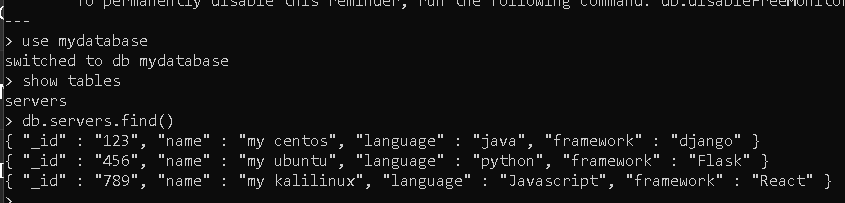
Detailed Explanation of Task1 .

Software IDE Used: IntellijIDEA

MongoDB Database name: mydatabase

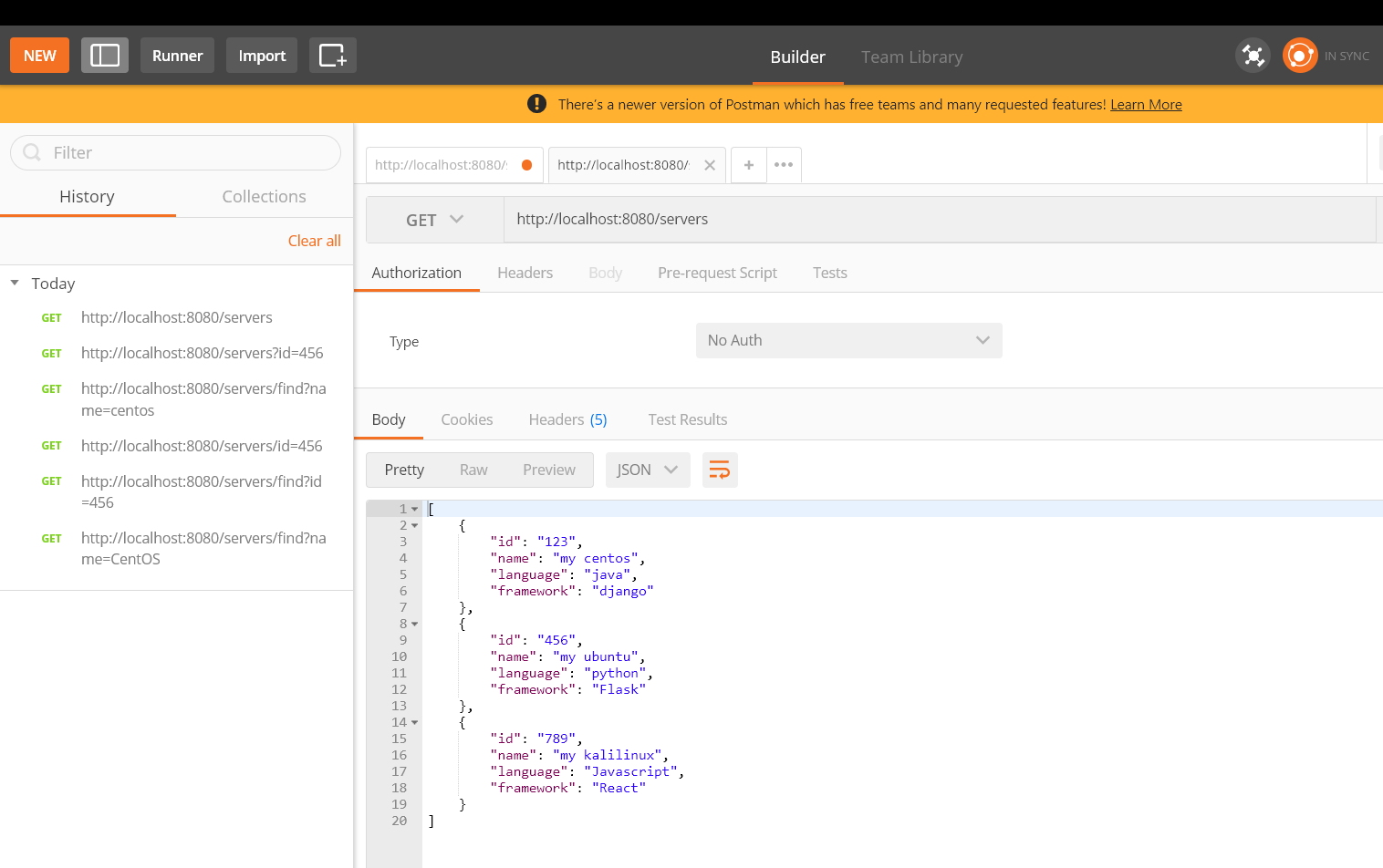
MongoDB collection name: servers



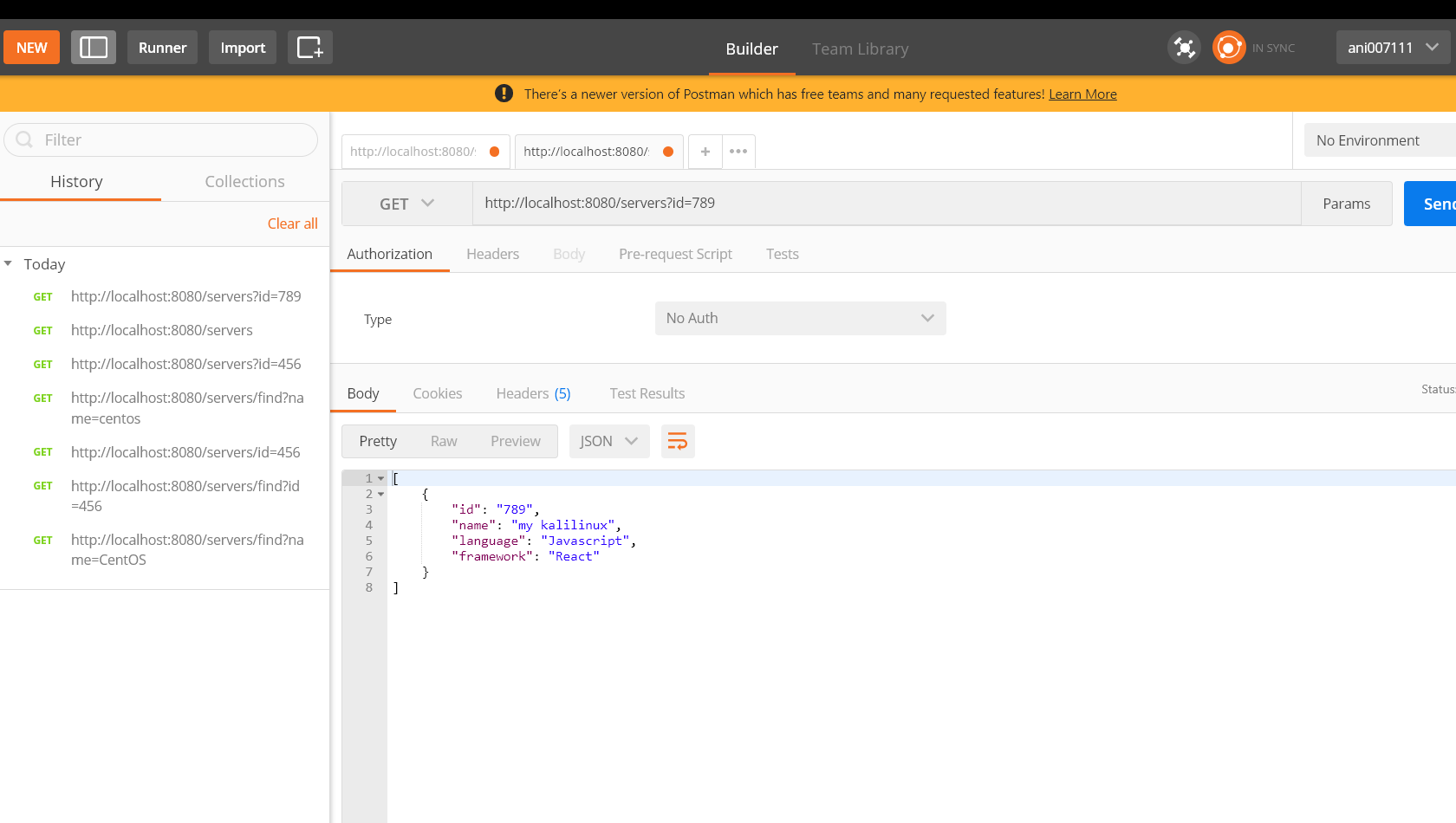
Now I test the http request flow using Postman API extension:

● GET servers. Should return all the servers if no parameters are passed. When server id is passed as a parameter

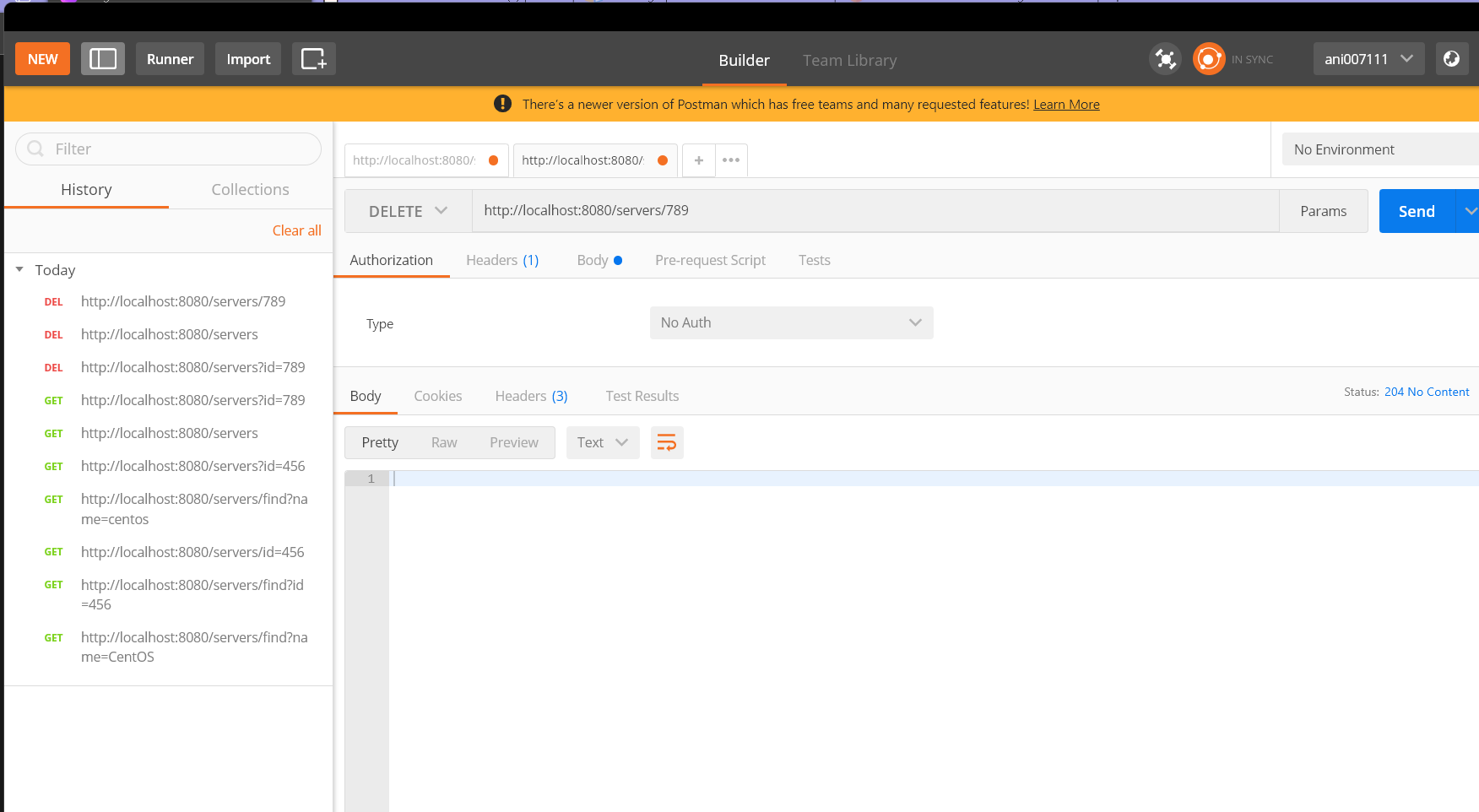
First i fetch details of all the servers present in the mongodb database.



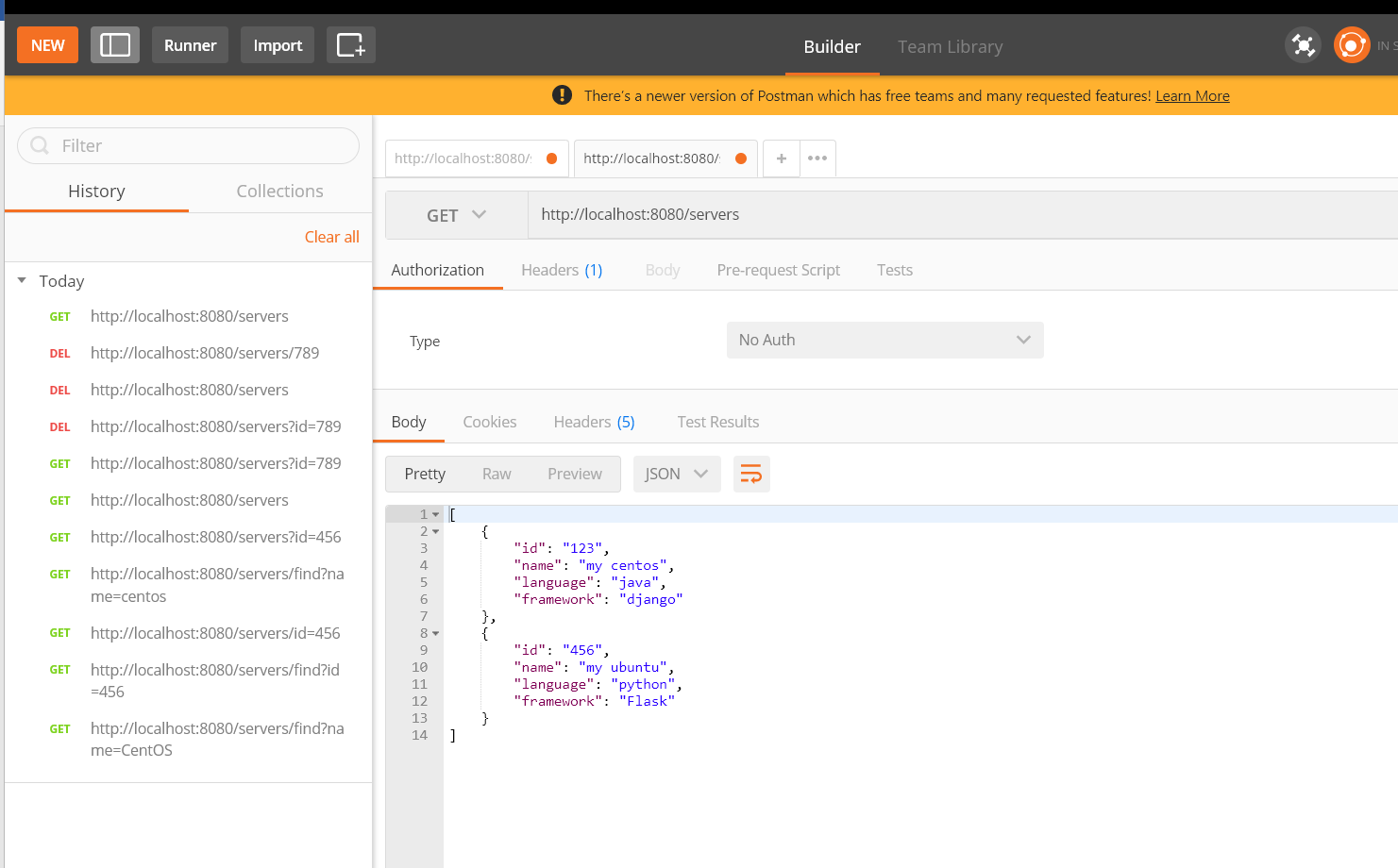
Now i fetch details of all the servers by calling id assigned to each servers object in mongodb.



● DELETE a server. The parameter is a server ID. Now i specify the id of server to be deleted in the url. For deleting we need to use DELETE request type.



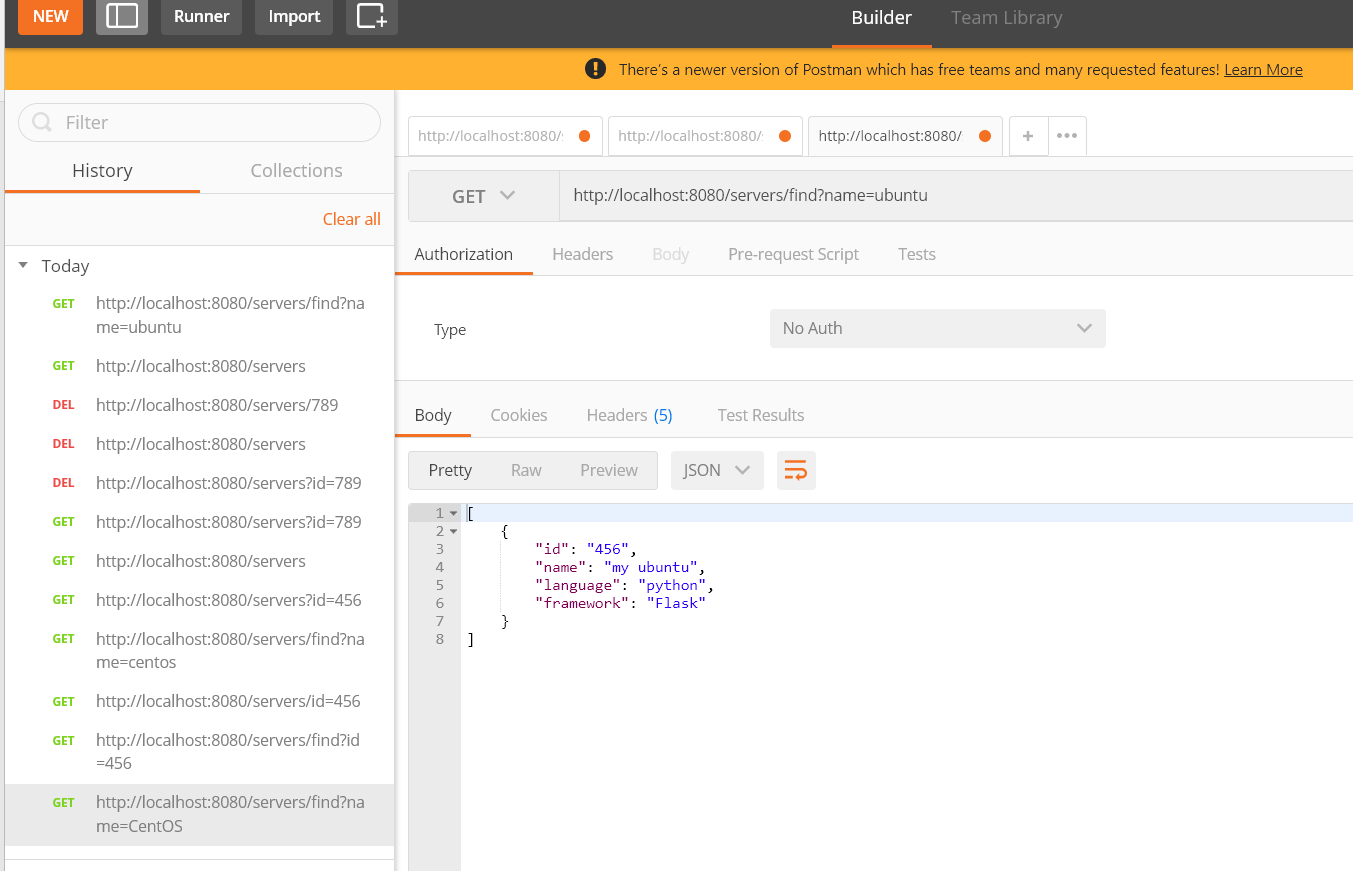
Below request to see if changes have been made after delete request of server id=789.



We can see server with id=789 has been deleted

● GET (find) servers by name. The parameter is a string. Must check if a server name contains this string

Now im getting server name "Ubuntu" by name parameter:



Explanation part of each files used in the project. We need to add dependencies for the spring boot in order to connect it with mongodb.

Add he following dependencies in build.gradle file

dependencies {

implementation 'org.springframework.boot:spring-boot-starter-data-mongodb'

implementation 'org.springframework.boot:spring-boot-starter-web'

implementation 'org.springframework.boot:spring-boot-starter-actuator'

testImplementation 'org.springframework.boot:spring-boot-starter-test'

}

**Server.java**

The purpose of the Server.java file is to define a Java class called Server that represents a server object with a name, language, and framework. This class is intended to be used in a Spring application that interacts with a MongoDB database.

The class includes a constructor that takes three arguments to initialize the instance variables for a Server object. It also includes getter and setter methods for each instance variable, which allows other classes to access and modify the values of these variables.

The class is annotated with the @Document annotation, which specifies the name of the MongoDB collection where the Server objects will be stored. Additionally, the @Id annotation is used to mark the id instance variable as the document ID for the MongoDB collection.

**Application.properties**

Since I have used springboot framework as a base for development we need to integrate it with our mongodb. We can define our database details in this file.

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database=mydatabase

**ServerRepository.java**

First we import the necessary classes for the ServerRepository class. The MongoRepository class is part of the Spring Data MongoDB library and provides a set of methods for interacting with MongoDB. The Repository annotation is used to mark the ServerRepository class as a Spring repository component.

The ServerRepository interface and extends the MongoRepository interface, which provides a set of methods for interacting with MongoDB. The ServerRepository interface is generic and takes two type parameters: the Server class, which represents the entity that will be stored in the MongoDB database, and String, which represents the type of the document ID.

Then we declare a custom method that can be used to search for servers by name. This method takes a single String parameter, name, which represents the name of the server to search for. The method returns a list of Server objects that contain the specified name parameter as a substring, regardless of case.

The purpose of the ServerRepository interface is to provide a set of methods for interacting with a MongoDB database that stores Server objects. The interface extends the MongoRepository interface, which provides a set of CRUD (Create, Read, Update, Delete) methods for interacting with the database. The custom method findByNameContainingIgnoreCase is a method that is not included in the MongoRepository interface, but is specific to the Server entity and allows for more targeted searches by name.

**ServerController.java**

This is a Java class file named ServerController that contains the implementation of REST API endpoints for CRUD (Create, Read, Update, Delete) operations on a Server model. The purpose of this file is to handle HTTP requests and responses related to the Server model

The getAllServers method returns a list of all servers or a single server by ID, depending on the value of the id query parameter. The getServersByName method returns a list of servers that match a given name. The updateServer method updates a server by ID with the given server object. The deleteServer method deletes a server by ID.

These methods use the ServerRepository object to interact with the underlying database and return appropriate HTTP responses based on the success or failure of the requested operation. The annotations used in this file, such as @RestController, @RequestMapping, @GetMapping, @PutMapping, and @DeleteMapping, are used to map the methods to specific HTTP requests and URLs.

--------------------------------------------------------------------------------------------------------------------------------------