isteğimizi hazırladığımızda Send'e basarak isteğimizi atabiliyoruz

Untitled Request

BUILD  

GET Enter request URL **1** **5** Send Save

**2** Params Authorization **3** Headers (6) **4** Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		



## HISTORY

Postman'nin sol tarafında bulunan History sekmesi önceden kullandığımız istekleri gün gün tutuyor Herhangi birine tıklayınca yeni sekmede tıklanılan istek açılıyor



History

Collections

APIs

☐ Save Responses

Clear all

m/oauth2/v3/token

POST https://oauth-login.cloud.huawei.com/oauth2/v3/token

POST https://oauth-login.cloud.huawei.com/oauth2/v3/token

POST https://oauth-login.cloud.huawei.com/oauth2/v3/token

POST https://oauth-login.cloud.huawei.com/oauth2/v3/token

POST https://oauth-login.cloud.huawei.com/oauth2/v3/token

> December 1

> November 30

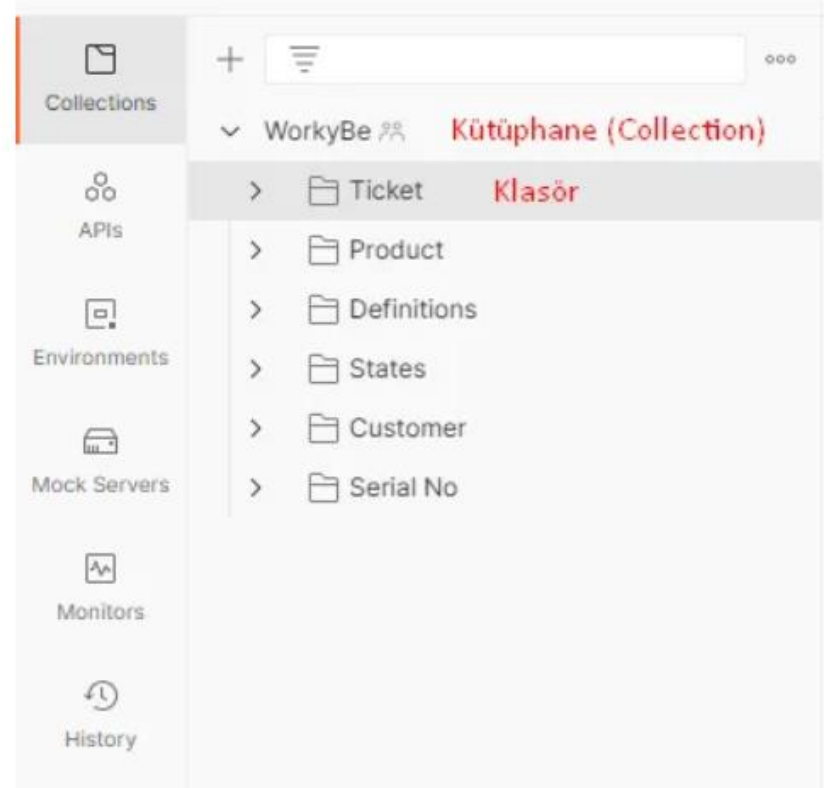
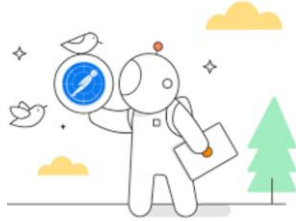
> November 27

✓ November 25



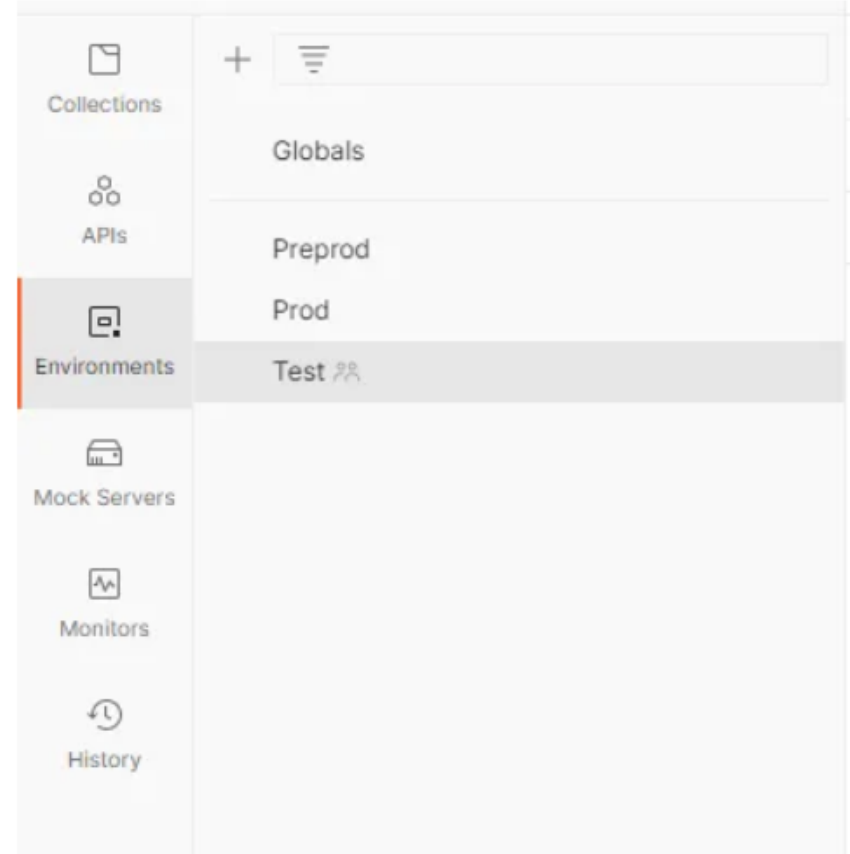


**Collection** : API sorgularını gruplamaya yardımcı kütüphanelerdir. Sorguları bir kütüphane içerisinde toplamak bize, kütüphaneyi paylaşırken, test ve dokümantasyonda kolaylık sağlar. Kütüphane altında birden fazla alt klasör yapısı kullanarak, API sorgularımızı daha düzenli bir şekilde gruplamamıza yardımcı olur.





**Environment** : En temelde her ürün bir test bir de canlı ortamdan oluşur. Bu ortamlara Pre prod ve dev de eklenebilir. Her ortamda değişen değişkenlerimiz vardır; url, kullanıcı adı ve şifre gibi. Farklı ortamlarda aynı testi çalıştırabilmek amacıyla, Postman Environment ile bu değişkenleri değiştirebilme olanağı sağlar.





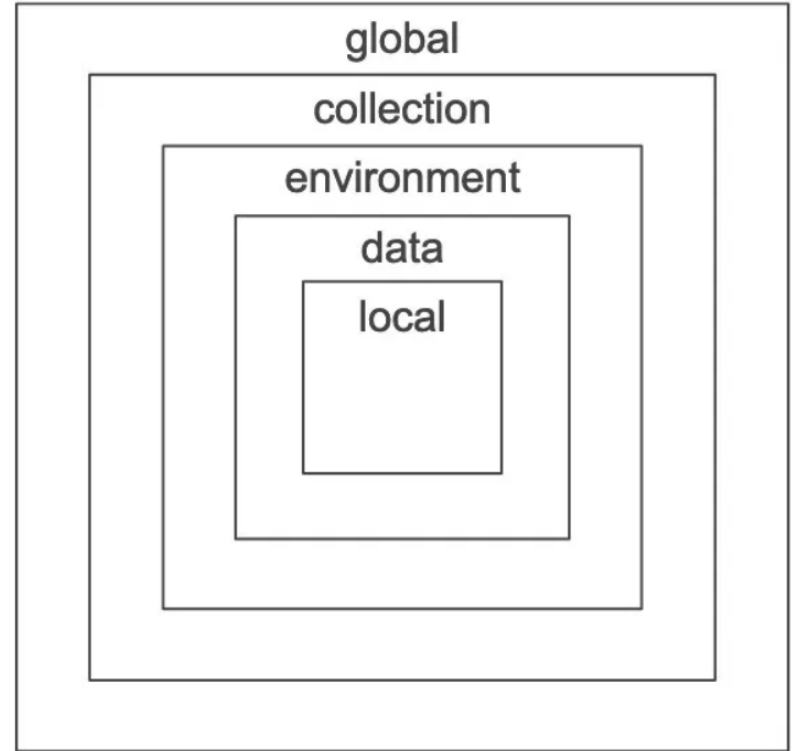
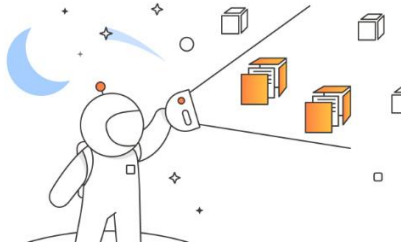
**Workspace** : Çalışma alanları, API çalışmalarınızı düzenlemenize ve ekip arkadaşlarınızla paylaşmanıza olanak sağlar. API projelerinizi birlikte düzenleyebileceğiniz bir çalışma alanı oluşturur. Çalışma alanındaki herkes, diğer ekip arkadaşlarının düzenledikleri ve eklediklerini görebilir ve müdahalede bulunabilir.

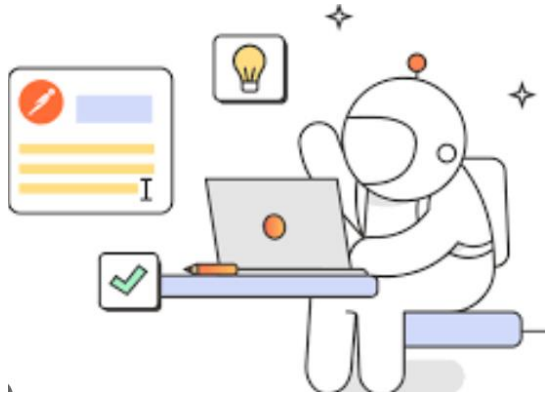




**Variables** : Variable'lar farklı requestlerde kullandığımız hep tekrar eden değişkinlerdir. Örneğin username ve password ya da token.

Değişkenleri; global, collection, environment data ve local olarak sınıflandırabiliriz.





Örnek vermek gerekirse, globaldeki bir değişken tüm requestler için geçerli olurken, environment'taki bir değişken sadece o environment için geçerli olacaktır. “url” diye bir değişken tanımladığımızı düşünürsek, bu url'i her environment'ta değiştirerek test, preprod ve prod ayırımını rahatlıkla yapabiliriz.



- **Global değişkenler**, workspace içerisinde tüm alanlarda geçerlidir. Collection, request, environments farketmez.
- **Collection değişkenleri**, Environment farketmeksizin tüm collection için geçerli değişkenleridir.
- **Environment değişkenleri**, oluşturmuş olduğunuz her environment için değişkenleri kullanmamızı sağlar. Özelleştirmelerde en kullanışlı değişken tanımlama şeklidir. Daha önceden de örneğini verdiğim gibi local, test, preprod, prod ortamlarının url'lerini ve giriş bilgilerini bu şekilde farklılaştırabiliriz.
- **Local değişkenler**, geçici değişkenlerdir. Sadece bir sorgu için ya da collection için kullanılabilir. Sorgu çağrımı bittiğinde kullanılmaz hale gelirler. Sadece bir defalık değişkeni değiştirmek istediğiniz durumlarda işe yararlar.
- **Data değişkenleri**, dışarıdan yüklenen CSV ve JSON dosyalarında data setleri ile çalıştırılırken kullanılır.

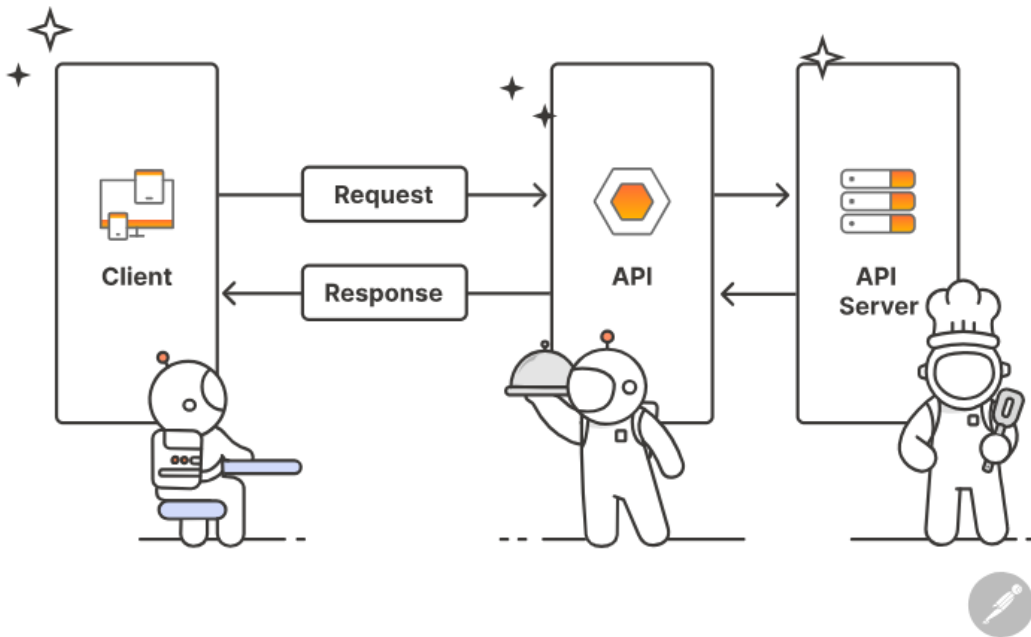


**POSTMAN** özetle ; bilinenin aksine API ler için sadece bir manuel test aracı değil aynı zamanda JavaScript kodlarıyla scriptler yazarak testleriniz için otomasyon da yapabileceğiniz bir tool'dur.





# API TEST CASE YAZIMI

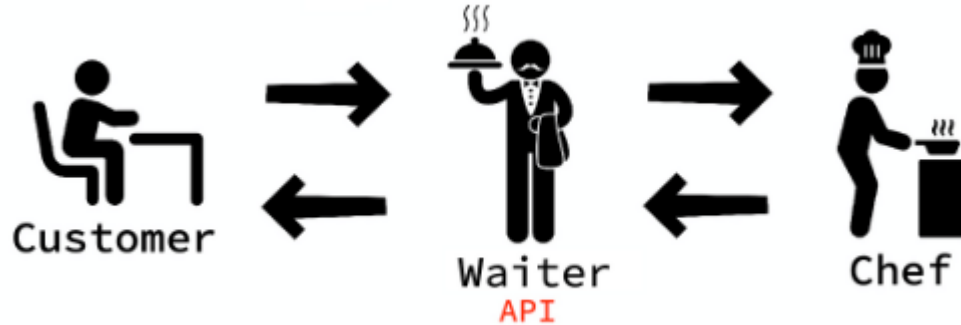






API test senaryoları, bir uygulamanın API'lerini test etmek ve doğru çalışıp çalışmadığını belirlemek amacıyla yazılır. Bu test senaryoları, *API'nin beklenen girdilere nasıl tepki vermesi gerektiğini ve doğru çıktıları üretip üretmediğini değerlendirir.*

İşte basit bir API test senaryosu örneği:





## Adım 1: Kullanıcı Oluşturma

**Amaç:** Yeni bir kullanıcı oluşturmak.

**HTTP Metodu:** **POST**

**Endpoint:** **/api/users**

**Girdi Parametreleri:**

- Kullanıcı Adı: "TestUser"
- E-Posta: "testuser@example.com"
- Şifre: "Test123"

**Beklenen Sonuçlar:**

- HTTP Durumu: 201 Created (Başarılı oluşturma)
- Yanıt Formatı: JSON
- Yanıt Mesajı: "Kullanıcı başarıyla oluşturuldu."

Kullanıcı kayıt işlemlerini yapan basit bir kullanıcı yönetimi API'sini test etmek istiyoruz.



1. Kullanıcı login olur ve token alır
2. Kullanıcı yeni kullanıcı için POST Request yapar
3. Status code nin 201 olduğunu doğrular
4. Content-Type ın JSON olduğunu doğrular
5. Kullanıcı gelen mesajın "Kullanıcı başarıyla oluşturuldu.» olduğunu doğrular



## Adım 2: Kullanıcı Getirme

**Amaç:** Oluşturulan kullanıcıyı getirmek.

**HTTP Metodu:** GET

**Endpoint:** `/api/users/{userID}`

(userID, önceki adımda oluşturulan kullanıcının ID'siyle değiştirilir.)

**Girdi Parametreleri:** Yok

**Beklenen Sonuçlar:**

- HTTP Durumu: 200 OK  
(Başarılı getirme)
- Yanıt Formatı: JSON
- Yanıt Mesajı: Kullanıcı bilgileri  
(ID, Kullanıcı Adı, E-Posta)

Kullanıcı getirme işlemlerini yapan basit bir kullanıcı yönetimi API'sini test etmek istiyoruz.



1. Kullanıcı login olur ve token alır
2. Kullanıcı yeni kullanıcı için GET Request yapar
3. Status code nin 200 olduğunu doğrular
4. Content-Type ın JSON olduğunu doğrular
5. Kullanıcı gelen kullanıcı datasını doğrular



**Amaç:** Varolan bir kullanıcının bilgilerini güncellemek.

**HTTP Metodu:** PUT

**Endpoint:** `/api/users/{userID}`

(userID, güncellenmek istenen kullanıcının ID'siyle değiştirilir.)

**Girdi Parametreleri:**

- Yeni Kullanıcı Adı: "YeniAd"
- Yeni E-Posta: "yenimail@example.com"

**Beklenen Sonuçlar:**

- HTTP Durumu: 200 OK
- Yanıt Formatı: JSON
- Yanıt Mesajı: "Kullanıcı bilgileri başarıyla güncellendi."

Kullanıcı bilgilerini güncelleme işlemlerini yapan basit bir kullanıcı yönetimi API'sini test etmek istiyoruz.



1. Kullanıcı login olur ve token alır
2. Kullanıcı güncelleme için PUT Request yapar
3. Status code nin 200 olduğunu doğrular
4. Content-Type ın JSON olduğunu doğrular
5. Kullanıcı gelen mesajın "Kullanıcı bilgileri başarıyla güncellendi.» olduğunu doğrular



**Amaç:** Varolan bir kullanıcıyı silmek.

**HTTP Metodu:** DELETE

**Endpoint:** `/api/users/{userID}`

(userID, silinmek istenen kullanıcının ID'siyle değiştirilir.)

**Girdi Parametreleri:** Yok

**Beklenen Sonuçlar:**

- HTTP Durumu: 204 No Content
- Yanıt Formatı: Yok  
(boş bir yanıt)

Kullanıcı silme işlemlerini yapan basit bir kullanıcı yönetimi API'sini test etmek istiyoruz.



1. Kullanıcı login olur ve token alır
2. Kullanıcı silme işlemi için DELETE Request yapar
3. Status code nin 204 olduğunu doğrular



Bu senaryolar bir okul sistemindeki öğrencilerin yada bir sisteme kayıtlı kişilerin API'sini test etmek için kullanılabilir. Her senaryo, belirli bir işlevi test eder ve API'nin beklenen sonuçları üreterek doğru çalışıp çalışmadığını kontrol eder. Gerçek uygulama senaryolarına ve gereksinimlerine göre daha karmaşık test senaryoları da oluşturulabilir.



### **Faydalanılan Kaynaklar**

<https://zeynepozay.medium.com/%C3%BC%C5%9Fenge%C3%A7ler-i%C3%A7in-postmande-api-testi-yazma-1-giri%C5%9F-f7fe38a475d0>

<https://chat.openai.com/>

<https://medium.com/huawei-developers-tr/postman-nedir-83eeaa5ed6ac>

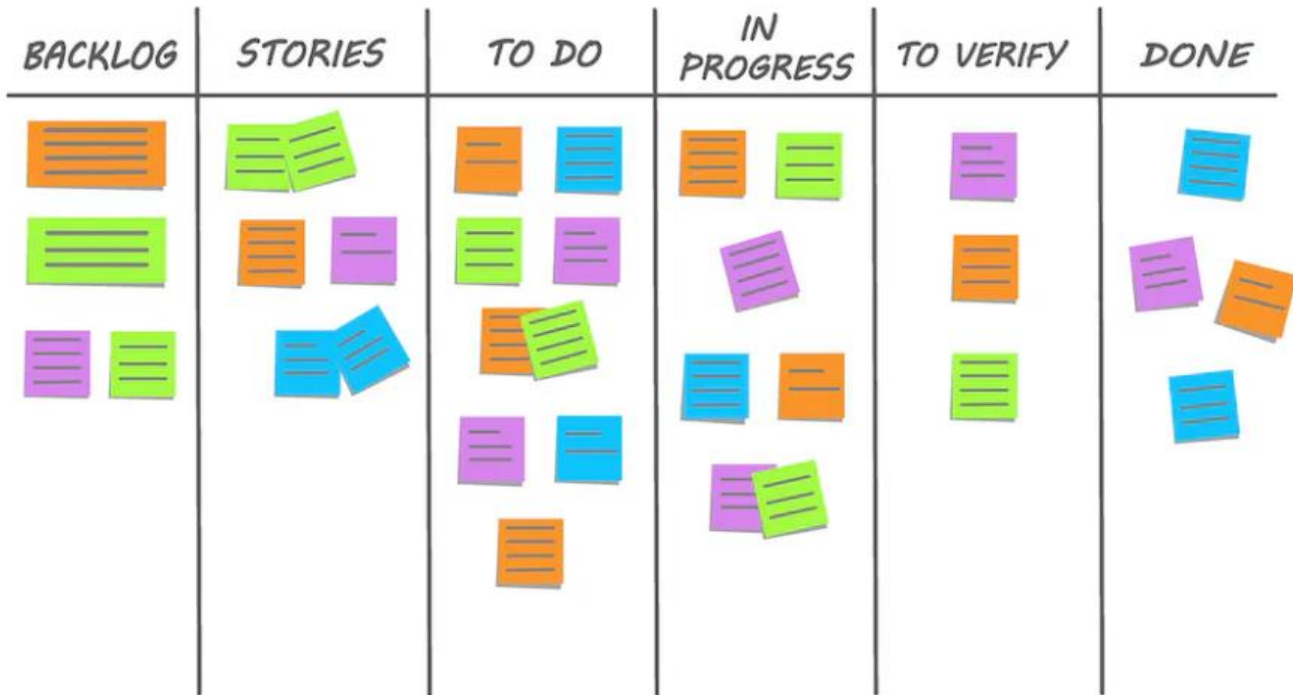
<https://katalon.com/resources-center/blog/test-cases-for-login-page>

### **Örnek Postman Kodları :**

<https://learning.postman.com/docs/writing-scripts/script-references/test-examples/>



# MENTORING TASKS







Verilen ek senaryo örnekleri için  
test case ve steplerini yazmaya çalışalım





## Senaryo 1: Ürün Listesi Alma

- **Amaç:** Tüm ürünleri listelemek.
- **HTTP Metodu:** **GET**
- **Endpoint:** **/api/products**
- **Girdi Parametreleri:** Yok
- **Beklenen Sonuçlar:**
  - HTTP Durumu: 200 OK
  - Yanıt Formatı: JSON
  - Yanıt Mesajı: Tüm ürünlerin listesi



## Senaryo 2: Ürün Oluşturma

- **Amaç:** Yeni bir ürün oluşturmak.
- **HTTP Metodu:** **POST**
- **Endpoint:** **/api/products**
- **Girdi Parametreleri:**
  - Ürün Adı: "Yeni Ürün"
  - Fiyat: 29.99
- **Beklenen Sonuçlar:**
  - HTTP Durumu: 201 Created
  - Yanıt Formatı: JSON
  - Yanıt Mesajı: "Ürün başarıyla oluşturuldu."



### Senaryo 3: Ürün Güncelleme

- **Amaç:** Var olan bir ürünü güncellemek.
- **HTTP Metodu:** PUT
- **Endpoint:** /api/products/{productID}  
(productID, güncellenmek istenen ürünün ID'siyle değiştirilir.)
- **Girdi Parametreleri:**
  - Ürün Adı: "Güncellenmiş Ürün"
  - Fiyat: 39.99
- **Beklenen Sonuçlar:**
  - HTTP Durumu: 200 OK
  - Yanıt Formatı: JSON
  - Yanıt Mesajı: "Ürün başarıyla güncellendi."



## Senaryo 4: Ürün Silme

- Amaç:** Var olan bir ürünü silmek.
- HTTP Metodu:** **DELETE**
- Endpoint:** **/api/products/{productID}**  
(productID, silinmek istenen ürünün ID'siyle değiştirilir.)
- Girdi Parametreleri:** Yok
- Beklenen Sonuçlar:**
  - HTTP Durumu: 204 No Content
  - Yanıt Formatı: Yok (boş bir yanıt)



# Mülakat Soruları





## REST tarafından desteklenen HTTP yöntemleri nelerdir?



REST HTTP tarafından desteklenen yöntemler şunlardır:

- GET - web sitelerinde ve API'lerde en yaygın olarak kullanılan yöntem olan GET, kaynakları belirli veri sunucusundan alır.
- POST - POST yöntemi aracılığıyla, kaynağı güncellemek için veriler API sunucusuna gönderilir. Bir sunucu verileri aldığı anda, onu HTTP istek gövdesinde saklar.
- PUT - kaynakları oluşturmak ve güncellemek için API'ye veri gönderir.
- DELETE - adından da anlaşılacağı gibi, bu yöntem belirli URL'lerdeki kaynakları silmek için kullanılır.



## PUT ve POST arasındaki fark nedir?

PUT ve POST arasındaki fark aşağıdaki gibidir:

- PUT - sağlanan (tek biçimli kaynak tanımlayıcı) URI'deki bir dosyayı veya kaynağı kesin ve özel olarak tanımlar. PUT, bu tek tip kaynak tanımlayıcısı - URI'de varsa, mevcut bir dosyayı değiştirir. PUT, zaten varsa bir dosya oluşturur. Ayrıca, PUT önemsizdir, bu da dosyaları henüz ne sıklıkta kullanıldığını etkilemediğini gösterir.

- POST - verileri ayrı bir tek tip kaynak tanımlayıcıya - URI'ye gönderir ve oradaki kaynak dosyasının talebi yönetmesini bekler. Şu anda, web sitesi sunucusu, seçilen dosya bağlamında verilerle ne yapılabileceğine karar verebilir. Ayrıca, POST stratejisi önemsiz değildir, bu da onu birden fazla kullanırsanız yeni dosyalar oluşturmaya devam edeceği anlamına gelir.







## HTTP Durum kodları nelerdir?

HTTP durumunda kullanılan standart kodlar, belirlenmiş sunucu görevi tamamlama durumlarına karşılık gelir. Örneğin, HTTP Durumu 404, sunucunun istenen kaynağa sahip olmadığını gösterir.



HTTP Status Codes		
Level 200	Level 400	Level 500
200: OK 201: Created 202: Accepted 203: Non-Authoritative Information 204: No content	400: Bad Request 401: Unauthorized 403: Forbidden 404: Not Found 409: Conflict	500: Internal Server Error 501: Not Implemented 502: Bad Gateway 503: Service Unavailable 504: Gateway Timeout 599: Network Timeout



HTTP durum kodlarına bakalım ve anlamlarını anlayalım:

- 200 - Tamam, başarı belli.
- 201 - bir POST veya PUT isteği başarıyla bir kaynak oluşturduğunda, yanıt kodu 201 - OLUŞTURULDU. Konum başlığını kullanarak URL'yi yeni oluşturulan kaynağa döndürün.
- 304 - koşullu GET istekleri durumunda, ağ bant genişliğinden tasarruf etmek için 304 DEĞİŞTİRİLMEDİ durum kodu kullanılır. Yanıt organları geçersiz olmalıdır. Tarihler, yerler ve diğer bilgiler başlıklarda olmalıdır.
- 400 - KÖTÜ İSTEK, eksik veri veya doğrulama hatası gibi geçersiz bir girişin sağlandığını gösterir.
- 401 - FORBIDDEN, kullanıcının, yönetici hakları olmadan erişimi silme gibi, kullanılan yönteme erişimi olmadığını belirtir.
- 404 - HATA, istenen yöntemin bulunamadığını belirtir.
- 409 - ÇATIŞMALAR Yöntem yürütüldüğünde, yinelenen girişler eklemek gibi çakışan bir sorunu belirtir.
- 500 - DAHİLİ SUNUCU HATA kodu, yöntem yürütülürken sunucunun bir istisna attığını gösterir.



## Postman Nedir ve Ne İşe Yarar?



- Cevap:* Postman, API'ları geliştirmek, test etmek ve belgelemek için kullanılan bir platformdur. API sorgularını yapmak, test senaryoları oluşturmak ve bu senaryoları otomatikleştirmek için kullanılır.



## Postman'de Collection Nedir?

- Cevap:* Bir Collection, bir dizi ilgili API isteğini ve bu isteklerin yapılandırmasını içeren bir grup. Bu, bir API'nin farklı özelliklerini ve senaryolarını düzenlemek ve yönetmek için kullanılır.





## Postman'de Test Script Nedir ve Ne İşe Yarar?



- Cevap:* Test script, bir API yanıtını değerlendirmek ve test etmek için kullanılan özel bir JavaScript kodudur. Bu kod, yanıtları doğrulamak ve test senaryolarını otomatikleştirmek için kullanılabilir.



## Postman Mock Servisleri Nedir?

- Cevap:* Postman Mock Servisleri, bir API'nin gerçek bir sunucu olmadan test edilmesini sağlayan, gerçekçi yanıtlar üreten bir hizmettir. Bu sayede geliştiriciler, API'lerini tasarlamadan önce test edebilirler.



## Postman'de Pre-request Script Nedir?



- Cevap:* Pre-request script, bir isteği göndermeden önce çalışan özel bir JavaScript kodudur. Bu kod, isteğin yapısını veya parametrelerini dinamik olarak ayarlamak için kullanılabilir.



# THANKS!

## Any questions?

