

README

I decided to implement the solution using NestJS, as it is one of the most commonly used JavaScript backend frameworks.

I used MongoDB as the database, with auto-generated ids

Example id: "66bd3679252175147b5f4747"

There must be a **MongoDB** server running at <http://localhost:27017>

The app will produce a **"course-db"** database automatically.

And it will create a collection called **"courses"** where there will be all the courses.

Modules to be installed:

```
npm i -g @nestjs/cli
npm install @nestjs/mongoose mongoose class-validator class-transformer
npm install --save-dev @nestjs/testing jest @types/jest ts-jest
npm install mongodb-memory-server
```

To run the app:

```
npm run start:dev
```

To test the endpoints:

```
npm run test
```

To access the REST API:

Server Endpoint:

<http://localhost:3000>

1) POST /courses/create

- Create a new course.

`http://localhost:3000/courses/create`

BODY

```
{
  "subject": "BIO",
  "courseNumber": "102",
  "description": "Introduction to Biology"
}
```

RETURN VALUE

```
{
  "subject": "BIO",
  "courseNumber": "102",
  "description": "Introduction to Biology",
  "_id": "66bd3679252175147b5f4749",
  "__v": 0
}
```

Examples:

Creating a new Course

The screenshot displays the Postman interface for a REST client. On the left sidebar, the 'CIVITAS' collection is expanded, and the 'CREATE' endpoint is selected. The main panel shows a POST request to `http://localhost:3000/courses/create`. The request body is a JSON object: `{ "subject": "BIO", "courseNumber": "102", "description": "Introduction to Biology" }`. The response is also in JSON format, showing the created course with an `_id` and `__v` field. The status bar at the bottom right indicates a `201 Created` status.

Request Details:

- Method: POST
- URL: `http://localhost:3000/courses/create`
- Body Type: JSON
- Body:

```
{
  "subject": "BIO",
  "courseNumber": "102",
  "description": "Introduction to Biology"
}
```

Response Details:

- Status: 201 Created
- Body:

```
{
  "subject": "BIO",
  "courseNumber": "102",
  "description": "Introduction to Biology",
  "_id": "66bd3679252175147b5f4749",
  "__v": 0
}
```

Attempt to create a Course with existing Subject:

The screenshot shows a REST client interface with a sidebar on the left containing a list of API endpoints under the 'CIVITAS' section. The main panel displays a POST request to `http://localhost:3000/courses/create`. The request body is a JSON object: `{ "subject": "PHY", "courseNumber": "105", "description": "Introduction to Physics" }`. The response status is `409 Conflict`, and the response body is `{ "message": "Course with this subject already exists.", "error": "Conflict", "statusCode": 409 }`, which is highlighted with a red box.

Attempt to create a Course with existing Number:

The screenshot shows the same REST client interface, but the request body is now `{ "subject": "LNG", "courseNumber": "105", "description": "Introduction to Physics" }`. The response status is `409 Conflict`, and the response body is `{ "message": "Course with this number already exists.", "error": "Conflict", "statusCode": 409 }`, which is highlighted with a red box.

2) GET /courses

- Gets All Courses.

`http://localhost:3000/courses`

Example:

The screenshot shows a REST client interface with the following details:

- Method:** GET (highlighted with a red box)
- URL:** `http://localhost:3000/courses` (highlighted with a red box)
- Body:** none (selected)
- Status:** 200 OK (highlighted with a red box)
- Response Body (JSON):** (highlighted with a red box)

```
1 {
2   "id": "66bd3679252175147b5f4747",
3   "subject": "BIO",
4   "courseNumber": "101",
5   "description": "Introduction to Biology",
6   "v": 0
7 },
8 {
9   "id": "66bd3679252175147b5f4749",
10  "subject": "BIO",
11  "courseNumber": "102",
12  "description": "Introduction to Biology",
13  "v": 0
14 }
15 ]
16
```

3) GET /courses/description

- Gets All Courses searching in the Description field.

```
http://localhost:3000/courses/description?description=<TEXT>
```

Headers:

```
description:string
```

The value to search into the "description" field

Examples:

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:3000/courses/description?description=Biology
- Query Params:** A table with one entry:

KEY	VALUE
description	Biology
- Status:** 200 OK
- Response Body (JSON):**

```
[{"_id":"66bd3679252175147b5f4747","subject":"BIO","courseNumber":"101","description":"Introduction to Biology","_v":0}, {"_id":"66bd3679252175147b5f4749","subject":"BIO","courseNumber":"102","description":"Introduction to Biology","_v":0}]
```

4) DELETE /courses/{id}

- Deletes Course by id.

```
http://localhost:3000/courses/<ID>
```

Examples:

Deleting an existing record:

The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:3000/courses/66bd3679252175147b5f4749`. The request is configured with the following parameters:

KEY	VALUE
Key	Value

The response is displayed in the 'Body' tab, showing a single item with the key '1' and the value 'true'.

Attempting to delete a nonexistent id:

The screenshot shows the Postman interface for a DELETE request to a nonexistent course ID. The URL is `http://localhost:3000/courses/66bd3679252175147b5f4749`. The response is displayed in the 'Body' tab, showing a JSON object with the following structure:

```
{ "message": "Course not found", "error": "Not Found", "statusCode": 404 }
```

The status bar at the bottom right indicates the response status: **Status: 404 Not Found**.

5) DELETE/courses/id

- Deletes Course by id.

`http://localhost:3000/courses/id?id=<ID>`

Examples:

Existing ID

The screenshot shows the Postman interface for a REST client. The left sidebar lists the API endpoints under the 'CIVITAS' collection, with 'GET FIND BY ID' selected. The main panel shows the request details for the endpoint `http://localhost:3000/courses/id?id=66bd7f42952e3763c323adff`. The 'Params' tab is active, showing a query parameter 'id' with the value '66bd7f42952e3763c323adff'. The 'Body' tab shows the response in JSON format, which is a course object with the same ID. The status is '200 OK'.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	66bd7f42952e3763c323adff	
Key	Value	Description

```
1 { "_id": "66bd7f42952e3763c323adff", "subject": "MAT", "courseNumber": "103", "description": "Introduction to Mathematics", "_v": 0 }
```

Wrong ID

The screenshot shows the Postman interface for a REST client. The left sidebar lists the API endpoints under the 'CIVITAS' collection, with 'GET FIND BY ID' selected. The main panel shows the request details for the endpoint `http://localhost:3000/courses/id?id=66bd7f42952e3763c323adf0`. The 'Params' tab is active, showing a query parameter 'id' with the value '66bd7f42952e3763c323adf0'. The 'Body' tab shows the response in JSON format, which is an error message indicating that the course was not found. The status is '404 Not Found'.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	66bd7f42952e3763c323adf0	
Key	Value	Description

```
1 { "message": "Course with ID 66bd7f42952e3763c323adf0 not found", "error": "Not Found", "statusCode": 404 }
```

TESTS:

```
npm run test

> civitas@0.0.1 test
> jest

PASS src/app.controller.spec.ts
PASS src/course/course.controller.spec.ts

Test Suites: 2 passed, 2 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.584 s, estimated 5 s
Ran all test suites.
```