Coding Task Solution by Stepanov Andrey

https://oauth:glpat-Lt-SiKrp3PLC24JPH3ix@gitlab.com/musala-coding-tasks-solutions/andrey-stepanov.git

In my model I made the following assumptions:

- drone weight is actually the weight of loaded medications
- therefore initially the (just registered) drone has a zero weight and IDLE state

The check for the drones available for loading takes the following requirements into account:

- only the drones with IDLE or LOADING status may be available
- the drone must be prevented from being in LOADING state if the battery level (capacity) is below 25%
- loading each new medication increases the drone weight so keep an eye on the overload.

  - loading process results in the status change for LOADING if only the overall weight of loaded medications does not exceed the weight limit of 500 g

  - when the weight limit is reached, the message 'medication overall weight exceeds allowed limit' is issued

Adding a new drone is possible only if drone fleet size limit (10) is not exceeded - the message 'no more drones can be registered' is shown

Some notes on implementation

Not according to the common practice I use natural primary keys for both the drone and medication tables/entities.

They work just fine

I have chosen H2 database, in-memory variant which is volatile, and the data is lost on application restart. The database is up and running on application start. Postman must be enough for the interaction with Rest API and H2. Still H2 Console is available at http://localhost:8080/h2-console/. Employ 'admin' as user name and 'password' as password to log into.

To translate custom business and validation exceptions into a better readable form (so that POSTMAN shows not just a wrong request status but a cause of exception) I introduced ExceptionApiHandler

Validation of the drone and medication parameters is implemented at the controller layer

Registering drones and medications is implemented by one

Assigning medications to the drone is done in batch. Sure you may load the drone with a single medication as well

Removing medications and drones was not mentioned as a functional requirements and is left out of the scope of the application

Sure it would be advantageous to add Swagger but I decided that it would be more decorative than functional for this task

A few notes on configuration

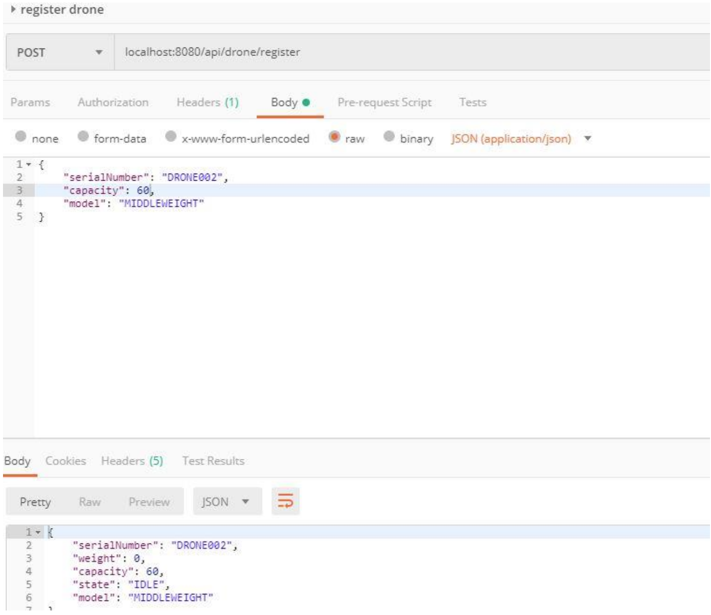
Periodic task to check battery levels is managed by the property 'scheduler.interval' and currently runs once a minute

Log file for the periodic battery check is saved to the 'logs' subfolder within the project directory but that is adjustable by setting the 'logging.file.name' property

Postman collection of the requests is in /resources/static/musala.postman_collection.json

Below are some of the screens representing functionality of the API

Please do not hesitate to ask if something remains unclear or I forgot to mention (or implement or push) some things or made some silly mistake while pushing to repo



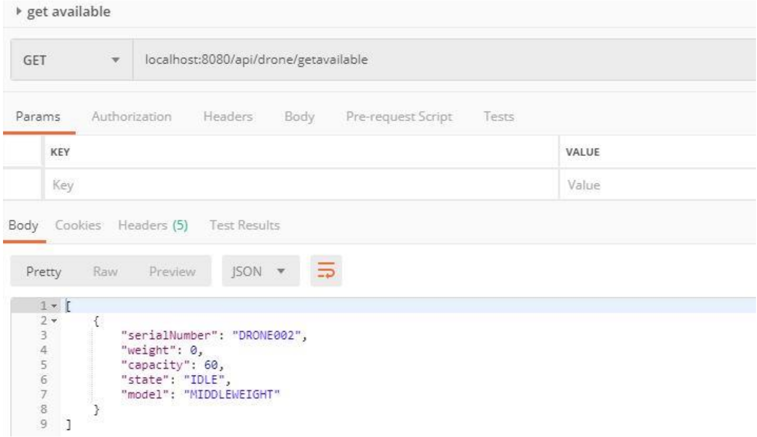
POST localhost:8080/api/drone/register

raw JSON (application/json)

```
{
    "serialNumber": "DRONE002",
    "capacity": 60,
    "model": "MIDDLEWEIGHT"
}
```

JSON

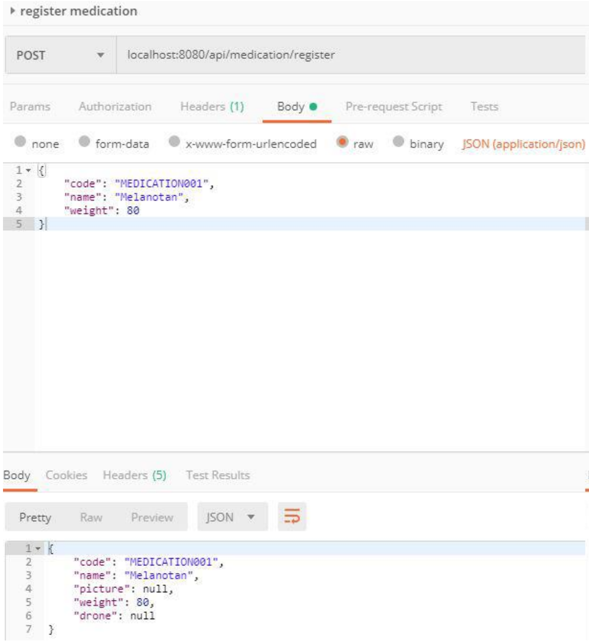
```
{
    "serialNumber": "DRONE002",
    "weight": 0,
    "capacity": 60,
    "state": "IDLE",
    "model": "MIDDLEWEIGHT"
```



GET localhost:8080/api/drone/getavailable

JSON

```
[
    {
        "serialNumber": "DRONE002",
        "weight": 0,
        "capacity": 60,
        "state": "IDLE",
        "model": "MIDDLEWEIGHT"
    }
]
```



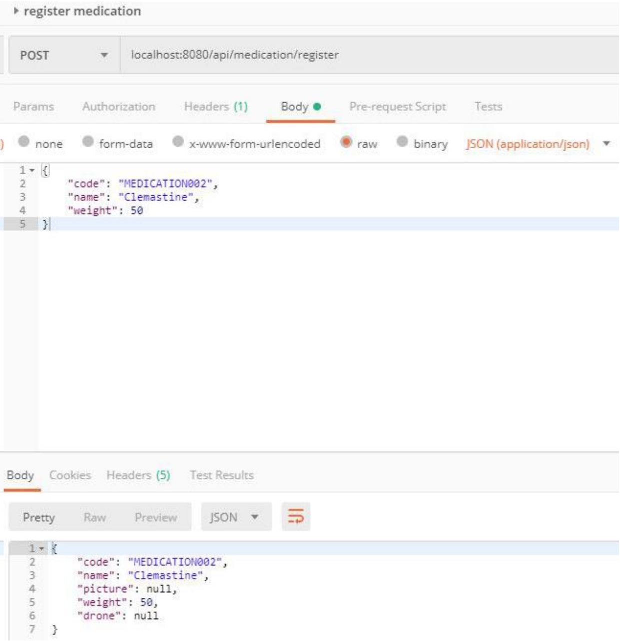
POST localhost:8080/api/medication/register

raw JSON (application/json)

```
{
    "code": "MEDICATION001",
    "name": "Melanotan",
    "weight": 80
}
```

JSON

```
{
    "code": "MEDICATION001",
    "name": "Melanotan",
    "picture": null,
    "weight": 80,
    "drone": null
}
```



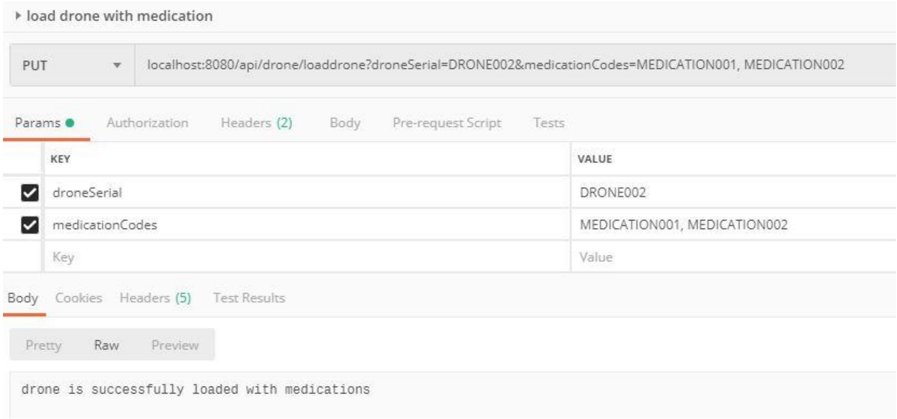
POST localhost:8080/api/medication/register

raw JSON (application/json)

```
{
    "code": "MEDICATION002",
    "name": "Clemastine",
    "weight": 50
}
```

JSON

```
{
    "code": "MEDICATION002",
    "name": "Clemastine",
    "picture": null,
    "weight": 50,
    "drone": null
}
```

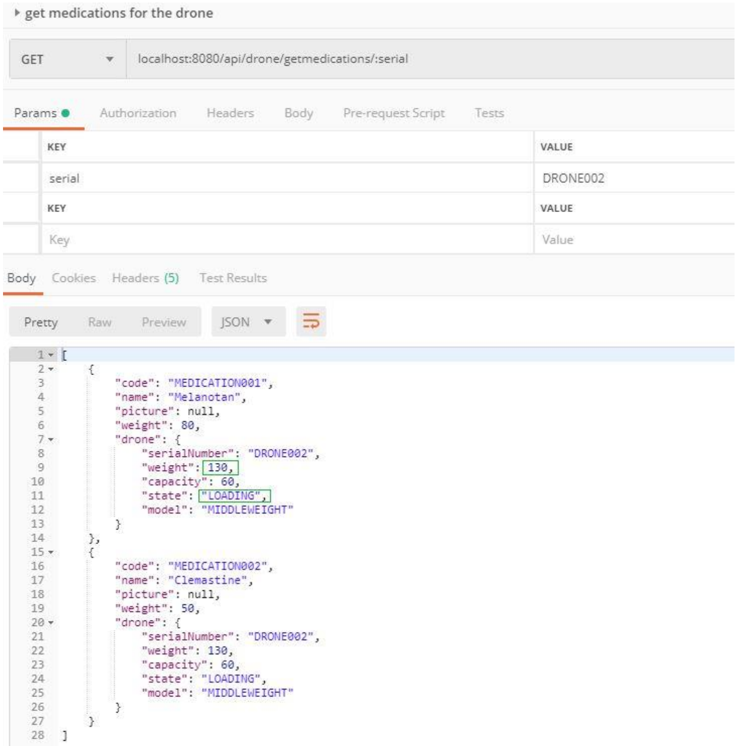


load drone with medication

PUT localhost:8080/api/drone/loaddrone?droneSerial=DRONE002&medicationCodes=MEDICATION001, MEDICATION002

	KEY	VALUE
☑	droneSerial	DRONE002
☑	medicationCodes	MEDICATION001, MEDICATION002
	Key	Value

```
drone is successfully loaded with medications
```

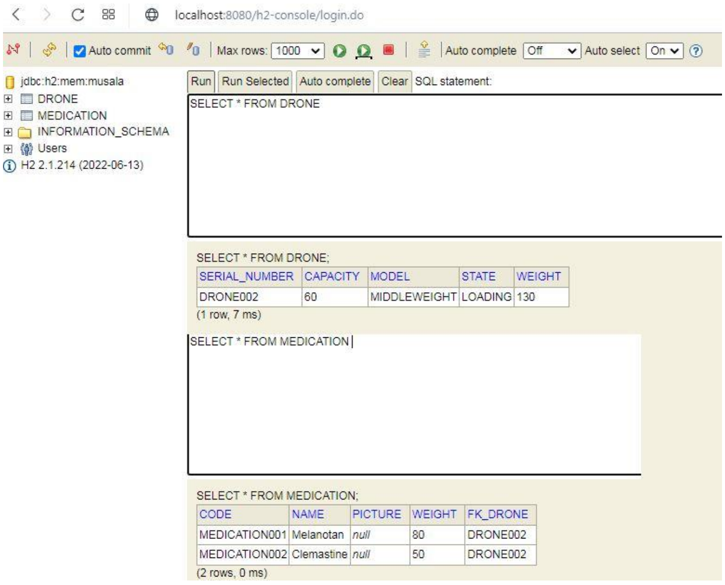


get medications for the drone

GET localhost:8080/api/drone/getmedications/:serial

KEY	VALUE
serial	DRONE002
KEY	**VALUE**
Key	Value

```
[
    {
        "code": "MEDICATION001",
        "name": "Melanotan",
        "picture": null,
        "weight": 80,
        "drone": {
            "serialNumber": "DRONE002",
            "weight": 130,
            "capacity": 60,
            "state": "LOADING",
            "model": "MIDDLEWEIGHT"
        }
    },
    {
        "code": "MEDICATION002",
        "name": "Clemastine",
        "picture": null,
        "weight": 50,
        "drone": {
            "serialNumber": "DRONE002",
            "weight": 130,
            "capacity": 60,
            "state": "LOADING",
            "model": "MIDDLEWEIGHT"
        }
    }
]
```



localhost:8080/h2-console/login.do

☑ Auto commit | Max rows: 1000 | Auto complete: Off | Auto select: On

```
SELECT * FROM DRONE
```

SELECT * FROM DRONE;

SERIAL_NUMBER	CAPACITY	MODEL	STATE	WEIGHT
DRONE002	60	MIDDLEWEIGHT	LOADING	130

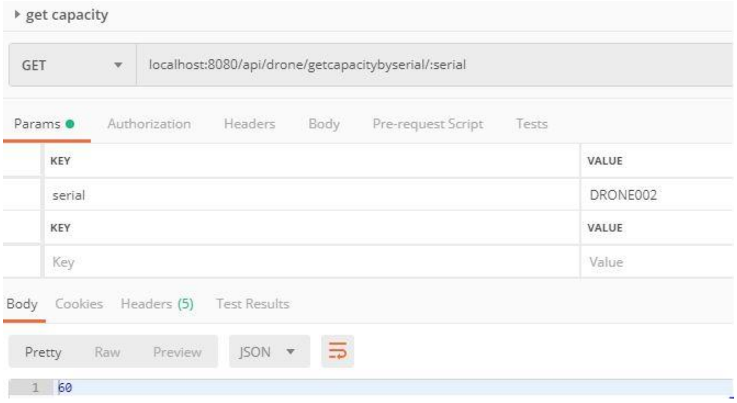
(1 row, 7 ms)

```
SELECT * FROM MEDICATION
```

SELECT * FROM MEDICATION;

CODE	NAME	PICTURE	WEIGHT	FK_DRONE
MEDICATION001	Melanotan	*null*	80	DRONE002
MEDICATION002	Clemastine	*null*	50	DRONE002

(2 rows, 0 ms)

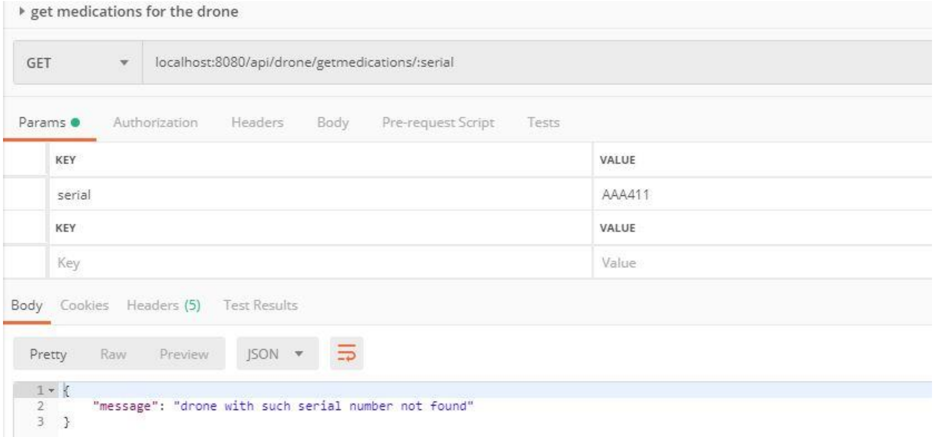


get capacity

GET localhost:8080/api/drone/getcapacitybyserial/:serial

KEY	VALUE
serial	DRONE002
KEY	VALUE
Key	Value

```
60
```

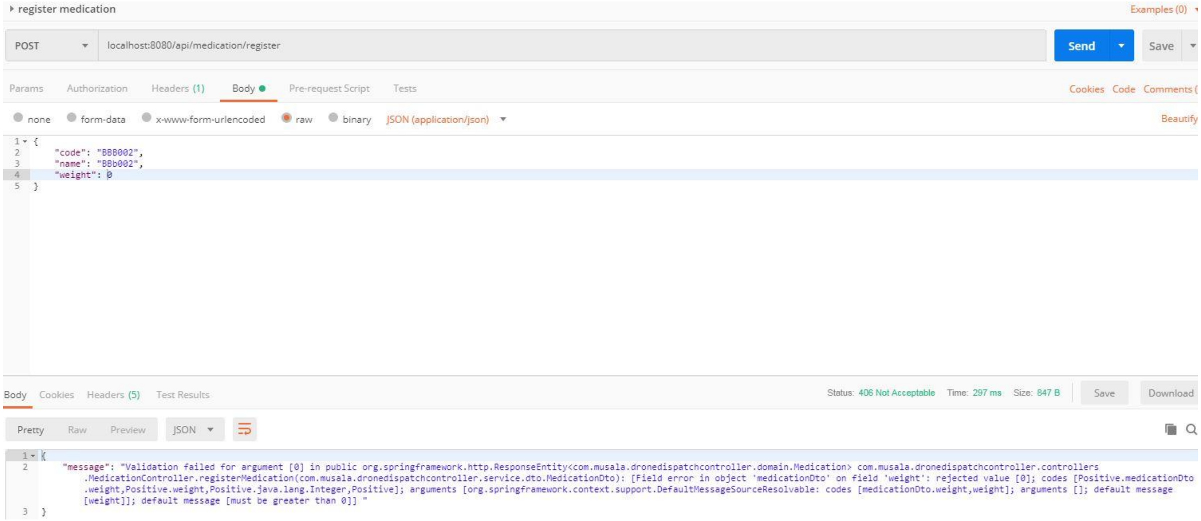


get medications for the drone

GET localhost:8080/api/drone/getmedications/:serial

KEY	VALUE
serial	AAA411
KEY	VALUE
Key	Value

```
{
    "message": "drone with such serial number not found"
}
```



register medication

POST localhost:8080/api/medication/register

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary JSON (application/json)

```
{
    "code": "BBB002",
    "name": "BBb002",
    "weight": 0
}
```

Status: 406 Not Acceptable Time: 297 ms Size: 847 B

```
{
    "message": "Validation failed for argument [0] in public org.springframework.http.ResponseEntity<com.musala.dronedispatchcontroller.domain.Medication> com.musala.dronedispatchcontroller.controllers
        .MedicationController.registerMedication(com.musala.dronedispatchcontroller.service.dto.MedicationDto): [Field error in object 'medicationDto' on field 'weight': rejected value [0]; codes [Positive.medicationDto
        .weight,Positive.weight,Positive.java.lang.Integer,Positive]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [medicationDto.weight,weight]; arguments []; default message
        [weight]]; default message [must be greater than 0]] "
}
```