

Knightsbridge Technologies

Api Document For Practical Test (Software Engineer)

- Content

1. How to set up the project.
2. Creating personal access client token.
3. Request to end points

1. How to setup the project

Pull the project from the github and run the seeder mentioned below..that command creates 100 sellers and products.

- ★ php artisan migrate
- ★ php artisan db:seed

Note:- the provided database has 100 records.

2. Client credentials to retrieve access token..

Client ID: 3

Client secret: V3RSYa44Ft4c4640xSLwj7jfkK0mDR5WsKmjOIR1

http://127.0.0.1:8000/oauth/token

Tests

JSON (application/json) ▾

```
1 {
2   "grant_type" : "client_credentials",
3   "client_id" : "3",
4   "client_secret" : "V3RSYa44Ft4c4640xSLwj7jfKkOMDr5WsKmJ0IR1"
5 }
```

Test Results



```
1 {  
2   "token_type": "Bearer",  
3   "expires_in": 31536000,  
4   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1Ni9yZjdhZWQioiIzIiwianRpIjoiaWwuyZWE5NmYzOTVhNDYzNDY4  
    kwHwiXzhWjIwNjE3ODk0OTA3LCJ2dWIiOiIiLCJzY29wZXMiOiJ0eHQiLmRlZCJ2Ym1CW92Fwpzlh2X0w_OQ1Wzu1hO5Lf-  
    uHybPRNKXlnfPk6NZqCYsYN3Fno7VXTW022Qiv5Je1Ih0EWzlpYhtPpdT2Xa13PWZEB0-JrmRS81sl1m4Hs0b7tylzipt34v-  
    -io5r9Bt21I8wpwKG1dSRjh0sEJHzHXjpJ46w359yWsCUUEB6cNV3recXTRQ28GAjgFAxmr0Zrr9Y81EUOI5TnEyRzf0nhgb4-  
    -IkOZISBZNra1YTytgmwzoA-Ck7m9F721szbkFYSA51WytIPAUSqd1jCrogrLQ05Mc1suQ1JZ_Ihj0hiXiE_kDKi6mywikF2yq-  
    -B_1tMmWms_jYUoVsxyeepV8M5m4FfoTK42QP5Y9Q1OFqHeYNLPgyWDr1MKIdSXHWbcuXc3EJoufkmtY4fIM1BEMKmZnsID5  
5 }
```

- After that we can request to following api from this access token
 1. Get Details of a product.

POST ▾

http://127.0.0.1:8000/api/product

Authorization

Headers (3)

Body ●

Pre-request Script

Tests

	Key	Value
<input checked="" type="checkbox"/>	Accept	application/json
<input checked="" type="checkbox"/>	Content-Type	application/json
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOi...
	New key	Value

Body

Cookies

Headers (8)

Test Results

Pretty

Raw

Preview

JSON ▾

```
1 {
2   "data": {
3     "id": 3,
4     "name": "Joannie Altenwerth MD",
5     "price": 7923.67,
6     "stock": 83,
7     "image": "http://localhost/storage/image/yj6bEkpjTx.jpg"
8   }
9 }
```

2. Get product list by seller id.

The screenshot displays a REST client interface with a POST request to `http://127.0.0.1:8000/api/products-by-seller`. The request body is a JSON object with an `"id": 4` field. The response is a JSON array of product objects, each containing `"id"`, `"name"`, `"price"`, `"stock"`, and `"image"` fields. The response is shown in the 'Body' tab, formatted as JSON.

Request:

```
POST http://127.0.0.1:8000/api/products-by-seller
{
  "id": 4
}
```

Response:

```
{
  "data": [
    {
      "id": 4,
      "name": "Marquise Wisozk PhD",
      "price": 7778.92,
      "stock": 15,
      "image": "http://localhost/storage/image/vMSBcxZony.jpg"
    },
    {
      "id": 5,
      "name": "Jovanny Heidenreich",
      "price": 6516.61,
      "stock": 23,
      "image": "http://localhost/storage/image/PQpwN8ChEz.jpg"
    },
    {
      "id": 6,
      "name": "Dr. Kendall O'Reilly MD",
      "price": 973.91,
      "stock": 47,
      "image": "http://localhost/storage/image/zP490cnLSt.jpg"
    }
  ]
}
```

3. Get seller detail for a product

★ In there I used a Resource collection (SellerResource) to filter the seller id name and email.

The screenshot displays a REST client interface. The top section shows a POST request to the URL `http://127.0.0.1:8000/api/seller-by-product`. The 'Body' tab is selected, showing a JSON payload:

```
{
  "id": 3
}
```

. The bottom section shows the response body, which is a JSON object:

```
{
  "data": [
    {
      "id": 3,
      "name": "Eudora Corkery",
      "email": "stevie18@example.org"
    }
  ]
}
```

. The response is formatted in 'Pretty' mode and is of type 'JSON'.