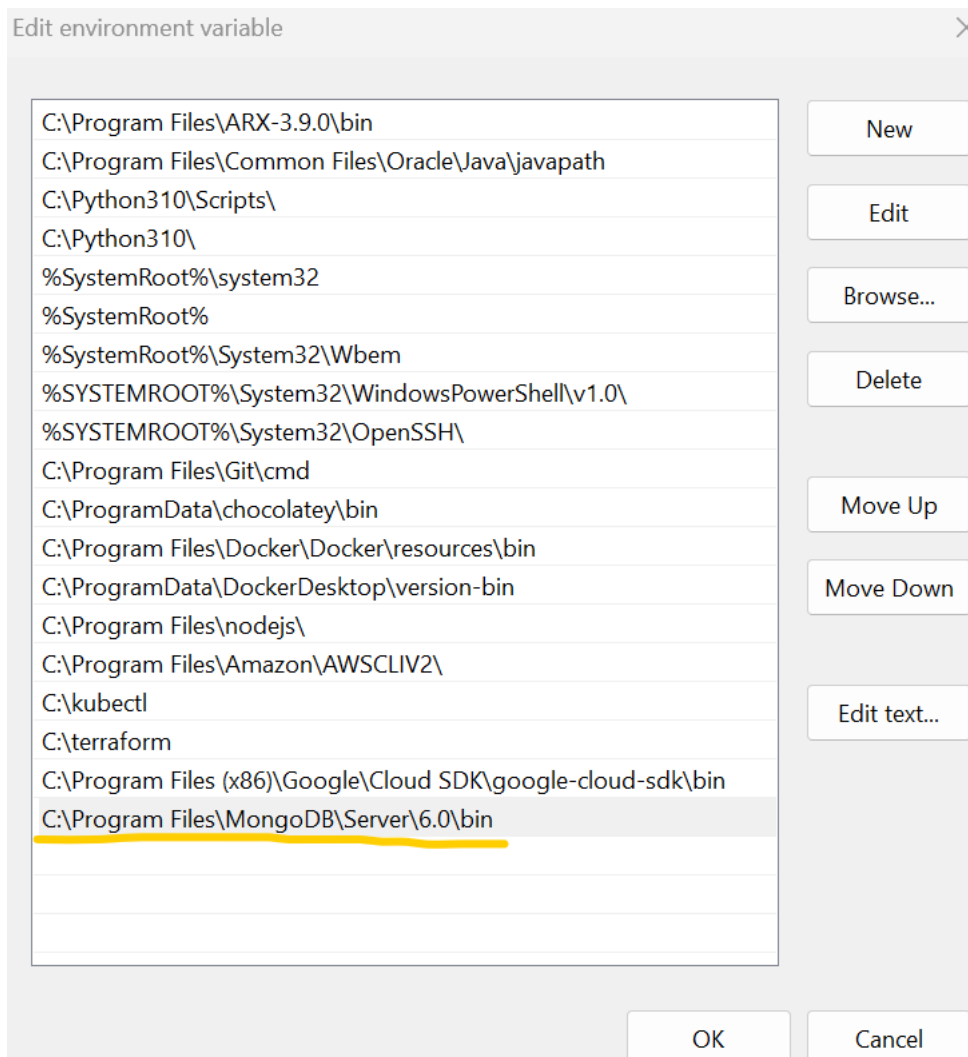


9.1P: Adding a database to your application

1. Installed MongoDB

- Downloaded version 6.0 of Mongo db and installed the complete version.
- Set the environment variable path to add mongo db's bin folder location



2. Create a MongoDB user with appropriate permissions for your application.

- Created new connection in MongoDB Compass by using New Connection > Authentication > username as admin and password as password

MongoDB Compass

Connect Edit View Help

Compass

New connection +

Saved connections

Recents

New Connection

Connect to a MongoDB deployment

FAVORITE

URI ⓘ Edit Connection String ☒

mongodb://admin:password@localhost:32000/?authMechanism=DEFAULT

▼ Advanced Connection Options

General **Authentication** TLS/SSL Proxy/SSH In-Use Encryption Advanced

Authentication Method

None **Username/Password** X.509 Kerberos LDAP AWS IAM

Username

admin

Password

.....

Authentication Database ⓘ

Optional

Authentication Mechanism

Default SCRAM-SHA-1 SCRAM-SHA-256

Save Save & Connect Connect

3. Configurations and commands –

- Configured **persistent** storage for the MongoDB database by creating a Persistent Volume and Persistent Volume Claim.
 - Ran the command **kubectl apply -f .** to configure all the yamls at once
 - Configured Kubernetes Secret file for the MongoDB user credentials and added them to the deployment manifest. Secret is an object that contains sensitive information such as password etc.
- There is a small yaml to configure this –

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
immutable: false
type: Opaque
data:
  password: cGFzc3dvcmQ
```

- Added MongoDB **secret** to deployment manifest.

```
env:
  - name: MONGO_INITDB_ROOT_USERNAME
    value: "admin"
  - name: MONGO_INITDB_ROOT_PASSWORD
    valueFrom:
      secretKeyRef:
        name: mongodb-secret
        key: password
```

- **MongoDb** is configured in server.js by -

Imported mongodb library as -->

const MongoClient = require('mongodb').MongoClient;

and then used the connection url shown as 'uri' below. Database name is 'crud' -

```
const uri = 'mongodb://admin:password@localhost:32000/?authMechanism=DEFAULT';
MongoClient.connect(uri, (err, client) => {
  if (err) return console.log(err);
  db = client.db('crud');

  app.use('/api/v1/users', usersCtrl);

  app.listen(3000, function() {
    console.log('server running on port 3000', '');
  });
});
```

- Below are the screenshots for all the created configurations explained above and executed with command **kubectl apply -f**.

These screenshot demonstrate successful creation of pv, pvc, etc that happened by running the kubectl get commands—

Pv and pvc –

```
C:\Users\Karthik>kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mongo-pvc     Bound     pvc-b201d266-7066-4f07-a7ac-e326761e76ef  500M       RWX             hostpath       40m

C:\Users\Karthik>kubectl get pv
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM          STORAGECLASS   REASON   AGE
mongo-pv     500M       RWX             Retain            Available      default/mongo-pvc  hostpath      40m
pvc-b201d266-7066-4f07-a7ac-e326761e76ef  500M       RWX             Delete            Bound                                     40m
```

Svc-

```
C:\Users\Karthik>kubectl get svc
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes    ClusterIP   10.96.0.1    <none>        443/TCP          8h
mongo-svc     NodePort    10.105.103.50 <none>        27017:32000/TCP  41m
```

Pods-

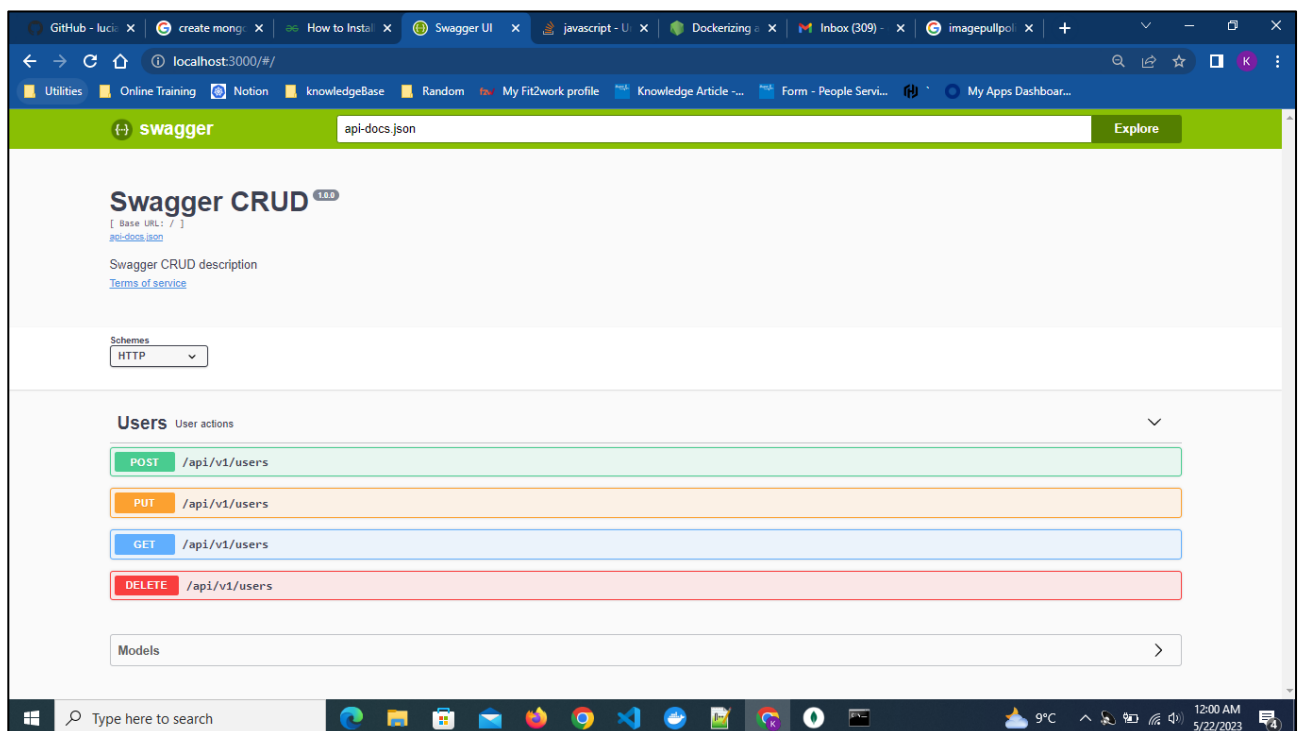
```
C:\Users\Karthik>kubect1 get all -A
```

| NAMESPACE | NAME | READY | STATUS | RESTARTS | AGE |
|-------------|--|-------|---------|----------------|-----|
| default | pod/mongo-5fc8bb68b-ck5jx | 1/1 | Running | 0 | 40m |
| default | pod/nodejs-dbd4998f6-qgzc5 | 1/1 | Running | 2 (17s ago) | 37s |
| kube-system | pod/coredns-565d847f94-69jmg | 1/1 | Running | 0 | 8h |
| kube-system | pod/coredns-565d847f94-78dpb | 1/1 | Running | 0 | 8h |
| kube-system | pod/etcd-docker-desktop | 1/1 | Running | 0 | 8h |
| kube-system | pod/kube-apiserver-docker-desktop | 1/1 | Running | 0 | 8h |
| kube-system | pod/kube-controller-manager-docker-desktop | 1/1 | Running | 0 | 8h |
| kube-system | pod/kube-proxy-841lr | 1/1 | Running | 0 | 8h |
| kube-system | pod/kube-scheduler-docker-desktop | 1/1 | Running | 2 (6h43m ago) | 8h |
| kube-system | pod/storage-provisioner | 1/1 | Running | 2 (6h43m ago) | 8h |
| kube-system | pod/vpnkit-controller | 1/1 | Running | 24 (7m46s ago) | 8h |

Both mongodb pod and nodejs pods are running

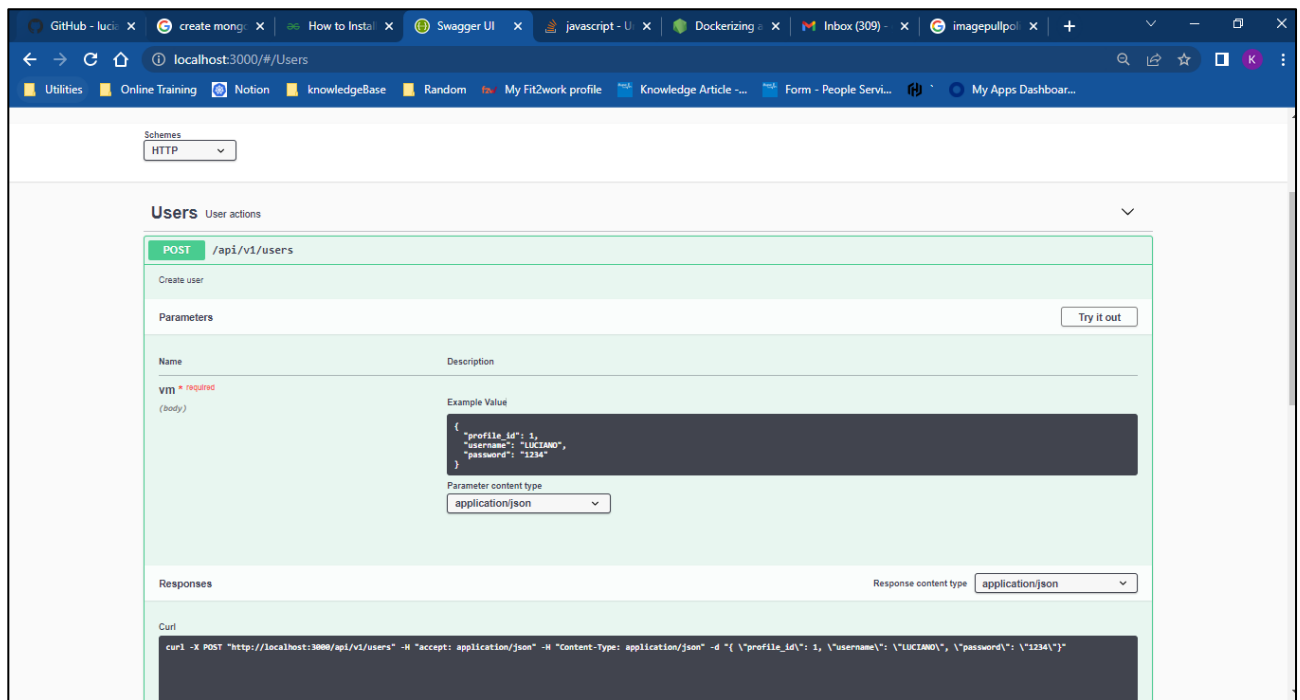
5. Tested the deployment for CRUD (Create, Read, Update, Delete) operations

Launched localhost:3000 as –

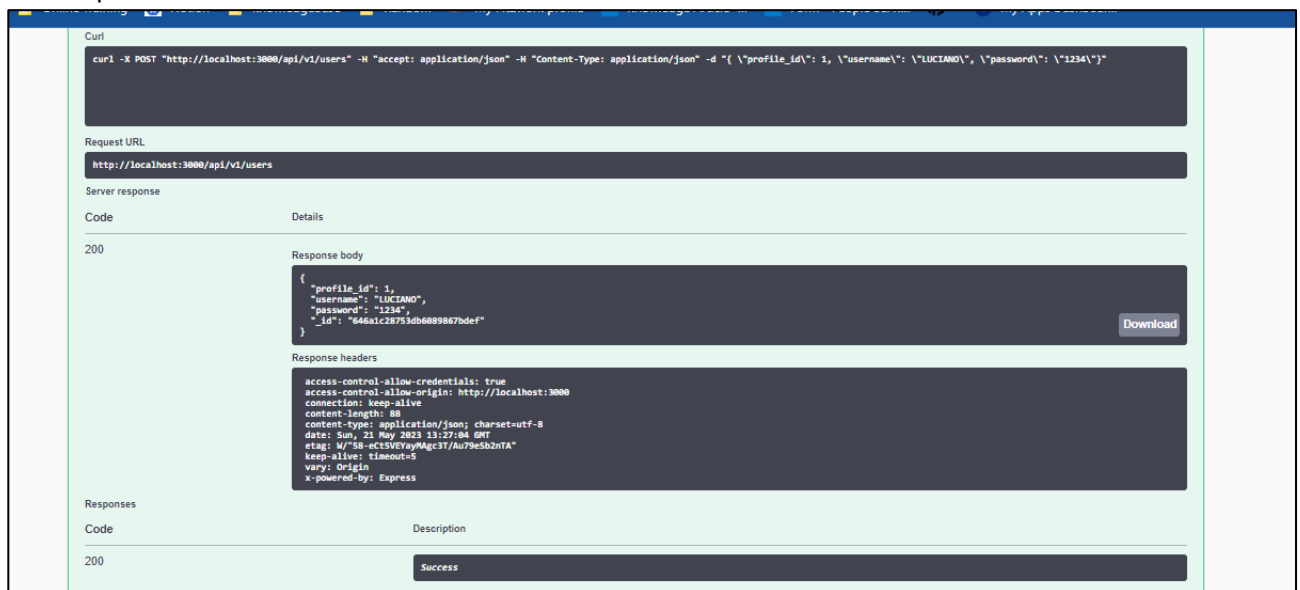


Step 1. Testing POST - /Create endpoint. The input and output screenshots are -

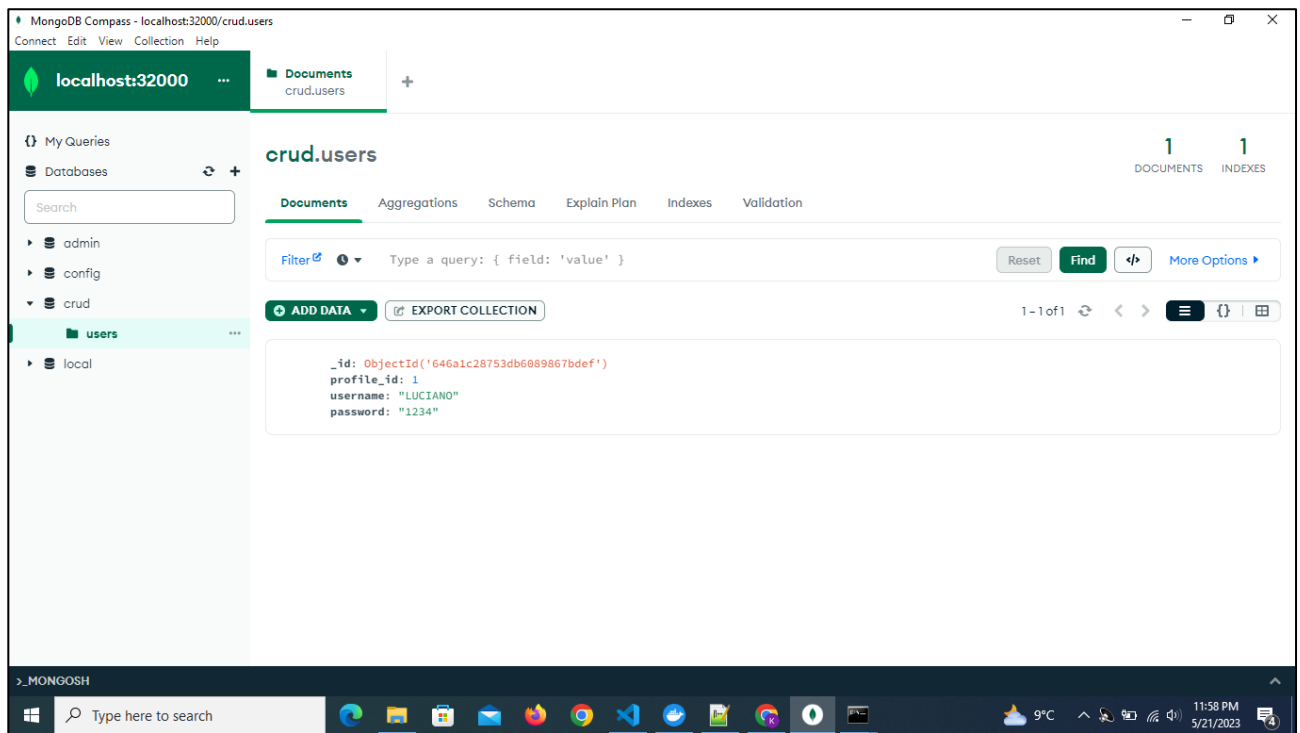
Inputted the body in POST as follows-



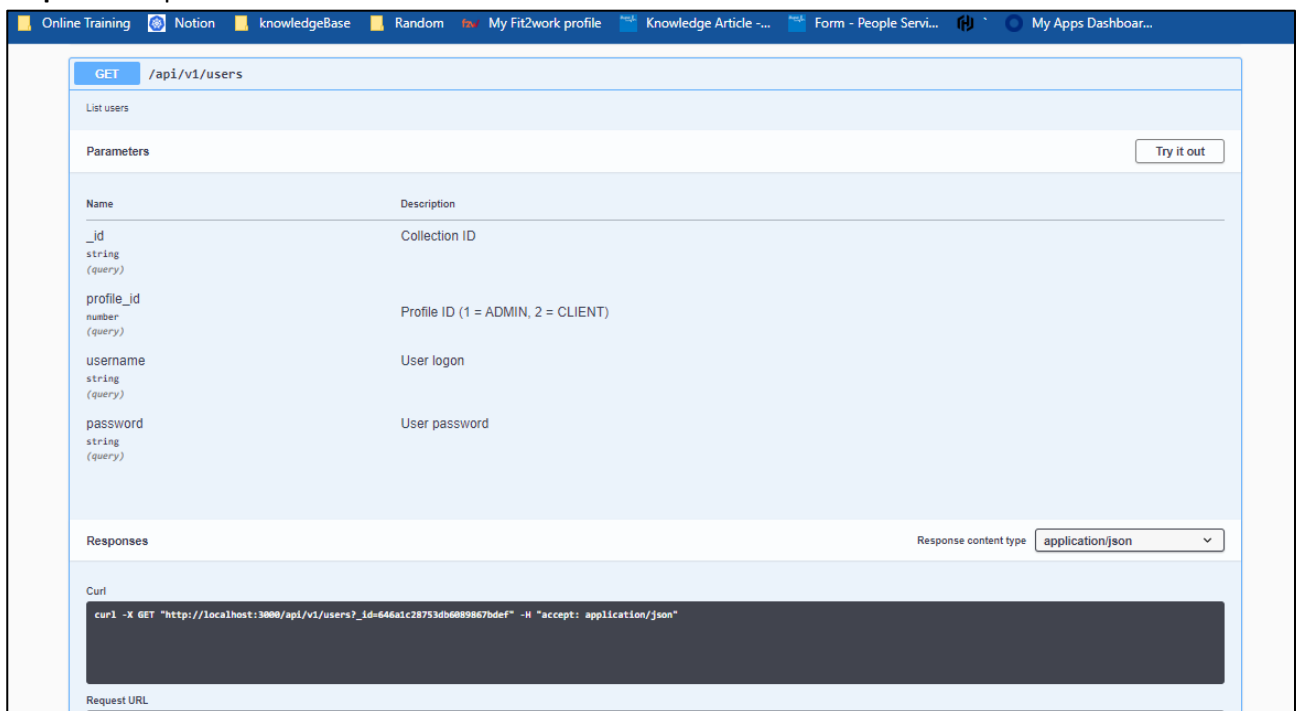
The output 200 was received as –



The Mongo db compass resultantly showed –



Step 2. GET operation –



| | |
|---|---|
| Request URL | |
| http://localhost:3000/api/v1/users?id=646a1c28753db689867bdef | |
| Server response | |
| Code | Details |
| 200 | <div>Response body</div> <pre>{ "id": "646a1c28753db689867bdef", "profile_id": 1, "username": "LUKIANO", "password": "1234" }</pre> <div>Download</div> <div>Response headers</div> <pre>access-control-allow-credentials: true connection: keep-alive content-length: 98 content-type: application/json; charset=utf-8 date: Sun, 21 May 2023 13:27:43 GMT etag: W/"5a-3rQ9wK2CQ9nKdPQn/WIEmj4" keep-alive: timeout=5 vary: Origin x-powered-by: Express</pre> |
| Responses | |
| Code | Description |

Step 3. PUT operation – Changed id from 1234 to 12345 for the object id created in the output of POST operation.

Input –

es

Online Training

Notion

knowledgeBase

Random

My Fit2work profile

Knowledge Article ...

Form - People Servi...

My Apps Dashboar...

PUT /api/v1/users

Update user

Parameters

Try it out

Name

Description

vm * required

(body)

Example Value

```
{
  "id": "5c8728bdc78e3dec08de6",
  "profile_id": 1,
  "username": "LUKIANO",
  "password": "12345"
}
```

Parameter content type

application/json

Responses

Response content type

application/json

Curl

```
curl -X PUT "http://localhost:3000/api/v1/users" -H "accept: application/json" -H "Content-Type: application/json" -d '{ "id": "5c8728bdc78e3dec08de6", "profile_id": 1, "username": "LUKIANO", "password": "12345" }'
```

Request URL

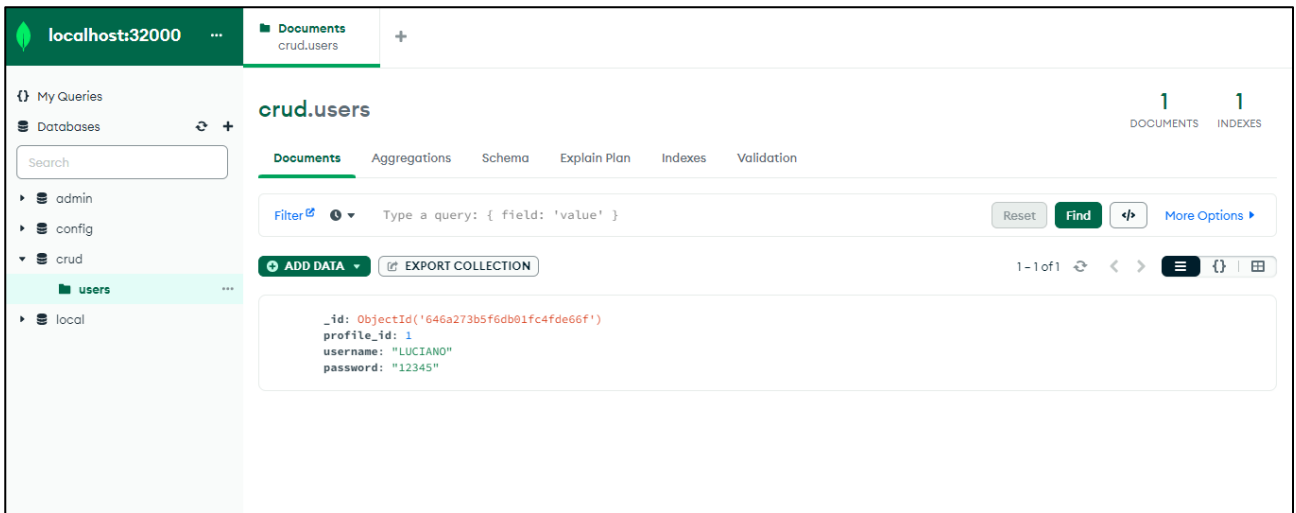
http://localhost:3000/api/v1/users

Server response

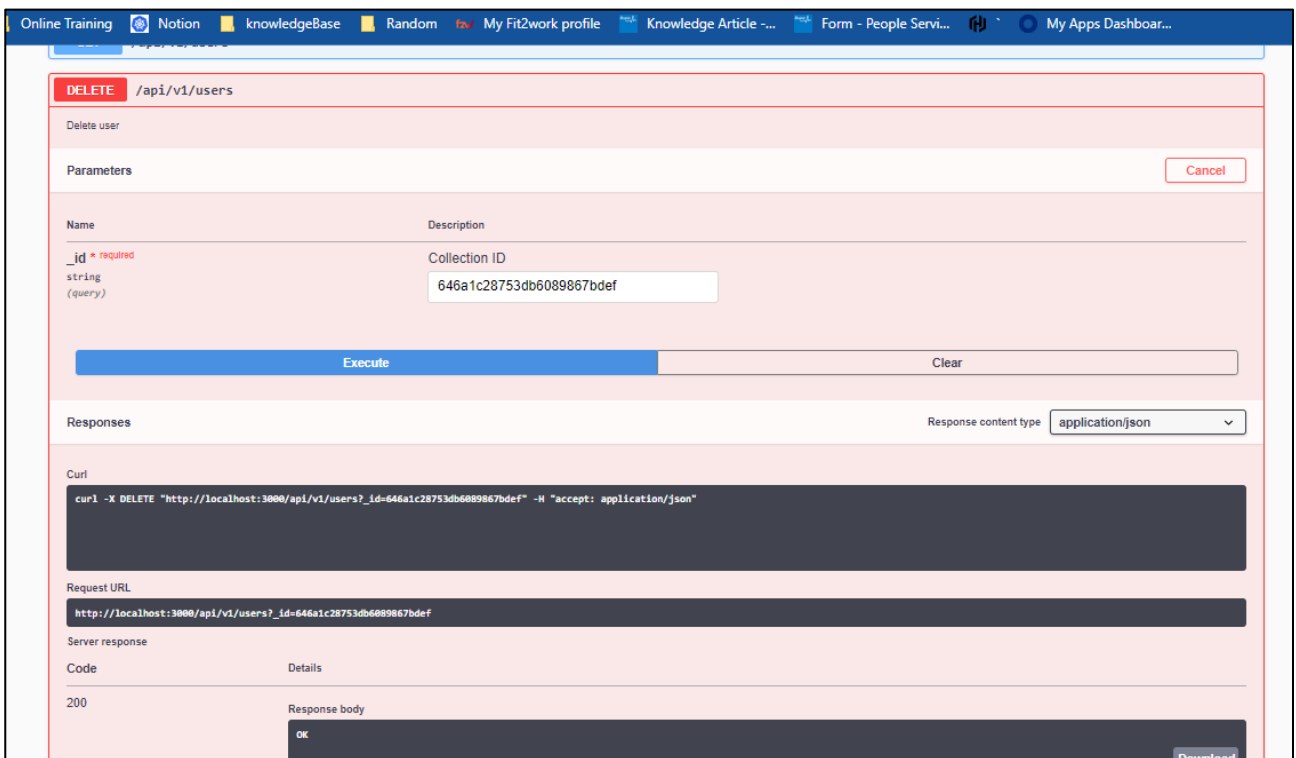
Output –

| | |
|------------------------------------|--|
| Request URL | |
| http://localhost:3000/api/v1/users | |
| Server response | |
| Code | Details |
| 200 | <div>Response body</div> <pre>OK</pre> <div>Download</div> <div>Response headers</div> <pre>access-control-allow-credentials: true access-control-allow-origin: http://localhost:3000 connection: keep-alive content-length: 2 content-type: text/plain; charset=utf-8 date: Sun, 21 May 2023 13:28:07 GMT etag: W/"2-m0D9Q1ThAg0uKt3Lzzdv3SLc" keep-alive: timeout=5 vary: Origin x-powered-by: Express</pre> |
| Responses | |

Corresponding change in db –



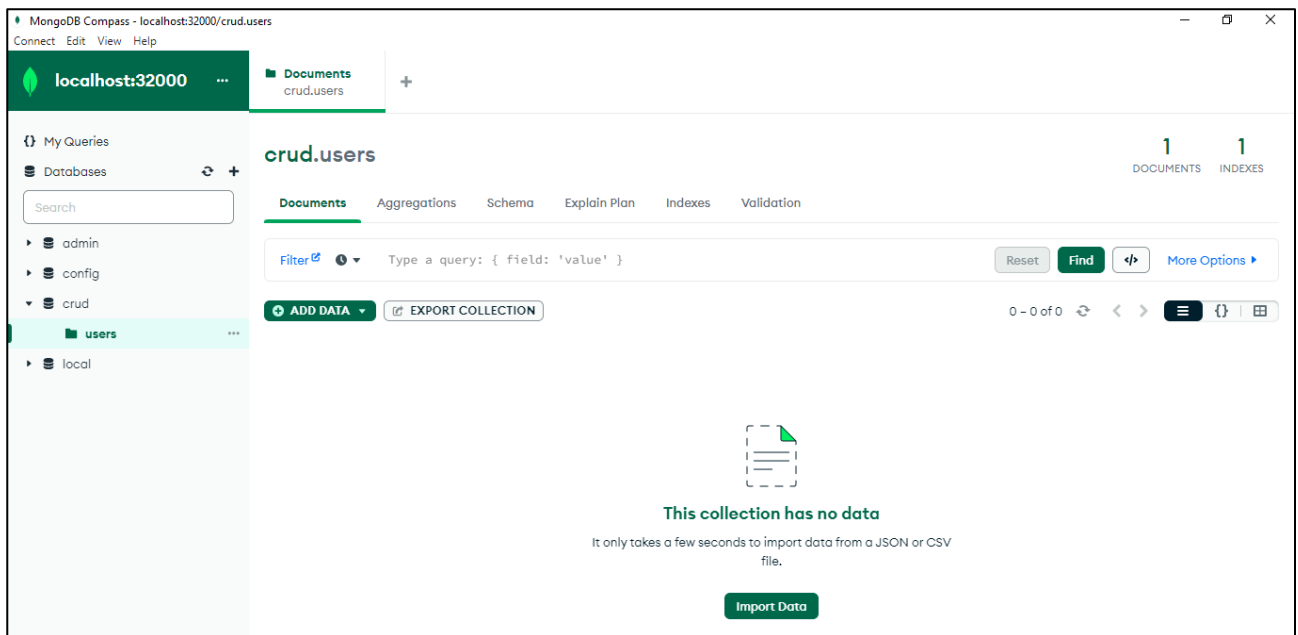
Step 4. DELETE operation using the id



Deletion success code 200 -



Effect on mongodb compass –



Note:

In the code submitted, all the Kubernetes code and configurations are done as per my understanding from the workshop content.

CRUD operations are learnt from research, mainly from the source –

[<https://github.com/lucianopereira86/NodeJS-MongoDB-Kubernetes/tree/master>]