TO00BS65-3001

Project 3 (API)

This application was written for the course `Full Stack Development TO00BS65-3001` at Laurea.

Table Of Contents

Introduction

This app implements a very simple `NodeJS API`. running on `ExpressJS`, and it's written in `TypeScript`. The `production` build of the application is deployed to `Heroku` and can be accessed via the following link laurea-api.herokuapp.com/](https://laurea-api.herokuapp.com/)

Local development

To run this application locally in development mode, you need to make sure you satisfy the following prerequisities:

- `node > 16`

```
➜  ~ node -v
Welcome to Node.js v16.2.0
```

- `NPM > 7`

```
7.1.1
```

- `Docker > 20`

```
➜  ~ docker -v
Docker version 20.10.6, build 370c289
```

1. clone this repo
2. `docker compose build`
3. `docker compose up -f docker-compose.dev-yml`

After the containers are up and running, you should see something like this on your `console`:

```
➜  TO00BS65-3001_3 git:(master) docker compose up
[+] Running 4/4
 ⠿ Network to00bs65-3001_3_app-network  Created
5.4s
 ⠿ Network to00bs65-3001_3_default      Created
4.2s
 ⠿ Container mongo-dev                 Created
0.2s
 ⠿ Container api-server                Created
11.7s
Attaching to api-server, mongo-dev
mongo-dev   | {"t":{"$date":"2022-04-
api-server  | [2022-04-25T20:14:29.797Z] [INFO] Connecting debug to
MongoDB at mongodb://host.docker.internal:27017/api
api-server  | [2022-04-25T20:14:30.053Z] [INFO] Connected to MongoDB at
mongodb://host.docker.internal:27017/api
api-server  | [2022-04-25T20:14:30.088Z] [INFO] 🌍 Express server started
at http://localhost:9009
api-server  | [2022-04-25T20:14:30.093Z] [INFO] process
api-server  | [2022-04-25T20:14:30.094Z] [INFO] ⚙  Swagger UI hosted at
http://localhost:9009/dev/api-docs
```

Now, the `MongoDB` port's should be exposed at the `27017` port, while the `API` itself on the `9009` port. For further information of the local docker envorinment, please refer to the compose implementation at `docker-compose.dev.yml`, and the `local.Dockerfile`.

General overview

Project structure

The project builds up by the following structure

```
.
├── __tests
│   └── errors
├── logs
├── scripts
└── src
    ├── controllers
```

```
│   └── book
├── errors
├── lib
├── middleware
├── models
└── public
```

GIT

The project's GIT repository can be found here.

There are GitHub actions in place, can be found here. More about these in the deployment section of this document.

There's a very basic branching model introduced in this project, which basically contains two branches on the HEAD:

- master
- release

The master is reserved, for active development, while new releases are triggered via Pull Requests on the release branch.

Implementation

Stack

This API is written using the TypeScript superset on top of Javascript using ES6 features, and ExpressJS.

The local development uses multiple Docker containers:

- backend, which builds the backend's source code
- mongodb, which contains a MongoDB instance

These two are composed with docker compose, and connected via a network bridge.

Routes

The API has the following routes:

route path	method	controler	description
/api/book/add	POST	BookController.add	insert's a new Book document into the database
api/book/all	GET	BookController.all	returns every Book document from the database
/api/book/search	GET	BookController.search	returns [Book] document(s) from the database

route path	method	controler	description
`/api/book/id/:bookId`	`GET`	`BookController.get`	returns the `Book` document from the database, by the given `bookId`
`/api/book/id/:bookId`	`DELETE`	`BookController.remove`	deletes the `Book` document from the database, by the given `bookId`

The implementation of the `routes` uses the `middleware` approach given by `expressJS`, where the `routes` controllers are passed in an `Array`:

```
const route: Route = api.use(path: String,
ReadOnlyArray<RequestHandlerParams>)
```

All the `routes` for this `API` is implemented in `src/routes.ts`.

Database

The application uses `MongoDB` to store application data, using the well-known `mongoose` interface.

There is a single `mongoose Schema` defined, which implements a `Book`:

```
export interface IBook extends Document {
  name: string;
  author: string;
  createdAt: Date;
  updatedAt: Date;
}
```

The `schema` is defined at `src/models/Book.ts`.

`Docker` (prod)

The `production` build also deployed using `Docker`. The setup however is slightly different than the `local` environment. The `API` is distributed via *multiple* `Heroku` apps, becasue it seems that `Heroku` can't handle multiple images in a single application. Therefore the `MongoDB` runs in one `dyno`, and the backend itslef runs on a separete one. These are composed together via a different `network` bridge in the backen'd docker image.

Deployment

The application's deployment is automated using `GitHub` actions. The release process will be auto-triggered, once a `PR` is getting merged from `master` into `release` branch. The pricess will rebuild, and re-publush the docker images to `Heroku`, and re-starts all the `dynos`.

Testing

The API can be easily tested using the foundamental way of API documentation, which is the SwaggerUI.

Every route has it's route defined in Swagger which not just document's but provides a playground to make every endpoint easy callable from their UI.

The SwaggerUI can be accessed via the (/dev/api-docs)[https://laurea-api.herokuapp.com/dev/api-docs/]. The manifest of the apidoc is defined in the openapi.json file.

Every request contains a schema which defines an example body (where needed), and can be executed as is. The response is also shown in the UI.

POST: /api/book/add


```
{
  "message": "Saved",
  "book": {
    "name": "The Hitchhiker's Guide to the Galaxy",
    "author": "Douglas Adams",
    "_id": "62670eec1ab01ad875ff38af",
    "createdAt": "2022-04-25T21:13:16.723Z",
    "updatedAt": "2022-04-25T21:13:16.723Z",
    "__v": 0
```


GET | /api/book/all


```
{
  "books": [
    {
      "_id": "62670eec1ab01ad875ff38af",
      "name": "The Hitchhiker's Guide to the Galaxy",
      "author": "Douglas Adams",
      "createdAt": "2022-04-25T21:13:16.723Z",
      "updatedAt": "2022-04-25T21:13:16.723Z",
      "__v": 0
    }
  ]
```