Metadata storage installation and usage instructions

Installation

MongoDB

Download mongoDB Community Server from here:

https://www.mongodb.com/download-center/community

Install mongoDB Community Server. During "Service configuration", specify a desired directory for data and logs (or keep the default ones). MongoDB compass is not necessary, if you are asked to install it.

NOTE: After installation, the service probably runs automatically (at least in Windows), i.e. you can run the `mongo` command below. If the service has not started, you can start it using the `mongod` program, in the `cd <mongodb installation dir>/bin` directory, providing a port as a parameter, if you need to change the default port (27017). See https://docs.mongodb.com/manual/reference/program/mongod/ for mor information.

Node.js

Download node.js from here:

https://nodejs.org/en/

Install node.js.

Metadata storage

Extract "metadata_storage.zip" in a desired directory. From the command line, move to the `metadata_storage` directory.

```
cd metadata_storage
```

Run the following:

```
npm install
```

This will download the necessary dependencies.

The code assumes that mongoDB runs at the default port 27017. If mongoDB runs at a different port, you need to change the port manuall from within the `server.js` file, by changing the following line:

```
var dbUrl = "mongodb://localhost:27017";
```

Usage

Start the API

To run the metadata storage API, run the following command, from the `metadata_storage` directory:

```
node server.js
```

This will start a server, listening to port 8081. If you want the server to run at a different port, you need to change the port manually by editing `server.js` at the following line:

```
var server = app.listen(8081, function () {
```

You can then use a REST client (Postman or any other), to call the API. Below are some examples.

Create a model.

URL: http://localhost:8081/createModel

Method: POST

Headers: Content-Type: application/json

Body:

```
{
    "modelID": "test_model_1",
    "modelParams": {
        "a": 6,
        "b": 7
    }
}
```

Expected response:

```
{
    "message": "The model was created successfully"
}
```

or:

```
{
    "message": "A model with modelID 'test_model_1' already exists."
}
```

Get a model.

URL: http://localhost:8081/getModel

Method: POST

Headers: Content-Type: application/json

Body:

```
{
    "modelID": "test_model_1"
}
```

Expected response:

```
{
    "_id": "5c544348c6393e1bd4b48039",
    "modelID": "test_model_1",
    "modelParams": {
        "a": 6,
        "b": 7
    }
}
```

Edit a model.

URL: http://localhost:8081/editModel

Method: POST

Headers: Content-Type: application/json

Body:

```
{
    "modelID": "test_model_1",
    "modelParams": {
        "a": 45
    }
}
```

Expected response:

```
{
    "_id": "5c544306c6393e1bd4b48038",
    "modelID": "test_model_1",
    "modelParams": {
        "a": 45,
        "b": 7
    }
}
```

Delete a model.

URL: http://localhost:8081/deleteModel

Method: POST

Headers: Content-Type: application/json

Body:

```
{
    "modelID": "test_model_1"
}
```

Expected response:

```
{
    "message": "The model was deleted successfully"
}
```

Unit tests

To run the unit tests for the metadata storage, run the following from the `metadata_storage` directory:

```
npm test
```

A number of tests will appear, each with a tick at the left, in case of success, and the time needed for execution at the right.