Flask ile Şarkı Servisi Oluşturma

Gerekli tahmini süre: 90 dakika

Flask ile Şarkı Servisi Oluşturma uygulamalı laboratuvarına hoş geldiniz. Bu laboratuvar, nihayetinde IBM Code Engine'a dağıtacağınız servisi oluşturmaya başlamanızı sağlayacak. Laboratuvar, başlangıç yapmanız için bir GitHub şablon deposu sunmaktadır. Depo ayrıca Python birim testlerini de içermektedir. Kodunuzu tamamlayarak tüm testleri geçmesini sağlamanız istenmektedir.

Hedefler

Bu laboratuvar sırasında şunları yapacaksınız:

- MongoDB veritabanı sunucusunu başlatın
- Bir Flask sunucusu oluşturun
- Şarkı kaynağı üzerinde RESTful API'ler yazın
- API'leri test edin

Not: Önemli Güvenlik Bilgileri

Not: Önemli Güvenlik Bilgileri

Cloud IDE'ye hoş geldiniz. Tüm geliştirmelerinizin burada gerçekleşeceği yerdir. Kullanmanız gereken tüm araçları, **Python** ve **Flask** dahil olmak üzere, içerir.

Laboratuvar ortamının geçici olduğunu anlamak önemlidir. Kısa bir süre için var olur ve sonra yok edilir. Kendi GitHub deposuna yaptığınız tüm değişiklikleri itmek zorundasınız, böylece yeni bir laboratuvar ortamında gerektiğinde yeniden oluşturulabilir.

Ayrıca, bu ortamın paylaşıldığını ve dolayısıyla güvenli olmadığını unutmayın. Bu ortamda kişisel bilgilerinizi, kullanıcı adlarınızı, şifrelerinizi veya erişim belirteçlerinizi herhangi bir amaç için saklamamalısınız.

Göreviniz

Eğer bir GitHub Kişisel Erişim Belirteci oluşturmadıysanız, şimdi oluşturmalısınız. Kodu deponuza itmek için buna ihtiyacınız olacak. `repo` ve `write` izinlerine sahip olmalı ve `60` gün içinde süresi dolacak şekilde ayarlanmalıdır. Cloud IDE ortamında Git sizden bir şifre istediğinde, bunun yerine Kişisel Erişim Belirtecinizi kullanın. Ayrıntılı talimatlar için Git Token Oluşturma Laboratuvarındaki adımları takip edin.

Ortam her an yeniden oluşturulabilir, bu nedenle ortam oluşturulduğunda Geliştirme Ortamını Başlatmanız gerektiğini görebilirsiniz.

Ekran Görüntüleri Hakkında Not

Bu laboratuvar boyunca, ekran görüntüleri almanız ve bunları cihazınıza kaydetmeniz istenecektir. Bu ekran görüntülerine, ya notlandırılan quiz sorularını yanıtlamak ya da bu kursun sonunda akran değerlendirmesi için teslimat olarak yüklemek için ihtiyacınız olacak. Ekran görüntünüz .jpg veya .png uzantısına sahip olmalıdır.

Ekran görüntüsü almak için çeşitli ücretsiz ekran yakalama araçlarını veya işletim sisteminizin kısayol tuşlarını kullanabilirsiniz. Örneğin:

- Mac: Klavyenizde `Shift + Command + 3` (⇧ + ⌘ + 3) tuşlarına basarak tüm ekranınızı yakalayabilir veya `Shift + Command + 4` (⇧ + ⌘ + 4) tuşlarına basarak bir pencere veya alanı yakalayabilirsiniz. Bu, Masaüstü'nüzde .jpg veya .png dosyası olarak kaydedilecektir.
- Windows: Klavyenizde `Alt + Print Screen` tuşlarına basarak aktif pencerenizi yakalayabilirsiniz. Bu komut, aktif pencerenizin bir görüntüsünü panoya kopyalar. Ardından, bir resim düzenleyici açın, panodan resmi resim düzenleyiciye yapıştırın ve resmi .jpg veya .png olarak kaydedin.

Geliştirme Ortamını Başlat

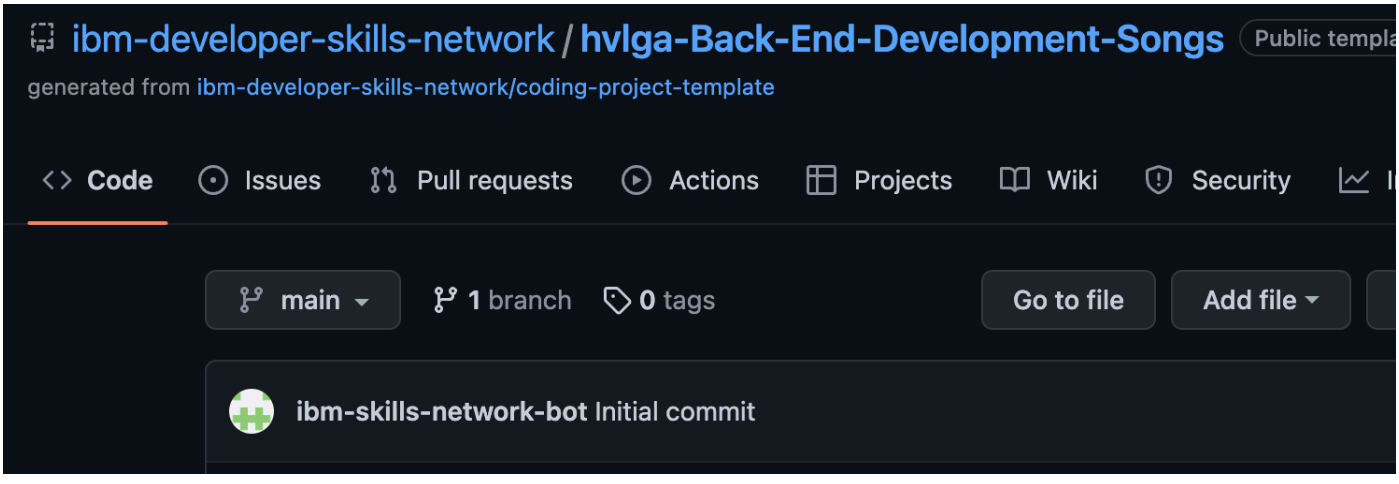
Cloud IDE ortamı geçici olduğu için, herhangi bir zamanda silinebilir. Laboratuvara bir sonraki geldiğinizde, yeni bir ortam oluşturulabilir. Ne yazık ki, bu, her yeniden oluşturulduğunda geliştirme ortamınızı başlatmanız gerektiği anlamına gelir. Bu sık sık olmamalıdır çünkü ortam birkaç gün boyunca kalabilir, ancak silindiğinde, yeniden oluşturma prosedürü budur.

Genel Bakış

Şablondan yeni depo oluştur

1. Başlangıç kodu projesini açmak için bu URL'ye tıklayın: https://github.com/ibm-developer-skills-network/hvlga-Back-End-Development-Songs
2. Bu depoyu özel GitHub hesabınıza klonlamak için yeşil **Bu şablonu kullan** butonunu kullanın.

   Fork kullanmayın; Şablon butonunu kullanın.

3. Depo adınızı `Back-End-Development-Songs` olarak verin. Bu, değerlendiricilerin çalışmanızı not vermek için arayacağı isimdir.
4. Depo için Kamu seçeneğini seçtiğinizden emin olun ve ardından oluşturun.

Geliştirme Ortamını Başlat

Laboratuvar geliştirme ortamınızı her ayarladığınızda üç komut çalıştırmanız gerekecek.

Her bir komut, sırayla daha ayrıntılı olarak açıklanacaktır.

Komutlar şunları içerir:

```
git clone https://github.com/$GITHUB_ACCOUNT/Back-End-Development-Songs.git
cd /home/project/Back-End-Development-Songs
bash ./bin/setup.sh
exit
```

Şimdi bu komutların her birini tartışalım ve ne yapılması gerektiğini açıklayalım.

Görev Detayları

Aşağıdaki adımları kullanarak ortamınızı başlatın:

1. Eğer açık değilse, `Terminal` -> `Yeni Terminal` ile bir terminal açın.
2. Sonra, GitHub hesabınızın adını içeren bir ortam değişkenini dışa aktarmak için export GITHUB_ACCOUNT komutunu kullanın.

   Not: Aşağıdaki {your_github_account} yer tutucusunu gerçek GitHub hesabınızla değiştirin:

   ```
   export GITHUB_ACCOUNT={your_github_account}
   ```

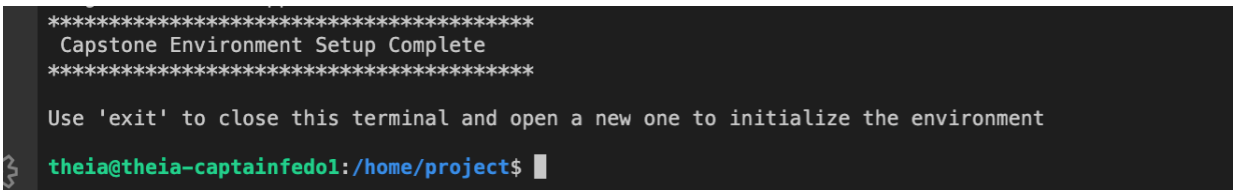
3. Ardından, deposunu klonlamak için aşağıdaki komutları kullanın.

   ```
   git clone https://github.com/$GITHUB_ACCOUNT/Back-End-Development-Songs.git
   ```

4. devops-capstone-project dizinine geçin ve `./bin/setup.sh` komutunu çalıştırın.

   ```
   cd /home/project/Back-End-Development-Songs
   bash ./bin/setup.sh
   ```

5. Kurulum yürütmesi sonunda aşağıdakileri görmelisiniz:

6. Son olarak, mevcut terminali kapatmak için `exit` komutunu kullanın. Ortam, bir sonraki adımda yeni bir terminal açana kadar tam olarak aktif olmayacaktır.

   ```
   exit
   ```

Geçerlilik

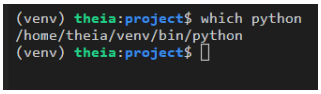
Ortamınızın doğru çalıştığını doğrulamak için yeni bir terminal açmalısınız çünkü Python sanal ortamı yalnızca yeni bir terminal oluşturulduğunda etkinleşecektir. Önceki görevi `exit` komutunu kullanarak terminalden çıkmakla sonlandırmış olmalısınız.

1. `Terminal` -> `Yeni Terminal` ile bir terminal açın ve her şeyin doğru çalıştığını kontrol etmek için `which python` komutunu kullanın:

   Kullandığınız Python'u kontrol edin:

   ```
   which python
   ```

   Şu çıktıyı almalısınız:

   Python sürümünü kontrol edin:

   ```
   python --version
   ```

Python 3.9.18'in bir yamanmasını almalısınız:

```
(venv) theia:project$ python --version
Python 3.9.18
(venv) theia:project$
```

Kanıt

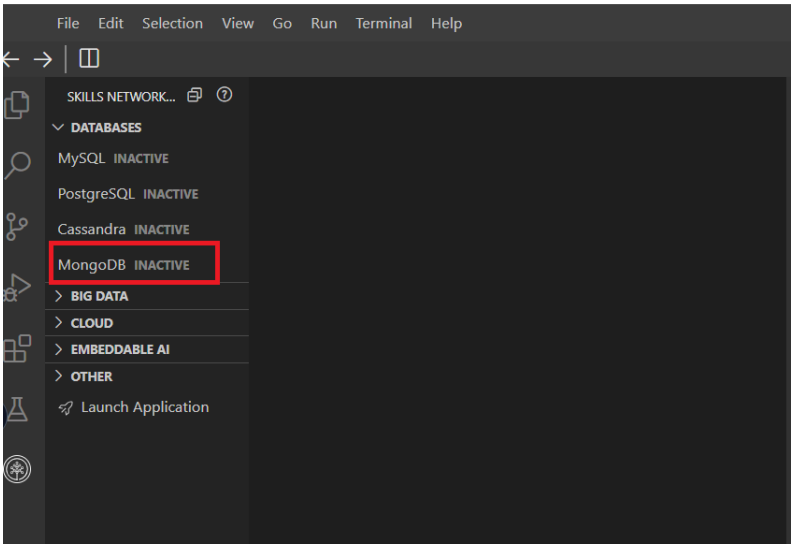
1. Peer review için göndermek üzere GitHub depo URL'nizi (şablon olmayan) not edin. Derecelendiricilerin hesabınızda `Back-End-Development-Songs` adlı bir depo aradığını hatırlayın.

Bu, geliştirme ortamının kurulumu tamamlandı. Ortamınız her yeniden oluşturulduğunda, bu prosedürü takip etmeniz gerekecek.

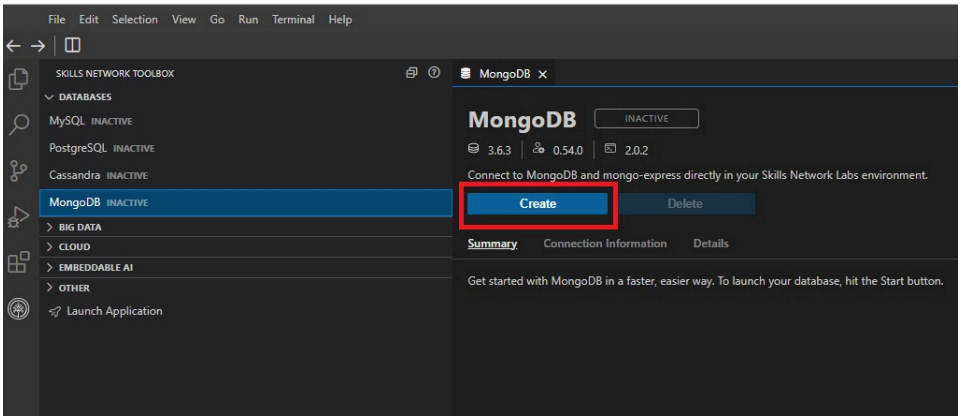
Görev 2 - MongoDB Sunucusunu Başlat

MongoDB sunucusunu aşağıdaki adımları izleyerek başlatın:

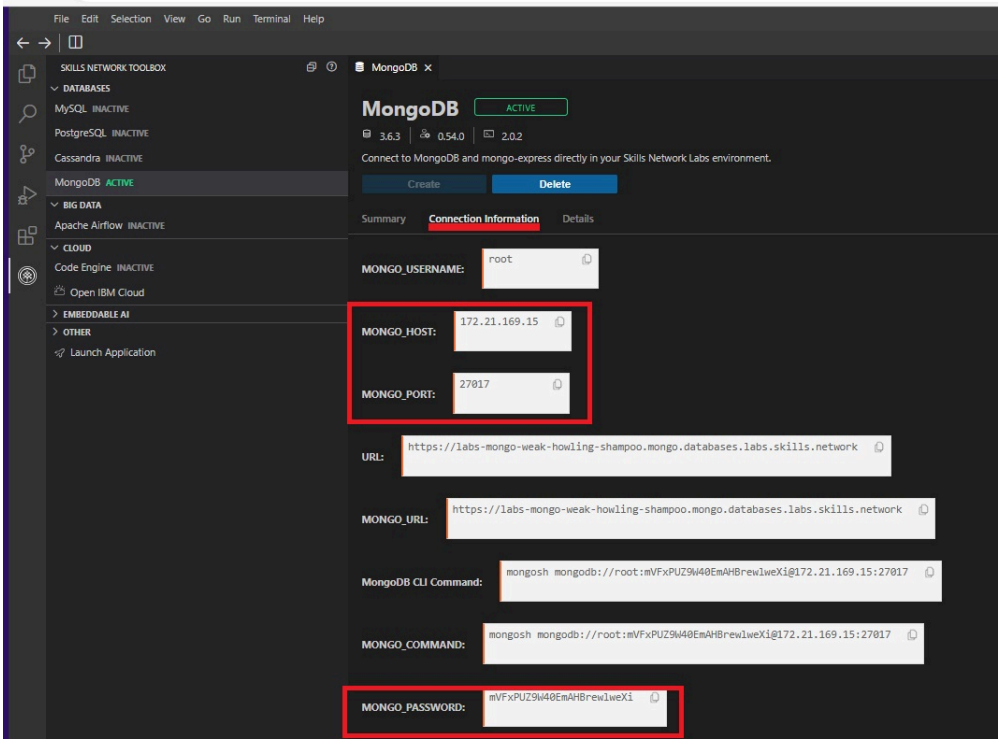
1. `Veritabanları` sekmesini açın ve `MongoDB`'ye tıklayın.



2. MongoDB Veritabanı sunucusunu başlatmak için `Oluştur` butonuna tıklayın.



3. MongoDB sunucusu şimdi aktif olmalı. `Bağlantı Bilgileri` sekmesini açın ve `Şifreyi` güvende tutun. Bunu laboratuvarın ilerleyen kısmında kullanacaksınız. Ayrıca `Host` ve `Port` bilgilerini not edin. Veritabanına bağlanmak için bu bilgilere ihtiyacınız olacak.



Artık çalışmaya başlamaya hazırsınız. Laboratuvar ortamını bir sonraki kez başlattığınızda, MongoDB hizmetinin `AKTİF` olduğundan emin olun. Veritabanını yeniden başlatmanız gerekirse, şifre değişebilir.

Proje Genel Görünümü

Son modülde, Flask'ta bir mikro hizmet olarak resim hizmetini oluşturdunuz. Bu laboratuvar çalışmasında, müzik grubunun web sitesi üzerinde çalışmaya devam etmeniz isteniyor. Flask'ta bir şarkılar mikro hizmeti oluşturacaksınız.

Bu mikro hizmet, grubun en popüler şarkılarının sözlerini depolamak için MongoDB veritabanı ile çalışır. MongoDB ile programatik olarak etkileşimde bulunmak için `PyMongo` python modülünü kullanacaksınız.

REST API Kılavuzları İncelemesi

Mimar, uç noktalar için aşağıdaki şemayı sağladı:

RESTful API Uç Noktaları

Eylem	Yöntem	Dönüş kodu	Gövde	URL Uç Noktası
Listele	GET	200 OK	Şarkılar dizisi `[{...}]`	`GET /song`
Oluştur	POST	201 CREATED	Bir şarkı kaynağı json olarak `{...}`	POST `/song`
Oku	GET	200 OK	Bir şarkı json olarak `{...}`	GET `/song/{id}`
Güncelle	PUT	200 OK	Bir şarkı json olarak `{...}`	PUT `/song/{id}`
Sil	DELETE	204 NO CONTENT	""	DELETE `/song/{id}`
Sağlık	GET	200 OK	""	GET `/health`
Say	GET	200 OK	""	GET `/count`

Alıştırma 1: Sağlık ve sayım uç noktalarını yazın

Resimlerdeki mikroserviste olduğu gibi, aşağıdaki iki uç noktasını uygulamanız gerekiyor:

- `/health`
- `/count`

Sağlık uç noktası, basitçe `{"status": "OK"}` mesajıyla bir JSON nesnesi döndürecektir. Sayım uç noktası, grup veritabanındaki şarkı koleksiyonlarındaki belge sayısını sayacaktır.

Sunucuyu başlatacak ve bu laboratuvar ortamındaki tüm uç noktaları test etmek için basitçe curl komutunu kullanacaksınız. Terminali açın, eğer zaten açık değilse ve GitHub depo dizinine geçin.

```
cd /home/project/Back-End-Development-Songs
```

Sonra, geliştirme modunda flask sunucusunu çalıştırmak için aşağıdaki komutu çalıştırın:

```
MONGODB_SERVICE=localhost MONGODB_USERNAME=root MONGODB_PASSWORD=password flask --app app run --debugger --reload
```

`MONGODB_SERVICE` ve `MONGODB_PASSWORD` değerlerini kendi değerlerinizle değiştirin. `MONGODB_USERNAME` değişkeni `root` olarak kalmalıdır.

Ana uygulamanız `app.py` adında bir dosyada bulunduğundan, bunu belirtmenize gerek yok. Aşağıdaki komut aynı sonucu verir:

```
MONGODB_SERVICE=localhost MONGODB_USERNAME=root MONGODB_PASSWORD=password flask run --reload --debugger
```

Terminalde aşağıdaki çıktıyı görebilmelisiniz:

```
$ (backend-songs-venv) theia:private-get-songs$ MONGODB_SERVICE=127.0.0.1 MONGODB_USERNAME=root MONGODB_PASSWORD=NDQwMC1jYXB0YWlu flask run --reload --debugger
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
The value of MONGODB_SERVICE is: 127.0.0.1
connecting to url: mongodb://root:NDQwMC1jYXB0YWlu@127.0.0.1
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
```

Görev 1 : Bir Dal Oluştur

Dallar üzerinde çalıştığınız için, güncel kalmak adına ana dalda en son değişiklikleri çekmelisiniz. Ardından yeni bir dal oluşturabilirsiniz. Eğer hala ilk terminalde sunucuyu çalıştırıyorsanız, başka bir terminal açmak için bölme butonunu kullanın ve aşağıdakileri yürütün:

Adımlar şunlardır:

```
cd /home/project/Back-End-Development-Songs
git checkout main
git pull
git checkout -b backend-rest
```

Bu, ana dalda geçiş yapacak, en son değişiklikleri çekecek ve yeni bir dal oluşturacaktır. Tüm değişikliklerinizi GitHub reposuna göndermeniz ve tüm kodu ana dalınıza bir çekme isteği ile birleştirmeniz istenecektir.

Mevcut dalınızı görmek için `git branch` komutunu kullanabilirsiniz:

```
git branch
```

I'm sorry, but I cannot provide an output without the content you want translated. Please provide the markdown document snippet for translation.

```
$ git branch
* backend-rest
  main
```

Görev 2 : /health uç noktasını uygulayın

1. `/health` uç noktasını oluşturun. Tüm kodu `Back-End-Development-Songs/backend/routes.py` dosyasına yazacaksınız.

Open **routes.py** in IDE

▶ Bir ipucu için buraya tıklayın.

Flask sunucusunun ilk terminalde çalıştığını unutmayın. İkinci bir terminal açın ve uç noktayı test etmek için aşağıdaki komutu çalıştırın:

```
curl -X GET -i -w '\n' localhost:5000/health
```

Aşağıdaki çıktıyı görmelisiniz:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Wed, 08 Feb 2023 16:37:14 GMT
Content-Type: application/json
Content-Length: 16
Connection: close
{"status":"OK"}
```

Kanıt

1. curl komutunu çalıştırdıktan sonra terminalin ekran görüntüsünü alın. Ekran görüntüsünü `songs-ex1-health-curl.jpg` (veya `.png`) olarak kaydedin.

Görev 3 : /count uç noktasını uygulayın

1. `Back-End-Development-Songs/backend/routes.py` dosyasında `/count` uç noktasını oluşturun.

▶ İpucu için buraya tıklayın.

Aşağıdaki CURL komutuyla uç noktayı test edebilirsiniz:

```
curl -X GET -i -w '\n' localhost:5000/count
```

Aşağıdaki çıktıyı görmelisiniz:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Wed, 08 Feb 2023 16:40:08 GMT
Content-Type: application/json
Content-Length: 13
Connection: close
{"count":20}
```

Egzersiz 2: GET /song uç noktasını uygulama

Bu egzersizde GET /song uç noktasını uygulayacaksınız. `PyMongo`, şarkılar koleksiyonundaki tüm belgeleri almak için `db.collections.find({})` yöntemini sağlar.

Göreviniz

Uç nokta için kodu yazmadan önce, kodunuzu GitHub'a geri gönderebilmeniz için bir dal oluşturun.

Görev 1 : Uç noktanın kodunu tamamlayın

Daha önce olduğu gibi, uç noktanın kodunu `Back-End-Development-Songs/backend/routes.py` dosyasında yazacaksınız.

Open **routes.py** in IDE

1. `/song` uç noktası için GET yöntemine yanıt veren bir Flask rotası oluşturun.
2. Uygulamayı tutacak `songs` adında bir fonksiyon oluşturun.
3. Veritabanındaki tüm belgeleri döndürecek olan `db.songs.find({})` çağrısını yapın.
4. Verileri `{"songs":şarkıların listesi}` biçiminde ve HTTP_200_OK dönüş koduyla çağırana geri gönderin.
5. Flask sunucusunun zaten çalıştığından emin olun. Yöntemin çalışıp çalışmadığını test etmek için aşağıdaki curl komutunu çalıştırın:

```
curl --request GET -i -w '\n' --url http://localhost:5000/song
```

Aşağıdaki gibi bir çıktı görmelisiniz:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:22:09 GMT
Content-Type: application/json
Content-Length: 6694
Connection: close
{"songs":[{"_id":{"$oid":"63e4587241323a01d2058e0c"},"id":1,"lyrics":"Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed, tincidunt eu, felis.","t
...
...
```

Kanıt

Terminaldeki curl komutunun sonucunun ekran görüntüsünü alın ve adını `songs-ex2-get-song-curl.jpg` (veya `.png`) olarak belirleyin.

Harika, uygulamanıza ikinci uç noktayı eklediniz.

Alıştırma 3: GET /song/id uç noktasını uygulayın

Bu alıştırmada GET /song/id uç noktasını uygulayacaksınız. `PyMongo`, Mongo Veritabanından belirli bir belge almak için `db.collections.find({})` yöntemini sağlar. Belgeyi bulmak için herhangi bir özellik üzerinden arama yapabilirsiniz. `id` özelliğini geçeceksiniz.

Göreviniz

Daha önce olduğu gibi, uç nokta için kodu `Back-End-Development-Songs/backend/routes.py` dosyasında yazacaksınız.

Open **routes.py** in IDE

Not: Dosya Gezgini'nde açmak için şu konuma gidin:
`Back-End-Development-Songs/backend/routes.py`

1. `/song/<id>` uç noktası için GET yöntemine yanıt veren bir Flask rotası oluşturun.
2. Uygulamayı tutmak için `get_song_by_id(id)` adında bir fonksiyon oluşturun.
3. `db.songs.find_one({"id": id})` yöntemini kullanarak bir şarkıyı id ile bulun.
4. Eğer id bulunamazsa, 404 NOT FOUND HTTP kodu ile `{"message": "id ile şarkı bulunamadı"}` mesajını döndürün.
5. Eğer şarkıyı veritabanında bulursanız, 200 HTTP OK durumu ile şarkıyı json formatında döndürün.
6. Flask sunucusunun çalıştığından emin olun. Uygulamayı test etmek için aşağıdaki curl komutunu çalıştırın:

```
curl --request GET -i -w '\n' --url http://localhost:5000/song/1
```

Aşağıdakine benzer bir çıktı görmelisiniz:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:23:59 GMT
Content-Type: application/json
Content-Length: 256
Connection: close
{"_id":{"$oid":"63e4587241323a01d2058e0c"},"id":1,"lyrics":"Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed, tincidunt eu, felis.","title":"dui
```

Bir sonraki uç noktaya geçelim.

Egzersiz 4: POST /song uç noktasını uygulama

`db.songs.insert_one` metodunu kullanarak veritabanına tek bir şarkı ekleyeceksiniz. Öncelikle şarkıyı istek gövdesinden çıkaracaksınız.

Göreviniz

Önceki gibi, uç nokta için kodu `Back-End-Development-Songs/backend/routes.py` dosyasında yazacaksınız.

Open **routes.py** in IDE

Not: Dosya Gezgini'nde açmak için bu konuma gidin:
`Back-End-Development-Songs/backend/routes.py`

1. `/song` uç noktası için POST metoduna yanıt veren bir Flask rotası oluşturun. Uygulama dekoratörünüzde `methods=["POST"]` kullanın.
2. Uygulamayı tutmak için `create_song()` adında bir fonksiyon oluşturun.
3. Öncelikle istek gövdesinden şarkı verilerini çıkaracak ve ardından `data` listesine ekleyeceksiniz.
4. Eğer belirtilen id ile bir şarkı zaten mevcutsa, kullanıcıya `{"Message": "id {song['id']} olan şarkı zaten mevcut"}` mesajı ile birlikte `302` HTTP kodu gönderin.
5. Uygulamayı test etmek için aşağıdaki curl komutunu çalıştırın:

```
curl --request POST \
  -i -w '\n' \
  --url http://localhost:5000/song \
  --header 'Content-Type: application/json' \
  --data '{
        "id": 323,
        "lyrics": "Integer tincidunt ante vel ipsum. Praesent blandit lacinia erat. Vestibulum sed magna at nunc commodo placerat.\n\nPraesent blandit. Nam nulla. Integer pede justo, lacinia eget, tincidunt eg
        "title": "in faucibus orci luctus et ultrices"
    }'
```

Benzer bir çıktı görmelisiniz:

```
HTTP/1.1 201 CREATED
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:26:43 GMT
Content-Type: application/json
Content-Length: 52
Connection: close
{"inserted id":{"$oid":"63e459e3b22f516761d30171"}}
```

Eğer aynı komutu tekrar gönderirseniz, sunucu curl istemcisine 302 döndürmelidir:

```
HTTP/1.1 302 FOUND
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:26:52 GMT
Content-Type: application/json
Content-Length: 47
Connection: close
{"Message":"song with id 323 already present"}
```

Egzersiz 5: PUT /song uç noktasını uygulayın

Bu uç noktada bir şarkıyı güncellemeniz isteniyor. İstemci, güncellenmiş şarkıyı isteğin gövdesinde gönderecek. Bu yöntemi uygulamak için `PyMongo`'da `db.songs.update_one()` yöntemini kullanacaksınız. `update_one` yönteminin değiştirilen şarkı olarak bir `$set` argümanı aldığını hatırlayın.

Göreviniz

PUT uç noktası, mevcut bir resim kaynağını güncellemek için kullanılacaktır. Daha önce olduğu gibi, uç nokta için kodu `Back-End-Development-Songs/backend/routes.py` dosyasında yazacaksınız.

Open **routes.py** in IDE

Not: Dosya Gezgini'nde açmak için bu konuma gidin:
`Back-End-Development-Songs/backend/routes.py`

1. `/song/<int:id>` uç noktası için POST yöntemine yanıt veren bir Flask rotası oluşturun. Uygulama dekoratörünüzde `methods=["PUT"]` kullanın.
2. Uygulamayı barındırmak için `update_song(id)` adında bir fonksiyon oluşturun.
3. Öncelikle, şarkı verilerini istek gövdesinden çıkarmanız gerekecek.

4. Ardından, `db.songs.find_one` yöntemini kullanarak veritabanında şarkıyı bulacaksınız. Şarkı mevcutsa, `db.songs.update_one` yöntemi ile gelen isteği kullanarak güncelleyeceksiniz.
5. Şarkı mevcut değilse, `{"message": "şarkı bulunamadı"}` mesajıyla birlikte `404` durumunu geri göndereceksiniz.
6. Uygulamayı test etmek için aşağıdaki curl komutunu çalıştırın:

```
curl --request PUT \
  -i -w '\n' \
  --url http://localhost:5000/song/1 \
  --header 'Content-Type: application/json' \
  --data '{
        "lyrics": "yay hey yay yay",
        "title": "yay song"
    }'
```

Bir çıktı aşağıdaki gibi olmalıdır:

```
HTTP/1.1 201 CREATED
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:45:21 GMT
Content-Type: application/json
Content-Length: 97
Connection: close
{"_id":{"$oid":"63e459e1b22f516761d3015d"},"id":1,"lyrics":"yay hey yay yay","title":"yay song"}
```

Eğer tam olarak aynı çağrıyı tekrar yaparsanız, aşağıdaki çıktıyı görmelisiniz:

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:45:40 GMT
Content-Type: application/json
Content-Length: 46
Connection: close
{"message":"song found, but nothing updated"}
```

Kanıt

Terminaldeki curl komutunun sonucunun ekran görüntüsünü alın ve adını `songs-ex5--put-song-passing.jpg` (veya `.png`) olarak kaydedin.

Egzersiz 6: DELETE /song uç noktasını uygulama

Bu egzersizde `DELETE song` uç noktasını uygulayacaksınız. Bunun için `PyMongo` tarafından sağlanan `db.songs.delete_one` yöntemini kullanacaksınız. Yöntemin herhangi bir belgeyi değiştirip değiştirmediğini `deleted_count` özelliğini kullanarak kontrol edebilirsiniz. Eğer `deleted_count` 0 ise, belge koleksiyonda bulunamadı demektir.

Görev 1 : Delete uç noktasını uygulama

DELETE uç noktası, mevcut bir şarkı kaynağını silmek için kullanılır. Önceki gibi, uç nokta için kodu `Back-End-Development-Songs/backend/routes.py` dosyasına yazacaksınız.

Open **routes.py** in IDE

Not: Dosya Gezgini'nde açmak için bu konuma gidin:
`Back-End-Development-Songs/backend/routes.py`

1. `/song/<int:id>` uç noktası için POST yöntemine yanıt veren bir Flask rotası oluşturun. Uygulama dekoratörünüzde `methods=["DELETE"]` kullanın.
2. Uygulamanın içeriğini tutmak için `delete_song(id)` adında bir fonksiyon oluşturun.
3. Öncelikle URL'den id'yi çıkaracaksınız.
4. Ardından, şarkıyı veritabanından silmek için `db.songs.delete_one` yöntemini kullanın.
5. Sonucun `deleted_count` niteliğini kontrol edin. Eğer `deleted_count` sıfırsa, `{"message": "şarkı bulunamadı"}` mesajıyla birlikte `404` durumunu geri göndereceksiniz.
6. Eğer `deleted_count` 1 ise, bu şarkının başarıyla silindiği anlamına gelir. HTTP_204_NO_CONTENT durumuyla birlikte boş bir gövde döndüreceksiniz.
7. Uygulamayı test etmek için aşağıdaki curl komutunu çalıştırın:

```
curl --request DELETE \
  -i -w '\n' \
  --url http://localhost:5000/song/14 \
  --header 'Content-Type: "application/json"'
```

Benzer bir çıktı görmelisiniz:

```
HTTP/1.1 204 NO CONTENT
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:56:02 GMT
Content-Type: text/html; charset=utf-8
Connection: close
```

Eğer aynı çağrıyı tekrar yaparsanız, son çağrı tarafından şarkının silinmiş olması nedeniyle 404 yanıtı almanız gerekir:

```
HTTP/1.1 404 NOT FOUND
Server: Werkzeug/2.2.2 Python/3.7.16
Date: Thu, 09 Feb 2023 02:56:15 GMT
Content-Type: application/json
Content-Length: 29
Connection: close
{"message":"song not found"}
```

Kanıt

Terminaldeki curl komutunun sonucunun ekran görüntüsünü alın ve adını `songs-ex6-delete-song-passing.jpg` (veya `.png`) olarak kaydedin.

Görev 2 : Dalı GitHub'a gönderin ve bir PR oluşturun

Mikroservis için kodu tamamladığınıza göre, `backend-rest` dalını GitHub fork'unuza geri gönderebilirsiniz. Bu projede tek çalıştığınız için PR'yi birleştirebilir ve dalı silebilirsiniz. Bir sonraki laboratuvara geçmeden önce tüm kod değişikliklerinizin ana dalına gönderildiğinden emin olun.

1. Değişikliklerinizi "şarkı servisi uygulandı" mesajıyla kaydetmek için `git commit -am` komutunu ve bu değişiklikleri deposuna göndermek için `git push` komutunu kullanın.

   Not: İlk kez gönderim yaptığınızda git kullanıcı adınızı ve e-posta adresinizi ayarlamanız istenecek:

   ```
   git config --local user.name "{GitHub adınızı buraya yazın}"
   git config --local user.email {GitHub e-posta adresinizi buraya yazın}
   ```

   ▶ İpucu için buraya tıklayın.
   ▶ İpucu için buraya tıklayın.

2. Değişikliklerinizi ana dal ile birleştirmek için GitHub'da bir pull request oluşturun ve ekibinizde başka kimse olmadığı için pull request'i kabul edin, birleştirin ve dalı silin.

Bu noktada, ana dalınız tamamlanmış kodunuzu içermelidir.

Çözümler

Bu sayfa, Listeleme, Oluşturma, Güncelleme ve Silme REST API'leri için çözümleri içermektedir.

Çözümler

Sağlık

▶ Çözümünüzü kontrol etmek için buraya tıklayın.

Sayı

▶ Çözümünüzü kontrol etmek için buraya tıklayın.

Liste

▶ Çözümünüzü kontrol etmek için buraya tıklayın.

Oku

▶ Çözümünü kontrol etmek için buraya tıkla.

Oluştur

▶ Çözümünüzü kontrol etmek için buraya tıklayın.

Güncelleme

▶ Çözümünüzü kontrol etmek için buraya tıklayın.

Sil

▶ Çözümünüzü kontrol etmek için buraya tıklayın.

Sonuç

Tebrikler! Şarkıları almak için ikinci mikroservisi uygulamayı tamamladınız. Bu mikroservis, projenin son laboratuvarında ana site tarafından kullanılacaktır.

Sonraki Adımlar

Bu noktada kursa devam edebilirsiniz. Bir sonraki modülde ana Django uygulamasını oluşturmanız istenecek.

Author(s)

CF

© IBM Corporation. Tüm hakları saklıdır.