Books Api

Api REST for resources books write in NodeJS + Express + MongoDB

Description

Goal of project is to expose API of Books resource. This project is part of *MEAN* (MongoDB Express Angular NodeJS) stack project used as example with *Kubernetes* and *Docker*.

Project configuration

To configure the project you can use `npm install`

Step used to create project

Init project and dependencies:

```
npm init
npm install express --save
npm install body-parser --save
```

Configuration of requirement

In file `index.js` i have defined this `constant`:

```
const express = require('express');
const bodyParser = require("body-parser");
const cors = require('cors');
const path = require('path');
```

express: is used to start application and manage rooting ***bodyParser***: is used to manage request with `body` ***cors***: is used to manage headers for cors filter ***path***: is used to use file as resources

App definition

Now we can create `app` to manage request/response of API.

```
const app = express();
```

Add middleware

In express we can use `middleware` as interceptor of request. This middleware can add behaviour befor each request and after each response.

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
app.use(cors({origin:'*'}));
```

Route definition

Now we can use `app` to define routing for each endpoint we want.

Example:

```
app.get('/', (req, res) => {
    console.log("HOME PAGE");
    res.status(200).sendFile(path.join(__dirname, '/www/index.html'));
});
```

In this example we have configure `app` to listen at root path `/` and response with html file `/www./index.html`

Run server

We can use app to listen request:

```
const host = process.env.HOST || 'localhost';
const protocol = process.env.PROTOCOL || 'http';
const port = process.env.PORT || 3000;
...
```

```
app.listen(port, () => {
    console.log(`Services listening at ${protocol}://${host}:${port}`)
    console.log("--> /home     : Home page");
    console.log("--> /health   : health check servizio");
})
```

Now if we want to start serve we can just use:

```
node index.js
```

Integrate docker

To integrate docker we can use a `dockerfile` to define command to create image and then run the `build`.

Example of dockerfile:

```
FROM node:17-alpine

COPY package*.json ./
COPY *.html ./
COPY *.js ./

RUN npm cache clear --force && npm install

ENTRYPOINT ["node", "index.js"]
```

Create image

With command `docker build . -t books-api` we tell docker to create an image with name `books-api` from the dockerfile.

Run / Stop container

To run image, we can use command `docker run`:

```
docker run -d -p 8080:3000 books-api
```

This command use different option:

- `-d`: Esecuzione del container in modalità `detach` (background)
- `-p`: Mappa le porte tra HOST e container (es.: 8080:3000 fa puntare la porta host 8080 sulla porta 3000 del contenitore)

When we want to stop container we can use command `docker container <name of container> stop`

> NOTE: to show list of container running we can use command `docker ps`