Item CRUD Backend Service

Item management backend service written with Java SpringBoot.

- **healthcheck** endpoint is actually not required but services in clouder providers or load balancers require this healtcheck.
- In order to use this backend service **8080** port should be available on your local development machine if you want to start the service from your local.
- You can find sample **POSTMAN** collection in the folder.

This project assumes you had already installed these tools:

1. Docker
2. Java 11
3. Maven 3.6.3

Useful informations about project:

```
* API URL: http://54.229.233.147/v1
* Web Server: AWS EC2
* Database: AWS RDS MySQL
* CI/CD: Github Actions
```

Database Usage

You can connect the database using an ordinary mysql client. Configurations are below:

```
MYSQL_HOST = free-tier-db.cmesiok52al9.eu-west-1.rds.amazonaws.com
MYSQL_PORT = 3306
MYSQL_USER = admin
MYSQL_PASSWORD = Ay3ZVQTTXW2HxR
MYSQL_DB = item_db
```

API Usage

You can start the API from your local computer or using docker.

Below you can find the sample docker usage.

```
cd <this directory>
docker build -t java-springboot-be .
docker run -d -p 80:8080 java-springboot-be:latest
```

After starting the system, API will expose itself through port: `80`.

Below you can find the sample local build.

```
cd <this directory>
mvn clean install
mvn spring-boot:run
```

After starting the system, API will expose itself through port: `8080`.

Endpoints

There are different endpoints for this API:

1. `/v1/healthcheck`

- This endpoint accepts *GET* request.
    - This endpoint is just useful for understanding of the application live status.

2. `/v1/items`

- This endpoint accepts *POST* and *GET* request.
    1. *GET*
    - You can send three optional parameters *page*, *count*, *category*.
    - *page* and *count* can be used for pagination purposes and if you include *category* parameters, endpoint will bring only items related with searched category.
    2. *POST*
    - Insert item to database.
    - Body Params: `{ "name": "item17", [mandatory] "category": "spor", [mandatory] "price": 123123, [mandatory], "description": "lorem ipsum", [optional], "picture_url": "www.sample.com" [optional], }`

3. `/v1/items/:itemid`

- This endpoint accepts *GET*, *PUT*, *DELETE* requests.
    1. *GET*
    - Get the information about the item with given :itemid
    2. *PUT*
    - If item is in database this endpoint updates it, or else it is creating a new item.
    - Body Params: `{ "name": "item17", [mandatory] "category": "spor", [mandatory] "price": 123123, [mandatory], "description": "lorem ipsum", [optional], "picture_url": "www.sample.com" [optional], }`
    3. *DELETE*
    - Delete the item with given :itemid

Test Scenarios

Controller, **Service** and **Repository** unit tests have been implemented. In order to run tests, commands below could be used:

```
cd <this directory>
mvn clean test
```