

# Uygulamalı Laboratuvar - CRUD işlemleri Python kullanarak



Gerekli Tahmini Süre: 45 dk

Bu laboratuvara, bir **Ürün listesi** oluşturmayı öğreneceksiniz. Uygulamanız, bir ürün eklemenize, ürünleri almanız, belirli bir ürünü kimliği ile almanız, belirli bir ürünü kimliği ile güncellemenize ve bir ürünü kimliği ile silmenize olanak tanıyacaktır. Tüm bu işlemler, Flask sunucunuzdaki REST API uç noktaları aracılığıyla gerçekleştirilecektir.

Yukarıdaki veriler üzerinde Create, Retrieve, Update ve Delete işlemleri gerçekleştirmek için API uç noktalarına sahip bir uygulama oluşturacaksınız.

Uygulanan uç noktaları test etmek için cURL ve POSTMAN kullanacaksınız.

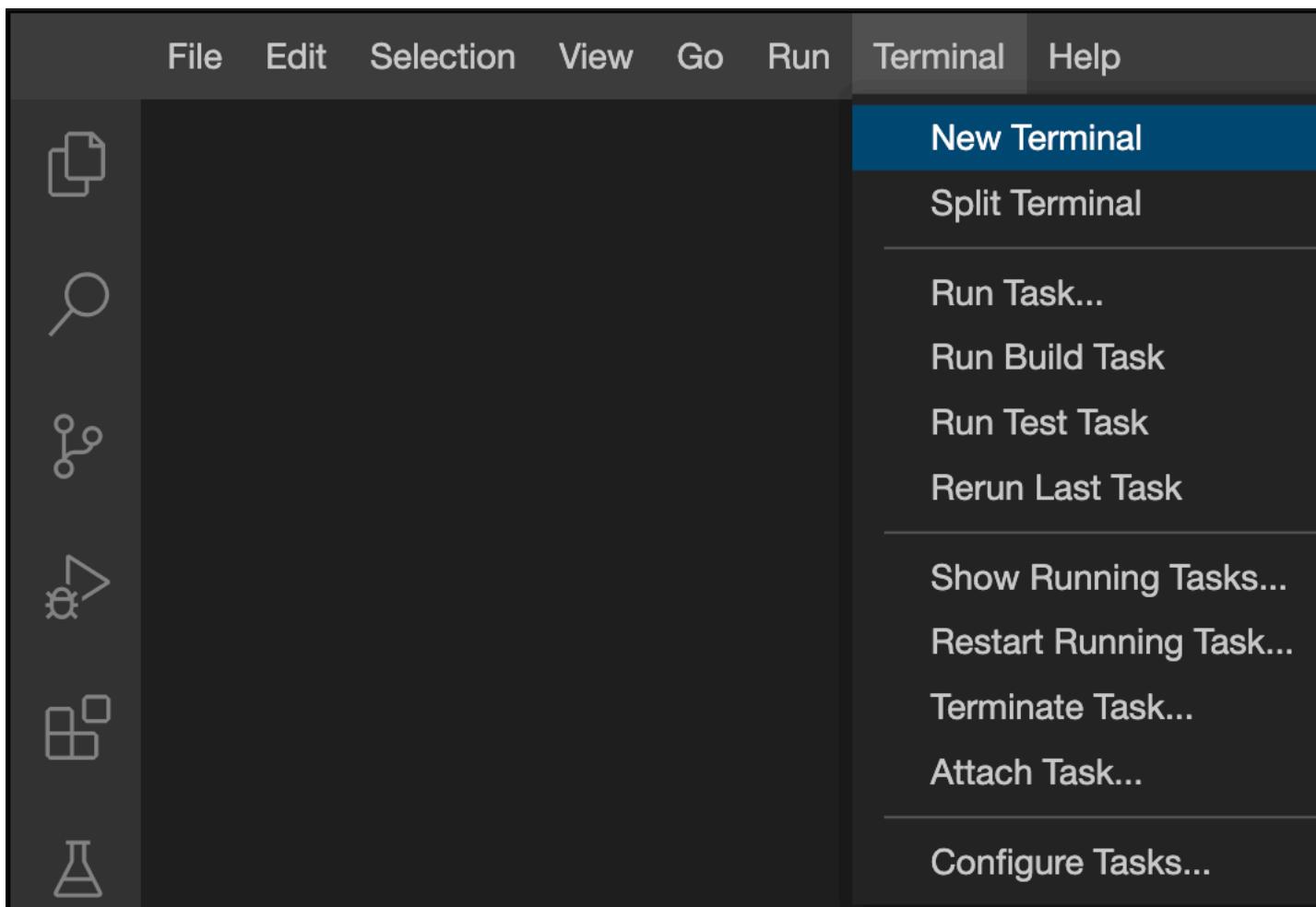
## Hedefler:

Bu laboratuvari tamamladıktan sonra şunları yapabileceksiniz:

- Flask sunucusu ile geçici veriler üzerinde Create, Retrieve, Update ve Delete işlemleri gerçekleştirmek için API uç noktaları oluşturmak.
- REST API uç noktaları oluşturmak ve REST API'lerinizi test etmek için POSTMAN kullanmak.

## Kurulum : Uygulama Oluştur

1. Editördeki menüyü kullanarak bir terminal penceresi açın: Terminal > Yeni Terminal.



2. Eğer proje klasöründe değilseniz, proje klasörünüze geçin.

```
cd /home/project
```

3. Bu laboratuvar için gerekli başlangıç kodunu içeren Git deposunu klonlamak için aşağıdaki komutu çalıştırın, eğer zaten mevcut değilse.

```
[ ! -d 'jmgdo-microservices' ] && git clone https://github.com/ibm-developer-skills-network/jmgdo-microservices.git
```

5. Laboratuvara başlamak için **jmgdo-microservices/CRUD** dizinine geçin.

```
cd jmgdo-microservices/CRUD
```

6. Bu dizinin içeriğini listeleyin, böylece bu laboratuvar için belgeleri görebilirsiniz.

```
ls
```

7. Gerekli paketleri yüklemek için terminalde aşağıdaki komutu çalıştırın.

```
python3 -m pip install flask flask_cors
```

## Egzersiz 1: Sunucu uygulamasını anlama

1. Dosya Gezgini’nde **jmgdo-microservices/CRUD** klasörünü açın ve **products.py** dosyasını görüntüleyin.

2. Öncelikle, Flask ile REST API’leri oluşturmak için gerekli paketleri içe aktarmanız gerekiyor.

```
from flask import Flask, jsonify, request
import json
```

3. Ardından, ürünleri eklemek, almak, güncellemek ve silmek için tüm REST API’lerini hizmet verecek Flask uygulamasını oluşturabilirsiniz.

```
app = Flask("Product Server")
```

4. Kod, listeye eklenmiş önceden oluşturulmuş ürünler içerir. Bunlar aşağıdaki kodda tanımlanmıştır.

```
products = [
    {'id': 143, 'name': 'Notebook', 'price': 5.49},
    {'id': 144, 'name': 'Black Marker', 'price': 1.99}
]
```

5. Sunucu tanımlandıktan sonra, REST API uç noktalarını oluşturacak ve her biri için yolları veya yolları tanımlayacaksınız, aşağıdaki işlemlerden biri için.

- Tüm ürünleri al - **GET** İstek Yöntemi
- Bir ürünü kimliği ile al - **GET** İstek Yöntemi
- Bir ürün ekle - **POST** İstek Yöntemi
- Bir ürünü kimliği ile güncelle - **PUT** İstek Yöntemi
- Bir ürünü kimliği ile sil - **DELETE** İstek Yöntemi

Aşağıdaki kodu products.py dosyasına verilen alan'a ekleyin.

```
# Example request - http://localhost:5000/products
@app.route('/products', methods=['GET'])
def get_products():
    return jsonify(products)
# Example request - http://localhost:5000/products/144 - with method GET
@app.route('/products/<id>', methods=['GET'])
def get_product(id):
    id = int(id)
    product = [x for x in products if x["id"] == id][0]
    return jsonify(product)
# Example request - http://localhost:5000/products - with method POST
@app.route('/products', methods=['POST'])
def add_product():
    products.append(request.get_json())
    return '', 201
# Example request - http://localhost:5000/products/144 - with method PUT
@app.route('/products/<id>', methods=['PUT'])
def update_product(id):
    id = int(id)
    updated_product = json.loads(request.data)
    product = [x for x in products if x["id"] == id][0]
    for key, value in updated_product.items():
        product[key] = value
    return '', 204
# Example request - http://localhost:5000/products/144 - with method DELETE
@app.route('/products/<id>', methods=['DELETE'])
def remove_product(id):
    id = int(id)
    product = [x for x in products if x["id"] == id][0]
    products.remove(product)
    return '', 204
```

## Sunucuyu cURL ile çalıştır ve test et

1. Terminalde, aşağıdaki komutu kullanarak python sunucusunu çalıştırın.

```
python3 products.py
```

Sunucu çalışmaya başlar ve 5000 numaralı portta dinlemeye başlar.

```
* Serving Flask app 'Product Server' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 145-923-897
```

2. Yeni Terminal seçenekini seçerek Terminal menüsünden başka bir Terminal açın.

3. Yeni terminalde, <http://localhost:5000/products> API uç noktasına erişmek için aşağıdaki komutu çalıştırın. curl komutu Client URL anlamına gelir ve REST API uç noktalarını sunan sunucu ile komut satırı arayüzü olarak kullanılır. Varsayılan olarak, bir GET isteğidir.

```
curl http://localhost:5000/products
```

Bu, önceden yüklenmiş ürünlerle birlikte bir JSON döndürür.

```
theia@theia-lavanyas:/home/project$ curl http://localhost:5000/products
[
  {
    "id": 143,
    "name": "Notebook",
    "price": 5.49
  },
  {
    "id": 144,
    "name": "Black Marker",
    "price": 1.99
  }
]
```

4. Terminalde, listeye bir ürün eklemek için aşağıdaki komutu çalıştırın. Bu, ürün parametresini JSON olarak geçeceğiniz bir POST isteği olacaktır.

```
curl -X POST -H "Content-Type: application/json" \
-d '{"id": 145, "name": "Pen", "price": 2.5}' \
http://localhost:5000/products
```

Bu komut herhangi bir çıktı döndürmeyecektir. Ürünü ürünler listesine ekleyecektir.

5. Ürünün eklenip eklenmediğini kontrol etmek için aşağıdaki komutu çalıştırın.

```
curl http://localhost:5000/products/145
```

Bu, bir POST isteği ile eklediğiniz ürünün detaylarını döndürmelidir.

```
theia@theia-lavanyas:/home/project$ curl http://localhost:5000/pro
{
  "id": 145,
  "name": "Pen",
  "price": 2.5
}
```

## POSTMAN ile REST API uç noktalarını test etme

GET uç noktaları, komut satırı arayüzü kullanarak `curl` ile test etmek kolayken, POST, PUT ve DELETE komutları zahmetli olabilir. Bu sorunu aşmak için, bir hizmet olarak sunulan Postman yazılımını kullanabilirsiniz.

Postman, harici bir varlık olduğu için sunucuya uzaktan erişim gerektirecektir. URL'yi almak için aşağıdaki butona tıklayın.

**Uygulamayı Başlat**

URL'yi bir not defterine veya başka bir metin düzenleyicisine kopyalayın. <https://www.postman.com/> adresine gidin ve POSTMAN'ı kullanmaya başlamadan önce kaydolun.

▼ Postman hizmetlerine kaydolmak için talimatlar için buraya tıklayın.

1. [www.postman.com](https://www.postman.com/) adresine gidin.
2. Kayıt işlemini başlatmak için **Kaydol** butonuna tıklayın.

Product ▾PricingEnterprise ▾Resources and Support ▾

E

# Build APIs together

Over 20 million developers use Postman. Get started by signing up or downloading the desktop app.

Sign Up for FreeDownload the desktop app

E-posta bilgilerinizi girip şifrenizi ayarlayarak kaydolabilir veya Google hesabınızla oturum açmayı tercih edebilirsiniz.

1. REST API uç noktasına bir istek oluşturmak için **Yeni Oluştur** butonuna tıklayın.

# Get started with Postman

## Start with something new

Create a new request, collection, or API in a workspace

[Create New →](#)

## Import an existing file

Import any API schema file from your local drive or Github

[Import file →](#)

2. Sunulan seçeneklerden **HTTP İsteğini** seçin.

## Create New

### Building Blocks



#### HTTP Request

Create a basic HTTP request



#### WebSocket Request BETA

Test and debug your WebSockets connections



#### Collection

Save your requests in a collection for reuse and sharing



#### Environment

Save values you frequently use in an environment

### Advanced



#### API Documentation

Create and publish beautiful documentation for your APIs



#### Mock Server

Create a mock server for your development APIs



#### API

Manage all aspects of API design, development, and testing



#### Flows BETA

Create API workflows by connecting series of requests through a drag-and-drop UI.

Not sure where to start? [Explore](#) featured APIs, collections, and workspaces published by the community.

3. Kopyaladığımız URL'yi adres çubuğuına yapıştırın. İstek türünü **POST** olarak değiştirin. JSON olarak girdi göndermek için **body->raw->JSON** seçeneğini seçin.

<https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/>

POST https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/

Params Authorization Headers (7) Body • Pre-request Script Tests S...

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

```
1 {
2   ...
3     "id":146,
4     "name":"Laptop Bag",
5     "price":45.00
}
```

4. Aşağıdaki JSON nesnesini girin ve Gönder butonuna tıklayın. Bu, ürünü önceki ürün listenize ekleyecektir.

```
{  
  "id":146,  
  "name":"Laptop Bag",  
  "price":45.00  
}
```

5. Ürünlerin listesini doğrulayın, böylece talebin geçtiğinden ve ürünün eklendiğinden emin olun. Talep türünü GET olarak değiştirin ve istek gövdesinden JSON nesnesini kaldırın. Gönder butonuna tıklayın ve yanıt penceresindeki çıktıyı gözlemleyin.

GET

https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.c

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Set

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

1

Body Cookies (1) Headers (10) Test Results

Pretty

Raw

Preview

Visualize

JSON



```
7   {
8     "id": 144,
9     "name": "Black Marker",
10    "price": 1.99
11  },
12  {
13    "id": 145,
14    "name": "Pen",
15    "price": 2.5
16  },
17  {
18    "id": 146,
19    "name": "Laptop Bag",
20    "price": 45.0
21  }
22 ]
```

5. Ürün detaylarını güncellemek için PUT uç noktasını test edin ve 146 numaralı ürünün fiyatını 42.00 olarak değiştirin. İstek türünü PUT olarak ayarlayın, URL'nin sonuna ürün kimliğini ekleyin ve değiştirilecek değeri JSON gövdesi olarak ekleyin, ardından gönder butonuna tıklayın.

```
{
  "price":42.00
}
```

<https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/p>

PUT ▼ <https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai/p>

Params Authorization Headers (8) **Body** • Pre-request Script Tests Schema

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

```
1 {  
2   ..... "price": 42.00  
3 }
```

6. İsteğin başarıyla geçtiğinden ve ürünün güncellendiğinden emin olmak için id'si 146 olan ürünü doğrulayın. İstek türünü GET olarak değiştirin ve istek gövdesinden JSON nesnesini kaldırın. Gönder butonuna tıklayın ve yanıt penceresindeki çıktıyı gözlemyein.

GET

https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.c

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Set

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

1

Body Cookies (1) Headers (10) Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "id": 146,  
3   "name": "Laptop Bag",  
4   "price": 42.0  
5 }
```

## Uygulama laboratuvarları

1. 144 numaralı ürünün fiyatını 2.50 olarak değiştirerek ürün detaylarını güncelleyin.

▼ İpucu için buraya tıklayın  
products/144 uç noktasına gidin. İstek türünü 'PUT' olarak ayarlayın ve URL'nin sonuna ürün kimliğini ekleyin, ardından değiştirecek değeri JSON gövdesi olarak ekleyin ve 'Gönder' butonuna tıklayın.

```
{  
  "price":2.50  
}
```

▼ Çözüm için buraya tıklayın

PUT



https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.c

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Set

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

```
1 {  
2   .... "price": 2.5  
3 }
```

2. Aşağıdaki ürünü POST yöntemiyle ekleyin.

```
{  
  "id":142,  
  "name":"Eraser",  
  "price":1.50  
}
```

▼ İpucu için buraya tıklayın

Önceki alıştırmanın 3. adımına bakın

3. 142 numaralı ürünü silin.

▼ İpucu için buraya tıklayın

products/142 uç noktasına gidin. İstek türünü 'DELETE' olarak ayarlayın ve 'Gönder' butonuna tıklayın.

▼ Çözüm için buraya tıklayın

DELETE



https://lavanyas-5000.theia-0-labs-prod-misc-tools-us-east-0.proxy.c

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Set

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

```
1
```

Tebrikler! Python ile CRUD işlemleri laboratuvarını tamamladınız.

**Özeti:**

Bu laboratuvara, bir Python sunucu uygulamasında GET, POST, PUT ve DELETE gibi CRUD işlemleri gerçekleştirdiniz ve yukarıdaki yöntemleri POSTMAN kullanarak test ettiniz.

## **Yazar(lar)**

Lavanya T S

© IBM Corporation. Tüm hakları saklıdır.