Uygulamalı Laboratuvar - Python Kullanarak REST API İçin Swagger Dokümantasyonu Oluşturma

Tahmini Süre: 45 dakika

Bu laboratuvarda, REST API'leriniz için nasıl Swagger dokümantasyonu oluşturacağınızı anlayacaksınız.

Öğrenme Hedefleri:

Bu egzersizi tamamladıktan sonra, aşağıdaki görevleri yerine getirebilmelisiniz:

- REST API için Swagger dokümantasyonu oluşturmak üzere Swagger Editor'ü kullanın
- Bir uygulamanın REST API uç noktalarına erişmek için SwaggerUI'yi kullanın
- Swagger dokümantasyonu ile kod oluşturun

Ön Koşullar

- Docker uygulamaları ve komutları hakkında bilgi sahibi olmalısınız
- REST API hakkında iyi bir anlayışa sahip olmalısınız.
- Python bilgisi şiddetle tavsiye edilir

Görev 1 - Uygulamanızı Başlatma

1. IDE'deki üst menüyü kullanarak bir terminal penceresi açın: **Terminal > Yeni Terminal**, eğer zaten açık değilse.

2. Terminalde, Swagger belgeleri ve REST API kodu hazır olan depoyu klonlamak için aşağıdaki komutu yapıştırın. Klonladığınız depo, görevleri organize etmek için kullanılabilecek bir REST API uygulamasını çalıştıracak koda sahiptir.

   ```
   git clone https://github.com/ibm-developer-skills-network/jmgdo-microservices.git
   ```

3. Çalışma dizinini **jmgdo-microservices/swagger_example** olarak değiştirmek için aşağıdaki komutu çalıştırın.

   ```
   cd jmgdo-microservices/swagger_example
   ```

4. Gerekli paketleri yüklemek için aşağıdaki komutları çalıştırın.

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

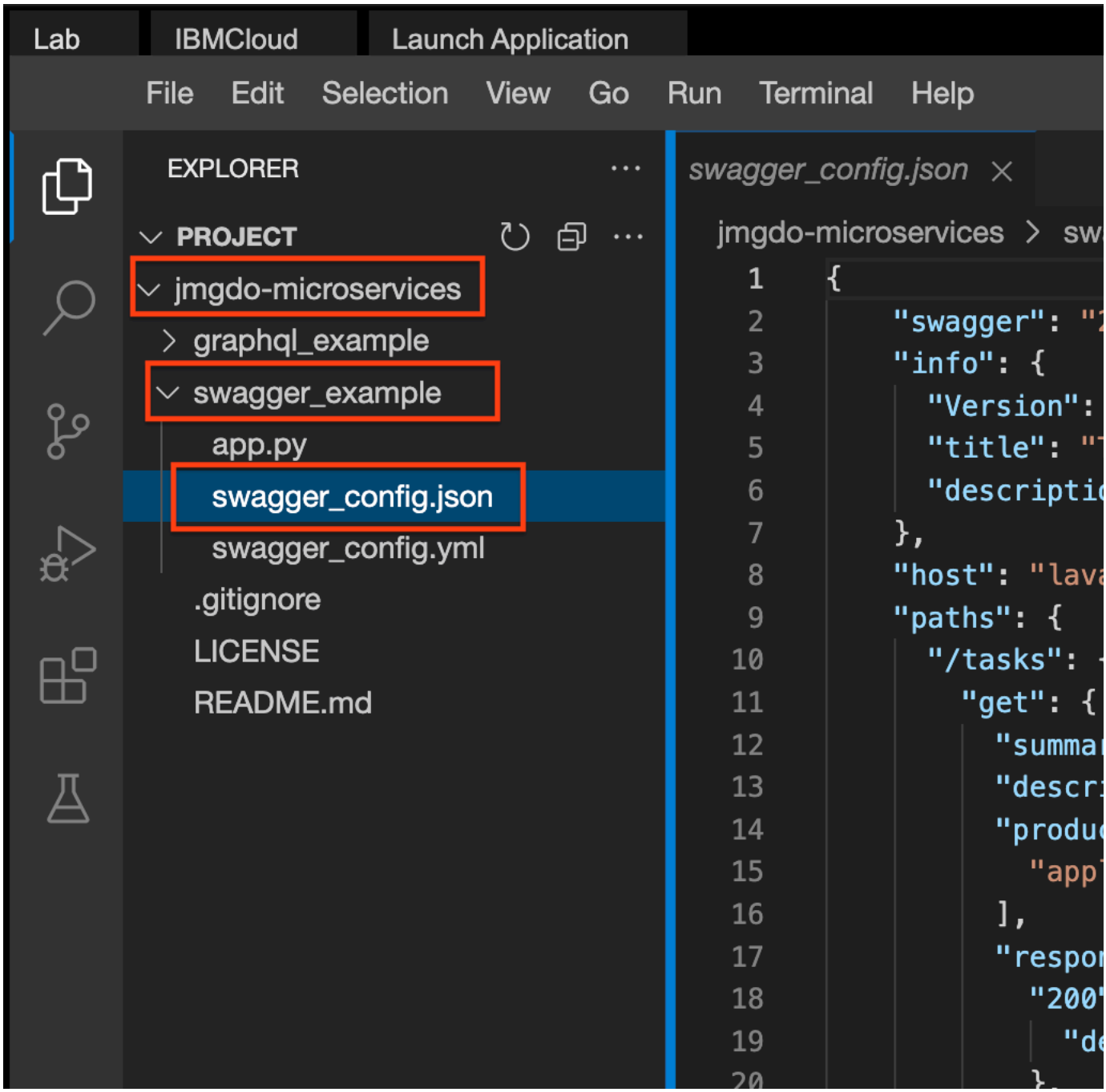
5. Şimdi 5000 numaralı portta REST API'yi sunan uygulamayı başlatın.

   ```
   python3 app.py
   ```

6. Soldaki Beceriler Ağı butonuna tıklayın, bu “Beceriler Ağı Araç Kutusu”nu açacaktır. Ardından Uygulamayı Başlat’a tıklayın, buradan port numarasını 5000 olarak girin ve Uygulamanız butonuna tıklayın. Bu, yeni bir tarayıcı sayfası açacak ve az önce çalıştırdığınız uygulamaya erişecektir.

7. Adres çubuğundaki URL’yi kopyalayın.

8. Dosya menüsünden jmgdo-microservices/swagger_example/swagger_config.json yoluna giderek dosyayı dosya düzenleyicisinde görüntüleyin.



9. Dosya düzenleyicisinde, kopyaladığınız uygulama URL’sini `**<Your application URL\>**` yazan yere yapıştırın, protokolü (https://) eklemeyin ve URL’nin sonuna "/" koymayın ve dosyayı kaydedin.

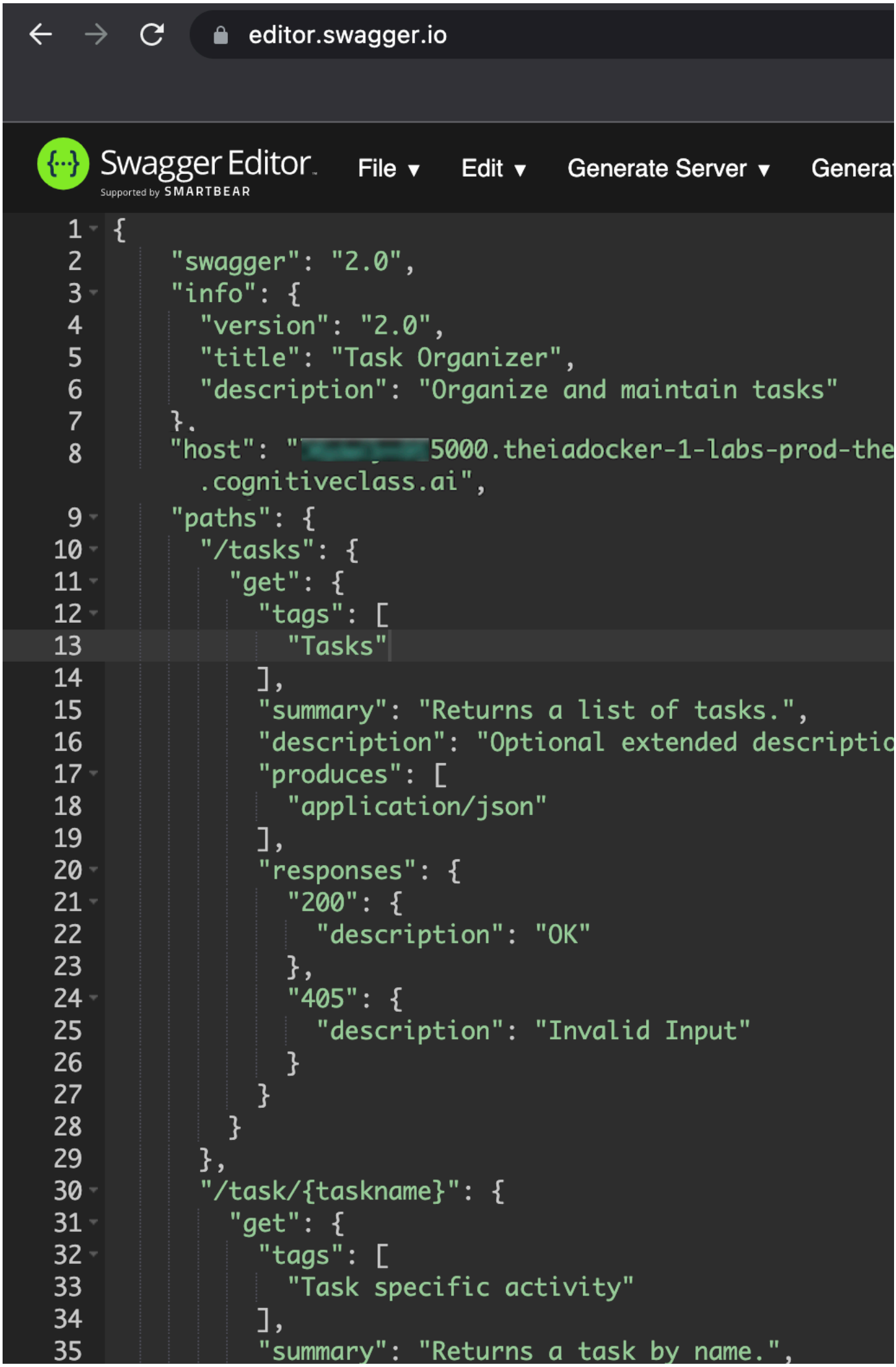
10. **swagger_config.json** dosyasının tamamını kopyalayın. SwaggerUI oluşturmak için bu kopyalanan içeriğe ihtiyacınız olacak.

11. Swagger Editörü’ne gitmek için bu bağlantıya tıklayın https://editor.swagger.io/.

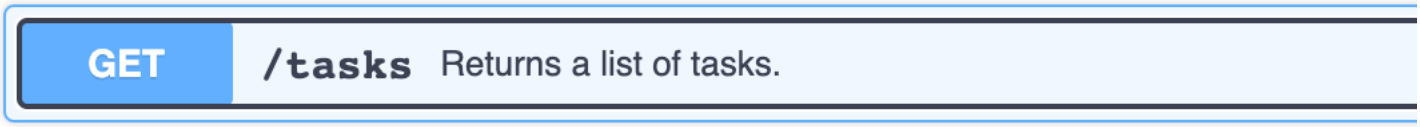
12. Dosya menüsünden, Swagger Editörü’nün içeriğini temizlemek için Düzenleyiciyi Temizle’ye tıklayın.

13. Sol tarafta kopyaladığınız **swagger_config.json** içeriğini yapıştırın. `JSON'unuzu YAML'ye dönüştürmek ister misiniz?` diyen bir istem alacaksınız. İçeriği yapıştırmak için `İptal` butonuna basın.

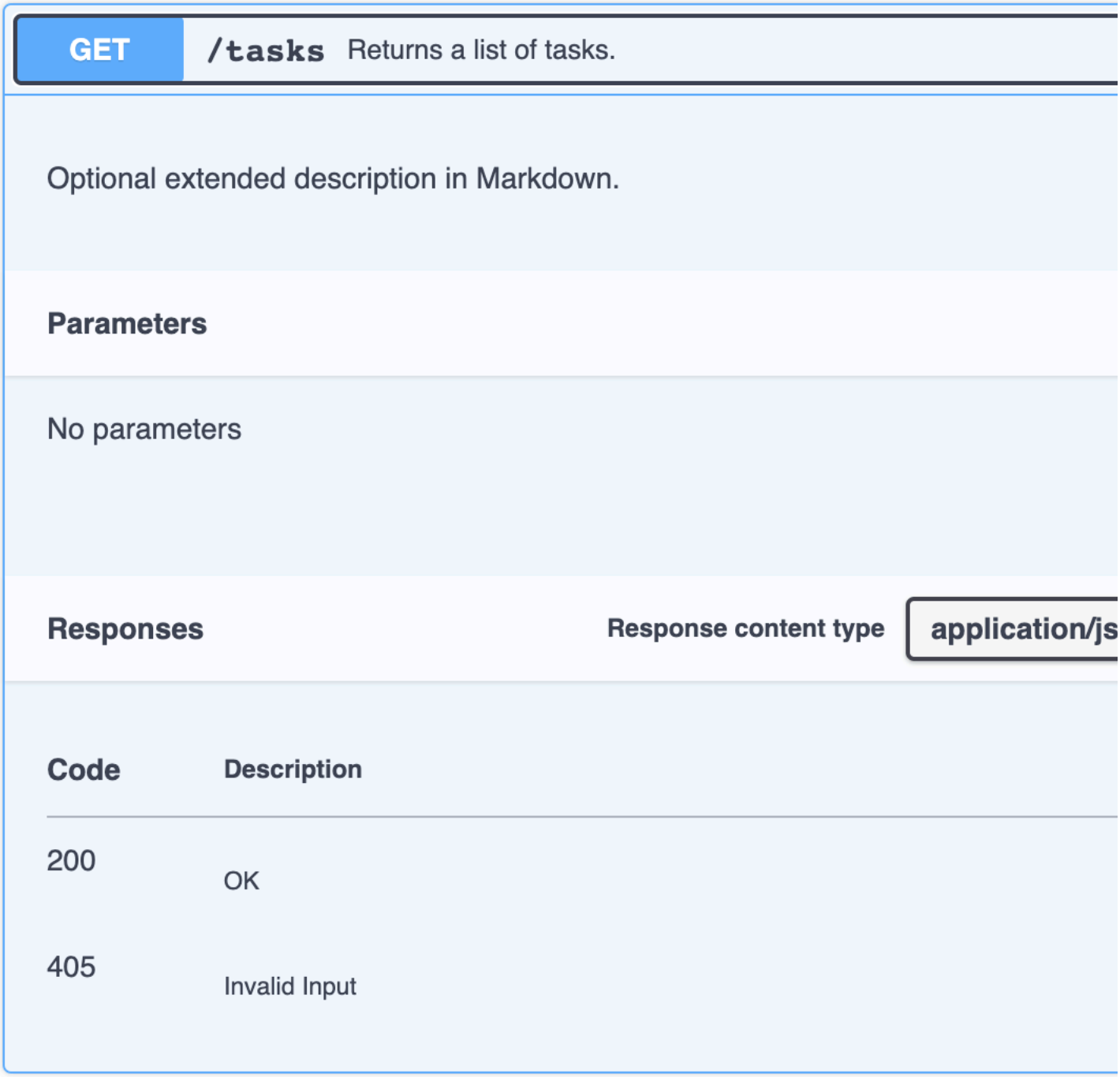
14. Sağ tarafta UI’nin otomatik olarak doldurulduğunu göreceksiniz.


editor.swagger.io
Swagger Editor
Supported by SMARTBEAR
File
Edit
Generate Server
Genera
{
"swagger": "2.0",
"info": {
"version": "2.0",
"title": "Task Organizer",
"description": "Organize and maintain tasks"
},
"host": "[redacted]5000.theiadocker-1-labs-prod-the
.cognitiveclass.ai",
"paths": {
"/tasks": {
"get": {
"tags": [
"Tasks"
],
"summary": "Returns a list of tasks.",
"description": "Optional extended descriptio
"produces": [
"application/json"
],
"responses": {
"200": {
"description": "OK"
},
"405": {
"description": "Invalid Input"
}
}
}
},
"/task/{taskname}": {
"get": {
"tags": [
"Task specific activity"
],
"summary": "Returns a task by name.",

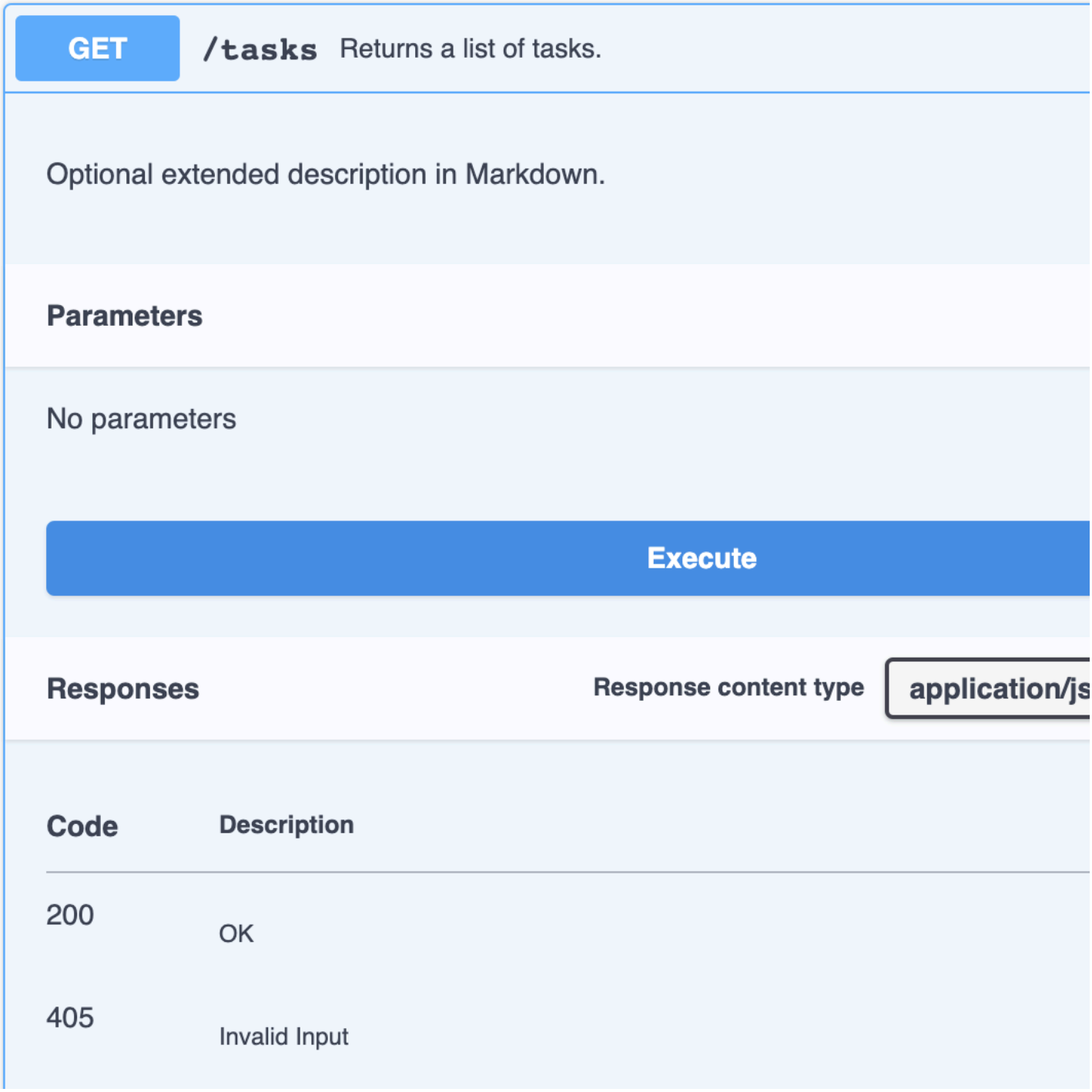
15. Artık her bir uç noktasını test edebilirsiniz. Uygulama başlatıldığında sizin için dört görev eklenmiştir. **GET /tasks**'ın yanındaki aşağı ok simgesine tıklayın.



16. **Try it out**'a tıklayın. Bu, REST API uç noktanızı denemenizi sağlar.



17. REST API'nize bir çağrı yapmak için **Execute**'a tıklayın. Bu, herhangi bir parametre almayan bir **GET** isteğidir. Görevi **application/json** formatında döndürür.



18. API çağrısının çıktısını görmek için aşağı kaydırabilirsiniz.

Curl

```
curl -X 'GET' \
  'https://lavanyas-5000.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cogni
  -H 'accept: application/json'
```

Request URL

```
https://lavanyas-5000.theiadocker-0-labs-prod-theiak8s-4-
tor01.proxy.cognitiveclass.ai/tasks
```

Server response

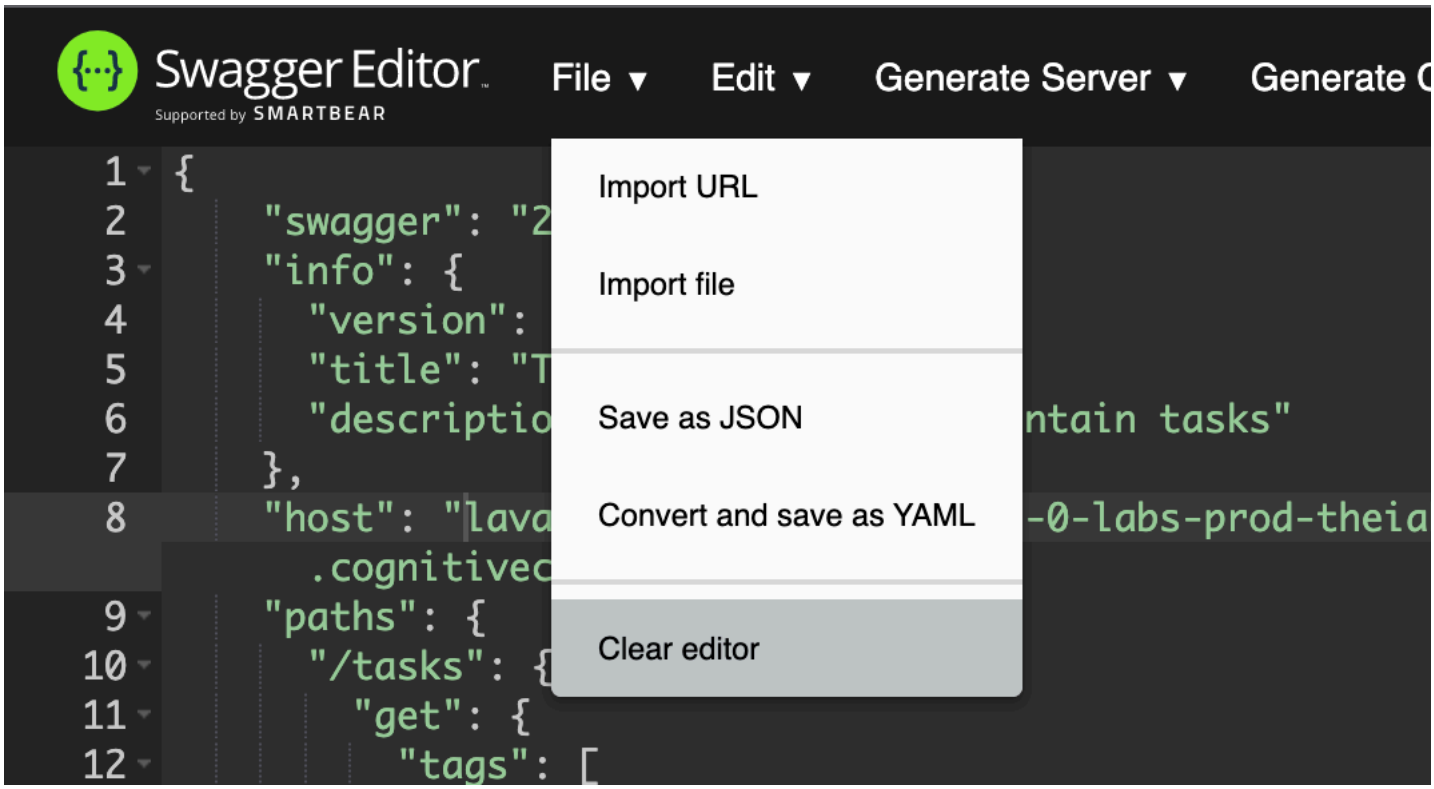
Code	Details
200	**Response body**


```
{
  "tasks": [
    {
      "description": "Do the laundry this weekend",
      "name": "Laundry"
    },
    {
      "description": "Finish assignment by Friday",
      "name": "Assignment"
    },
    {
      "description": "Call family Sunday morning",
      "name": "Call family"
    },
    {
      "description": "Pay the electricity and water bill",
      "name": "Pay bills"
    }
  ]
}
```


19. Aşağıdakileri yapmayı deneyin:

- Bir görev ekleyin
- Görevlerinizi almak için listeyi kontrol edin ve görevinizin eklenip eklenmediğini görün
- Bir görevin detaylarını alın
- Bir görevi silin ve silindiğini doğrulamak için listeyi kontrol edin.

20. **Edit** menüsünden **Clear Editor**'a tıklayarak Swagger Editor'ün içeriğini temizleyin.



Görev 2 - Docker ile Selamlaşma API'si Oluşturma ve Çalıştırma

Bu görevde, farklı dillerde selamlarla yanıt veren basit bir REST API sunucusu kuracaksınız ve bunun için önceden oluşturulmuş bir python-flask sunucu paketini kullanacağız.

1. Yeni bir terminal açın ve **/home/project** dizinine gidin.

   ```
   cd /home/project
   ```

2. Terminalde aşağıdaki komutu çalıştırarak Python-flask Sunucu Kodunu indirin.

   ```
   wget -O python-flask-server-generated.zip "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/me5rJsBIv8MNqFSyBk5g6Q/python-flas
   ```

3. İndirdiğiniz zip dosyasının var olup olmadığını kontrol edin.

   ```
   ls python-flask-server-generated.zip
   ```

4. Zip dosyasının içeriğini **python-flask-server-generated** adında bir dizine çıkarmak için aşağıdaki komutu çalıştırın.

   ```
   unzip python-flask-server-generated.zip -d python-flask-server-generated/
   ```

5. Zip dosyasını çıkardığınız klasörün içindeki **python-flask-server** klasörüne geçin.

```
cd python-flask-server-generated/python-flask-server-generated/python-flask-server-generated
```

6. Tüm sunucu ayarları ve uç nokta sizin için zaten yapılmış durumda. Şimdi sunucu kodunu oluşturalım.

```
docker build . -t mynewserver
```

Bu biraz zaman alır. Eğer derleme başarılı bir şekilde çalışırsa, mynewserver etiketine sahip yeni bir konteynere sahip olacaksınız.

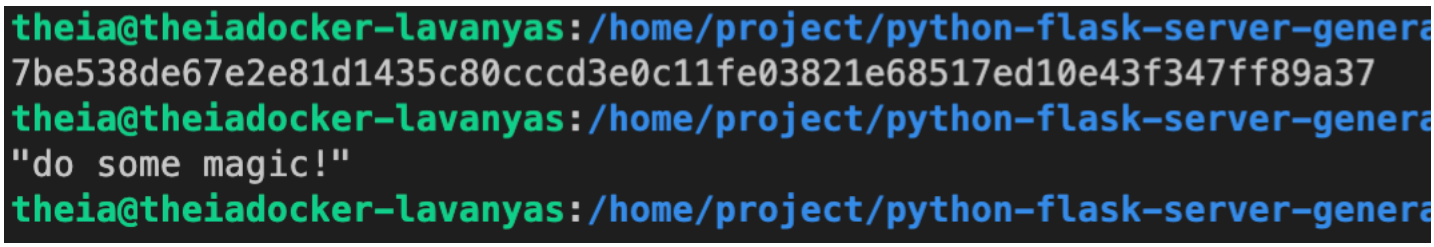
7. Docker uygulamasını şimdi aşağıdaki komutu çalıştırarak başlatın. Otomatik olarak oluşturulan sunucu kodu 8080 portunda çalışacak şekilde yapılandırılmıştır.

```
docker run -dp 8080:8080 mynewserver
```

Uygulamanın başladığını belirten bir hex kodu alacaksınız.

8. Servisin çalıştığını ve REST API'nizin çalıştığını doğrulamak için aşağıdaki komutu çalıştırın.

```
curl localhost:8080/greetings
```



Gördüğünüz çıktı, yapmanız gerekeni gösteriyor. **biraz sihir yapın!**

▶ Hata ile karşılaşırsanız ipucu için buraya tıklayın