# ****Prerequisites****

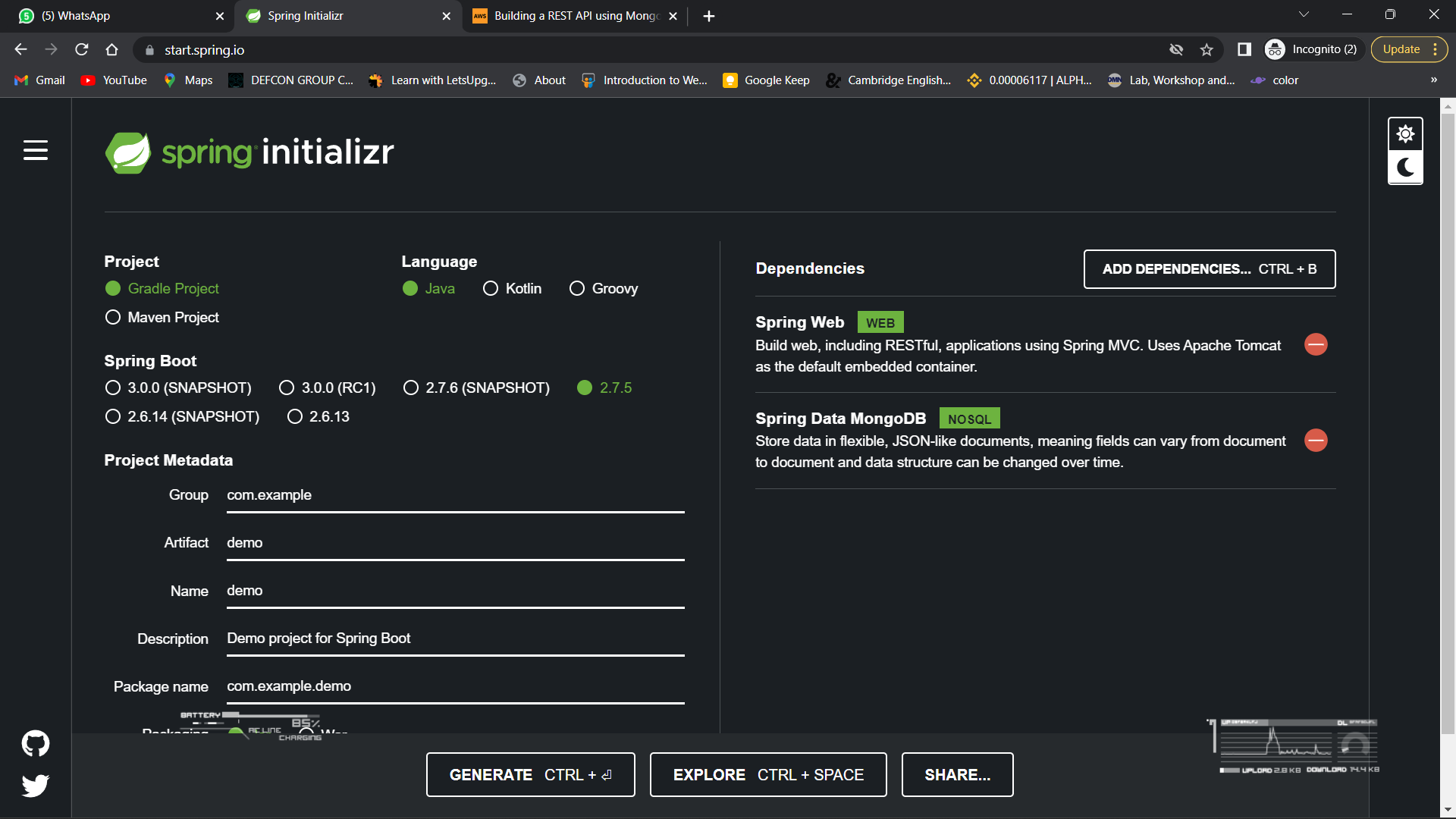
In order to complete this tutorial, you should have knowledge of beginner or intermediate Java 8 programming skills, some familiarity with Spring Boot, and also you should have a general familiarity with the Windows command prompt.

# Tools Used in this Project

1. Java 8
2. Spring Boot
3. Gradle
4. MongoDB
5. Postman

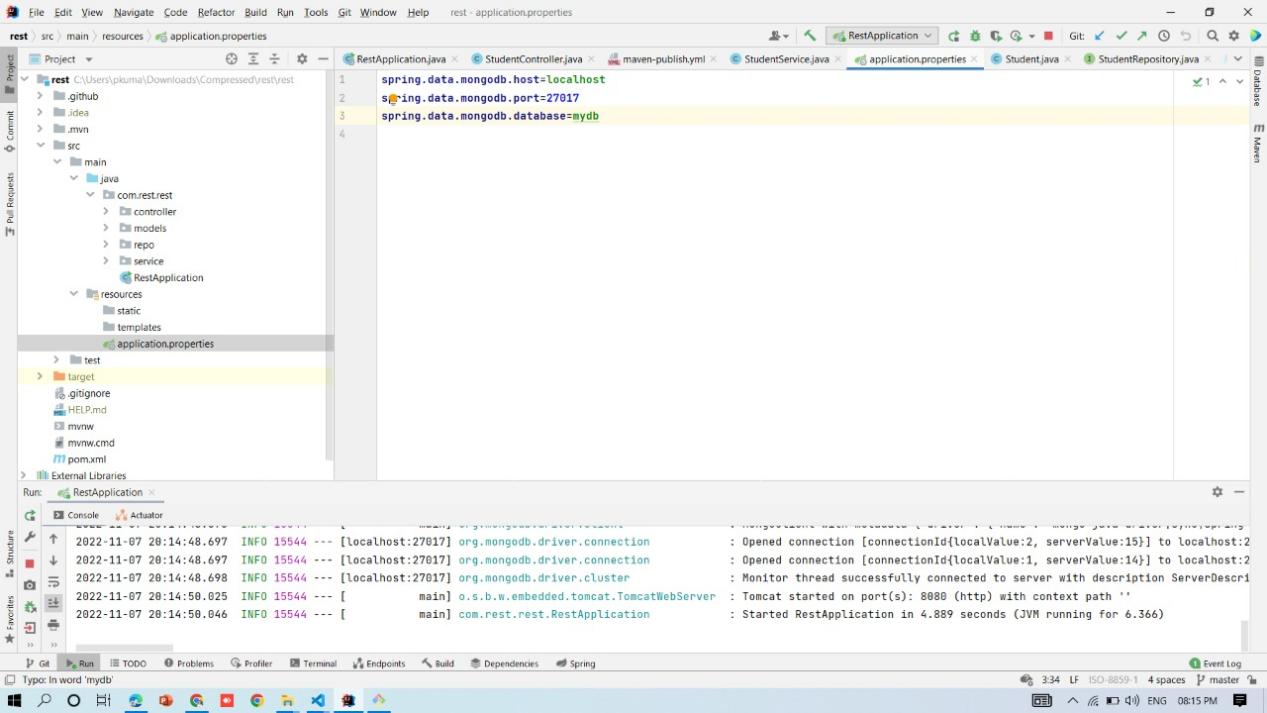
# ****Step 2. Spring boot Project Setup****

We will make use of the Spring Initializr tool for quickly setting up the project. Don’t forget to add the dependencies Spring WEB & Spring Data MongoDB.



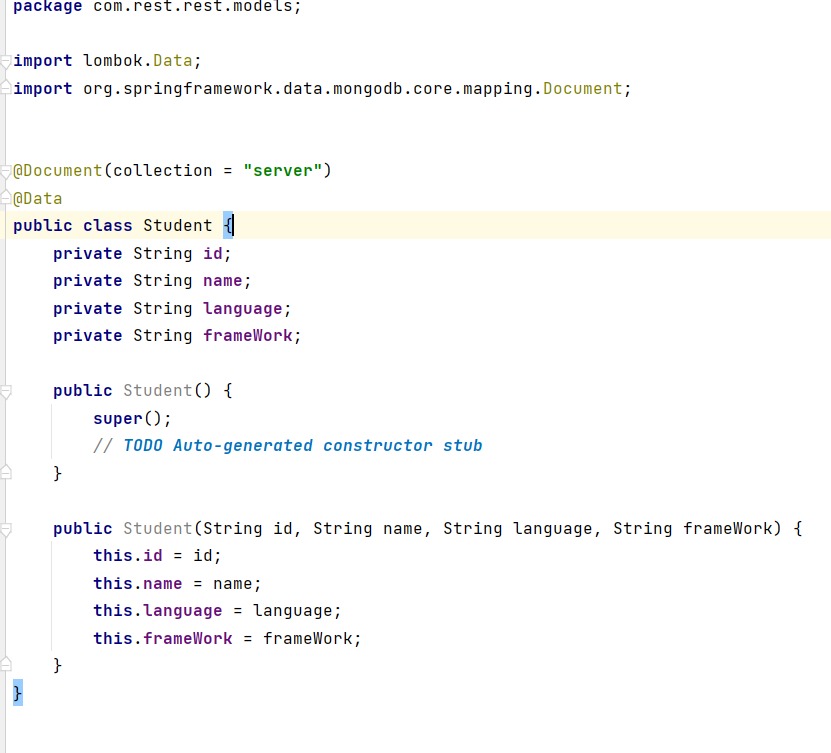
# ****Step 3. Configure MongoDB****

To configure our MongoDB in Spring boot, we will need to add connection details of our DB to the **application.properties** file, located in the “src/main/resources” folder. Add the following lines to the property file and replace the information in brackets with the information specific to your MongoDB instance:



# Adding Model to Spring boot Project

Let’s start by create a simple Model class.

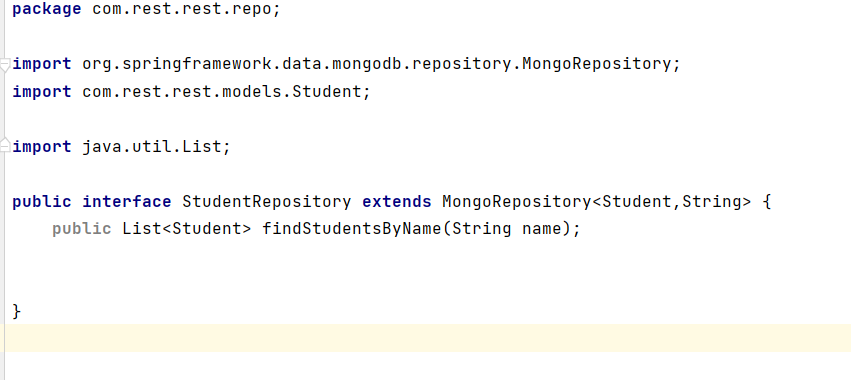


Here you have a Student class with four attributes: id, name, language, and frameWork. The **@Document** annotation marks a class as being a domain object that we want to persist to the database. Note: This annotation is the Mongo equivalent of **@Entity** in JPA.

The id is mostly for internal use by MongoDB. The **@Id** annotation tells spring that the id field will be used as the primary identifier. The rest of the class contains the basic constructors,and in place of getters, and setters I used lombook for Student object.

# ****Step 5. Spring Data MongoDB — MongoRepository****

We will now create a repository by making an interface.

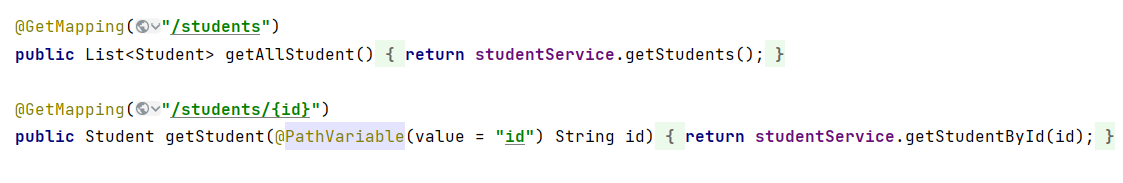


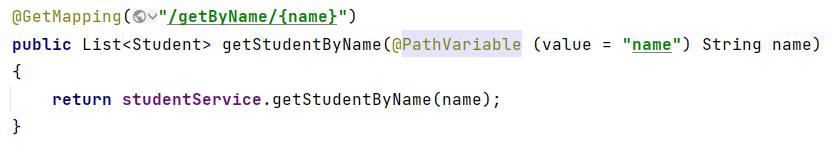
StudentRepository extends the MongoRepository interface and plugs in the type of values and ID that it works with: Movie and Integer, respectively. This will give us access to all the CRUD (create, read, update, and delete) operations around the MongoDB collection.

# ****Step 6. Defining the Controller****

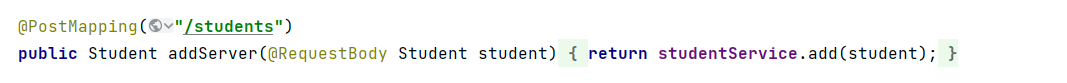
Finally, we will create the REST controller. The APIs which we will be creating will access the studentService.

**GET - Get all StudentData & Get student by ID & Get student by NAME**

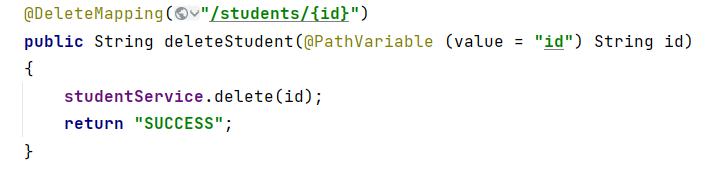
****

****

**POST - Add a STUDENT**

****

**DELETE - Delete a student**



## ****Completed Controller****

Here is the completed controller with all the methods added to it

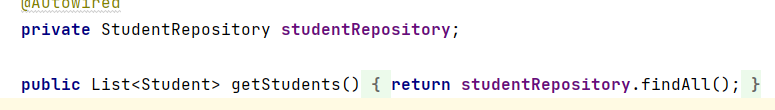


# ****Step 6. Defining the Service****

Finally, we will create the Service. The APIs which we will be creating will access the studentRepository dependency, which will internally use Spring Data MongoRepository API. NOTE: We do not have to write any database interaction code in the interface as Spring Data does it all for us ;).

## ****Adding the REST Endpoints****

The @Autowired annotation creates an instance of the studentRepository object that will allow us to access and modify the student database.



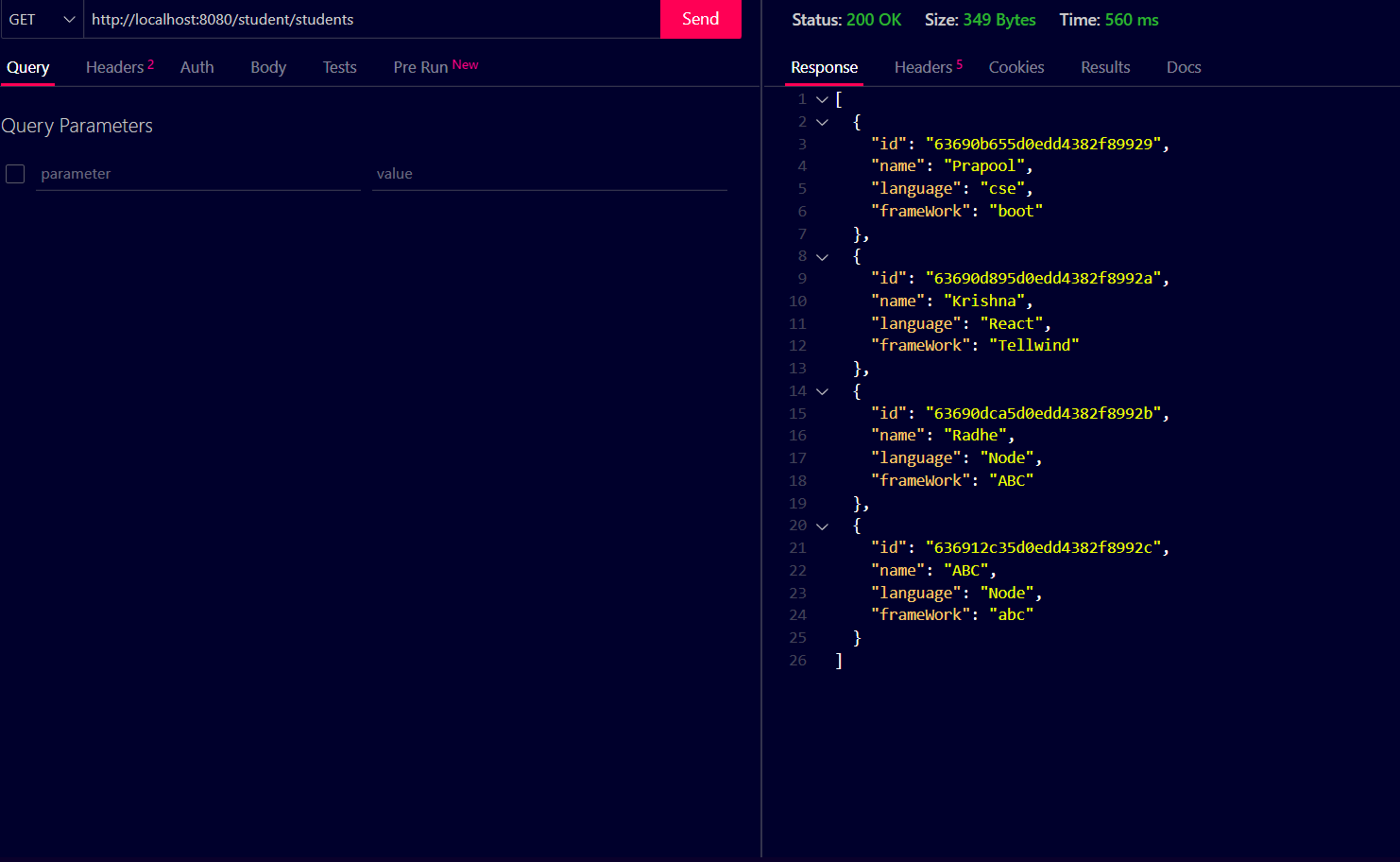
Service

Graphical user interface, text, application, email

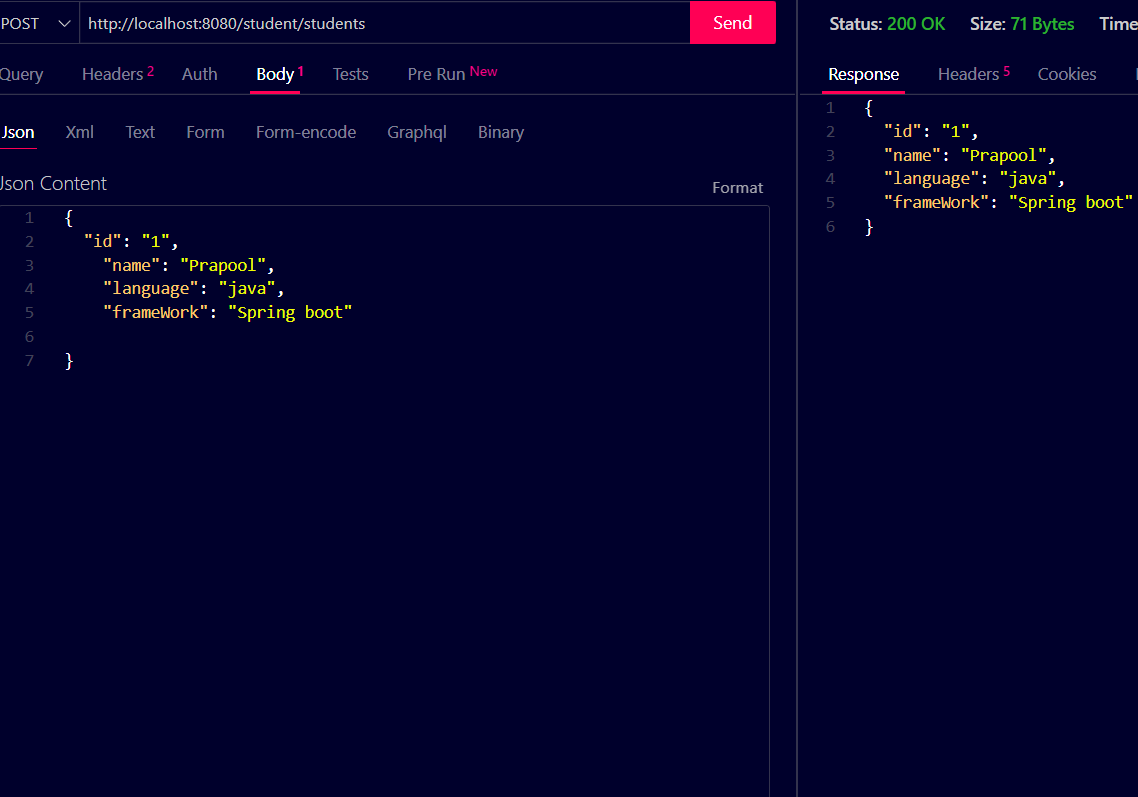
Description automatically generated

Output

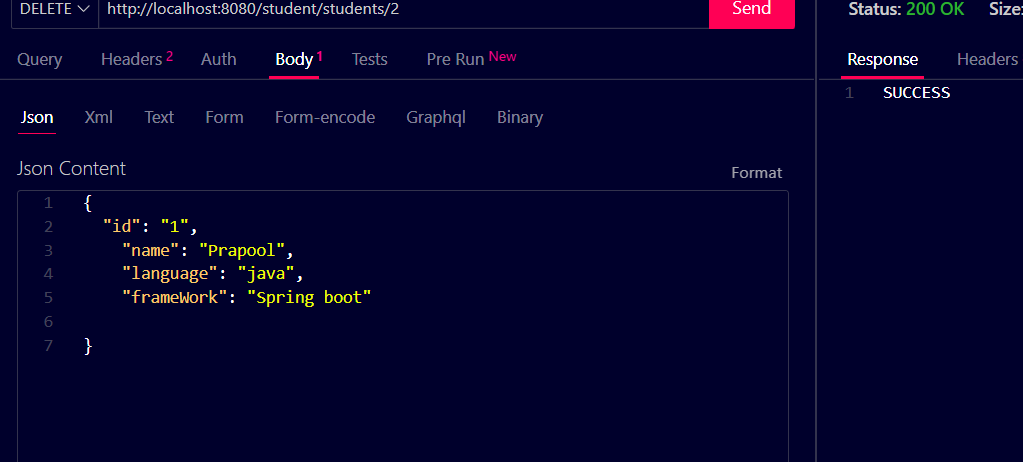
Get Method Output



Post



Delete



On MongoDb compass

