HOMEWORK 7

**NoSQL DATABASES. ODM**

## Tasks

1. Install mongodb (any way acceptable but the usage of docker container is preferred).
2. Create database in mongodb.
3. Create collection **cities** and fill it with the mock data by adding multiple documents into it with the following schema (or a similar one):

|  |
| --- |
| {  name: ‘Brest’,  country: ‘Belarus’,  capital: false,  location: {  lat: 52.097621,  long: 23.734050  }  } |

1. Write a simple web server which will return a random cityon every request to it (you can modify already existed one).
2. Install mongoose package.
3. Make your solution for **task 4** use mongoose instead of the native implementation (define **city** model).
4. Create models for **user** and **product** via mongoose (use appropriate module files from **Homework 1**).
5. Generate mock data for users and products and importall of them via mongoose in **users** and **products** collections inside the database.
6. Add validations for appropriate fields of your models (e.g. capital field in **city** model).
7. Modify application to respond all routes from **Homework 4** and return data fromthe database.
8. Add additional routes and make your application responds on them:

|  |  |  |
| --- | --- | --- |
| **URL** | **METHOD** | **ACTION** |
| /api/users/:id | DELETE | Deletes *SINGLE* user |
| /api/products/:id | DELETE | Deletes *SINGLE* product |
| /api/cities | GET | Returns *ALL* cities |
| /api/cities | POST | Adds *NEW* city and returns it |
| /api/cities/:id | PUT | Updates *SINGLE* city by *id* if exists or adds *NEW* city with the given *id* otherwise |
| /api/cities/:id | DELETE | Deletes *SINGLE* city |

1. Implement a function which will add extra field called lastModifiedDate with the current date for every created/updated item (every **PUT** and **POST** request for all **user**, **product** and **city** entities).

## Evaluation Criteria

1. All required packages installed, database created (*tasks 1-2*).
2. Simple web server implemented which returns random city in response (*tasks 3-5*).
3. All models implemented and data imported to database (*task 6-8*).
4. Validations applied and all implemented routes return data from database (*tasks 9-10*).
5. All routes (including additional) return data from database with extra field added automatically on creation/update (*task* *11-12*).