Implement API endpoints using Express-Mongo

Skills Network

Estimated time needed: 90 minutes

Congratulations! You are one step closer to finishing the capstone project. You created a web application and added user authentication using the Django framework and REACT in the previous modules. Take a pause and pat yourself on the back. However, it is not done yet! In this module, you are asked to create the backend services for your application.

The Django application will talk to MongoDB using backend services. You are asked to create these services in JavaScript. You will write these services in an Express app and deploy it on Code Engine.

Working with Mongoose to provide API endpoints

1. Open a new terminal and clone your project from GitHub if the environment has been reset.

2. Change to the directory with the data files.

```
cd /home/project/xrwvm-fullstack_developer_capstone/server/database
```

Copied! Executed!

For your online course lab project, you have been provided two schema files for Reviews and Dealerships entities, along with JSON files containing dealership and review data to be loaded into MongoDB and served through endpoints.

- server/database/data/dealerships.json
- server/database/data/reviews.json

3. The Node app will use `mongoose` to interact with the MongoDB. The schemas for Reviews and Dealerships are defined in `review.js` and `dealership.js`, respectively.

4. View the content in `server/database/app.js`. It will provide the following endpoints:

- fetchReviews (for fetching all reviews)
- fetchReviews/dealer/:id (for fetching reviews of a particular dealer)
- fetchDealers (for fetching all dealerships)
- fetchDealers/:state (for fetching all dealerships in a particular state)
- fetchDealer/:id (for dealer by id)
- insert_review (for inserting reviews)

Some of the endpoints have been implemented for you. Use the ideas and prior learning to implement the endpoints that are not implemented.

5. Run the following command to build the Docker app.

    Remember to do this, every time you make changes to app.js.

```
docker build . -t nodeapp
```

Copied! Executed!

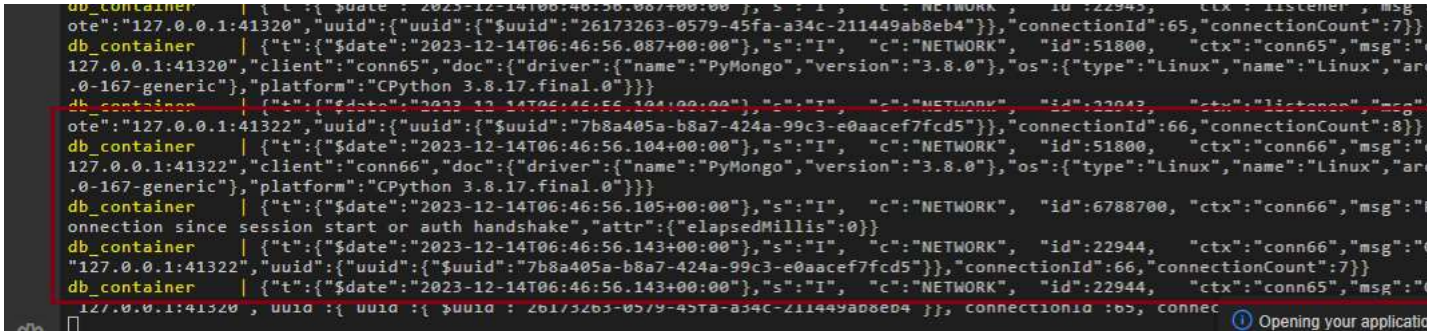
The first time you build, it takes up to two minutes to successfully build.

6. The `docker-compose.yml` has been created to run two containers, one for Mongo and the other for the Node app. Run the following command to run the server:

```
docker-compose up
```

Copied! Executed!

- When you see the output on the terminal as shown in the following image, it would mean that the server has successfully started, and you can connect to it.



```
db_container    | {"t":{"$date":"2023-12-14T06:46:56.087+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg
ote":"127.0.0.1:41320","uuid":{"uuid":{"$uuid":"26173263-0579-45fa-a34c-211449ab8eb4"}},"connectionId":65,"connectionCount":7}}
db_container    | {"t":{"$date":"2023-12-14T06:46:56.087+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn65","msg":"
127.0.0.1:41320","client":"conn65","doc":{"driver":{"name":"PyMongo","version":"3.8.0"},"os":{"type":"Linux","name":"Linux","ar
.0-167-generic"},"platform":"CPython 3.8.17.final.0"}}}
db_container    | {"t":{"$date":"2023-12-14T06:46:56.104+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg"
ote":"127.0.0.1:41322","uuid":{"uuid":{"$uuid":"7b8a405a-b8a7-424a-99c3-e0aacef7fcd5"}},"connectionId":66,"connectionCount":8}}
db_container    | {"t":{"$date":"2023-12-14T06:46:56.104+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn66","msg":"
127.0.0.1:41322","client":"conn66","doc":{"driver":{"name":"PyMongo","version":"3.8.0"},"os":{"type":"Linux","name":"Linux","ar
.0-167-generic"},"platform":"CPython 3.8.17.final.0"}}}
db_container    | {"t":{"$date":"2023-12-14T06:46:56.105+00:00"},"s":"I",  "c":"NETWORK",  "id":6788700, "ctx":"conn66","msg":"
onnection since session start or auth handshake","attr":{"elapsedMillis":0}}
db_container    | {"t":{"$date":"2023-12-14T06:46:56.143+00:00"},"s":"I",  "c":"NETWORK",  "id":22944,   "ctx":"conn66","msg":"
"127.0.0.1:41322","uuid":{"uuid":{"$uuid":"7b8a405a-b8a7-424a-99c3-e0aacef7fcd5"}},"connectionId":66,"connectionCount":7}}
db_container    | {"t":{"$date":"2023-12-14T06:46:56.143+00:00"},"s":"I",  "c":"NETWORK",  "id":22944,   "ctx":"conn65","msg":"
127.0.0.1:41320","uuid":{"uuid":{"$uuid":"26173263-0579-45fa-a34c-211449ab8eb4"}},"connectionId":65,"connec
```

7. Click `Fetch Reviews` below to test if the API endpoint returns all the reviews as intended.

Fetch Reviews

Create API Endpoint URLs

1. Implement the following endpoints that are yet to be implemented in `server/database/app.js`.
   - fetchDealers
   - fetchDealers/:state
   - fetchDealer/:id
2. Stop your Docker application that you started in the previous task.
3. Execute the `docker build` and `docker compose` commands again.
4. Click `Fetch Reviews` below to test all endpoints by replacing the route in the address bar.

Fetch Reviews

5. Test the following endpoints, take a screenshot of the same, and save as specified.

- /fetchReviews/dealer/29 - Save the screenshot as `dealer_review.png` or `dealer_review.jpg`.
- /fetchDealers - Save the screenshot as `dealerships.png` or `dealerships.jpg`.
- /fetchDealer/3 - Save the screenshot as `dealer_details.png` or `dealer_details.jpg`.
- /fetchDealers/Kansas - Save the screenshot as `kansasDealers.png` or `kansasDealers.jpg`.

6. Push the updated app.js code to your GitHub.

Submission

Ensure you have the following screenshots to submit for peer review:

1. `dealer_review` in png or jpg format.
2. `dealerships` in png or jpg format.
3. `dealer_details` in png or jpg format.
4. `kansasDealers` in png or jpg format.

Summary

In this lab, you created a containerized Node.js application that uses MongoDB as a backend to serve API endpoints.

Author(s)

Lavanya T S

Other Contributor(s)

Upkar Lidder
Yan Luo