**6. Spring Boot**

1. What is an API?

**API stands for Application Programming Interface, which allows software programs to communicate with each other. It defines the way for requesting and receiving information or data from one software to another. An API acts as an interface between two applications and enables them to share data and functionality, allowing for the creation of web-based and mobile applications that can interact with other systems and services.**

1. What are http methods?

**HTTP methods are used to specify the intended operation to be performed on a specific resource. The most commonly used HTTP methods are:**

**GET**: retrieves information from the specified resource.

**POST**: submits data to the specified resource for further processing.

**PUT**: updates an existing resource on the server.

**DELETE**: deletes the specified resource from the server.

**HEAD**: retrieves header information from the specified resource without returning the body of the response.

**PATCH**: updates a portion of the specified resource on the server.

**OPTIONS**: returns the allowed HTTP methods for a specified resource

1. Create a spring boot application

In a Java Spring Boot project, the controller, service, and repository classes play different roles in the application's architecture.

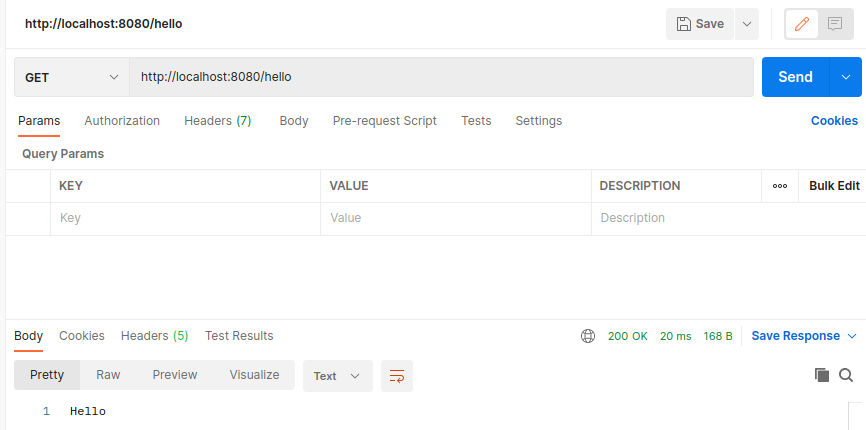
**Controller**: A controller is a component that handles HTTP requests. It processes user input and returns the appropriate response, typically an HTML page, but can also return data such as JSON or XML. The controller acts as a mediator between the user interface and the service layer.

**Service**: A service is a component that contains business logic. It communicates with the repository to perform operations on the data, such as retrieving or storing information. The service layer abstracts the underlying data storage and is responsible for handling transactions.

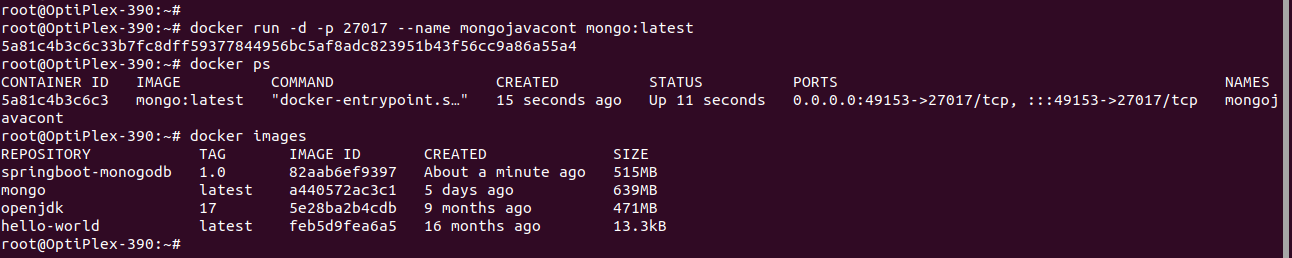
**Repository**: A repository is a component that provides an abstraction over the data storage, such as a database. The repository communicates with the database to perform CRUD operations, such as creating, retrieving, updating, or deleting data. It allows the service layer to perform these operations without having to know the underlying database technology.

A repository acts as a bridge between the application and the data source, while the **model** defines the structure of the data and the behavior of the application. The repository is used to access and manipulate the data, while the model is used to represent the data and the behavior of the application.

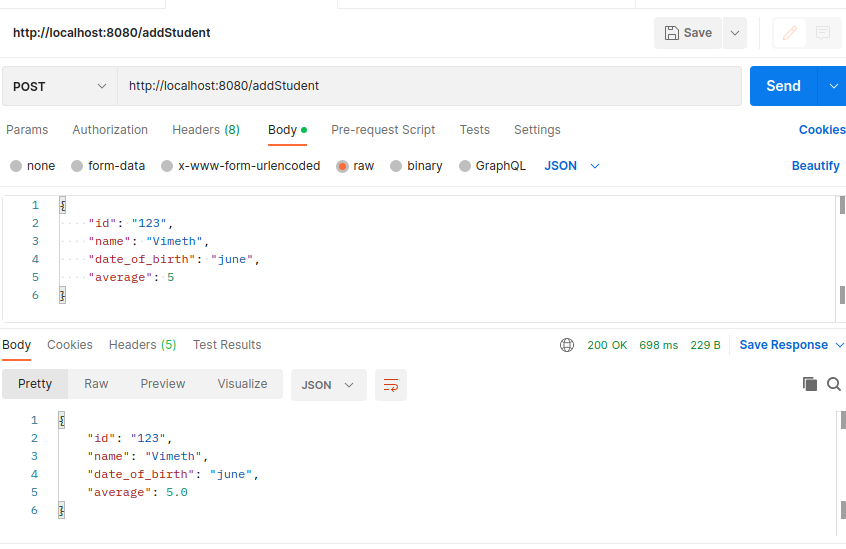
1. Create a new controller class
2. Add a new end-point to return and string
3. Test the endpoint with postman



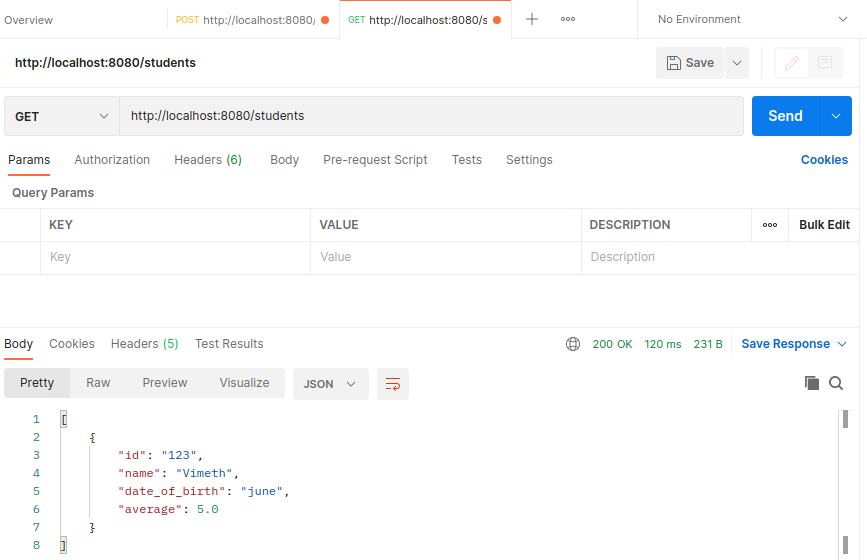
1. Create a Student class (attributes: id, name, date of birth, average)
2. Create a new controller class for students
3. Create a new service class for students
4. Add an endpoint to get list of students
5. Add an endpoint to get a student with id
6. Add new endpoint to create a student
7. Run mongodb as a docker container



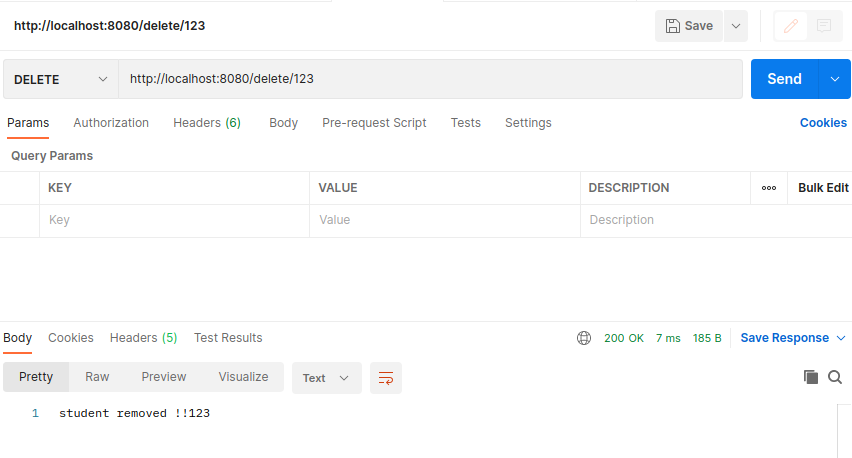
1. Insert student received to endpoint created in step 12 to database



1. Extend step 10 and 11 to query data from database



1. Add an endpoint to delete a student with id.
2. Delete the student with id from database



1. What are http status codes

**HTTP status codes are 3-digit codes returned by a server to a client after receiving an HTTP request. They indicate the result of the request, whether it was successful or encountered an error.**

1. Briefly explain the meaning of following status codes

200, 201, 301, 400, 401, 403, 404, 405, 500, 501, 502, 503, 504

**200 OK**: The request was successful.

**201 Created**: The request resulted in a new resource being created.

**301 Moved Permanently:** means that the requested resource has been permanently moved to a new URL and any future requests for that resource should use the new URL

**400 Bad Request**: The request could not be understood or was missing required parameters.

**401 Unauthorized**: The request requires user authentication.

**403 Forbidden**: The server understood the request, but it refuses to authorize it.

**404 Not Found**: The requested resource could not be found.

**500 Internal Server Error**: An error occurred on the server

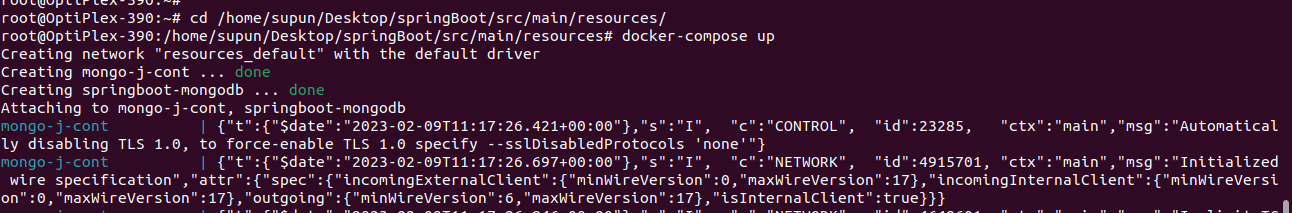
**501 Not Implemented:** means that the server does not support the functionality required to fulfill the request.

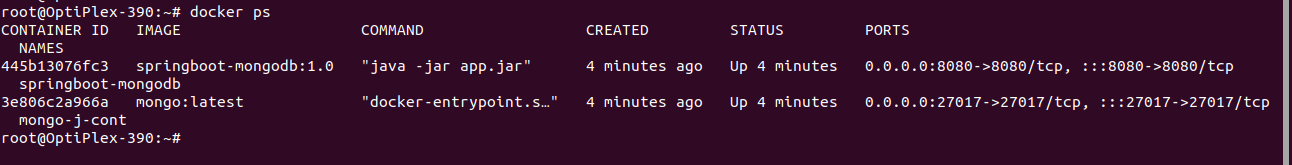
**502 Bad Gateway**: means that the server received an invalid response from an upstream server while attempting to fulfill the request.

**503 Service Unavailable**: means that the server is temporarily unable to handle the request due to maintenance or overloading.

**504 Gateway Timeout**: means that the server did not receive a timely response from an upstream server while attempting to fulfill the request.

1. Using docker-compose run spring boot application and mongodb





1. Create new branch “spring-boot-app-v1” and push the project you created
2. Add your codes and answer sheet to a directory named “spring-boot-basic-training-v1” and push it to your training github repository
3. Create a pull request to main branch and assign it to your trainer

Additional:

In Spring Framework, annotations are used to provide metadata about the components in your application, such as classes, methods, and properties. The following are some commonly used annotations in Spring:

1. @Component: Used to mark a class as a component and make it eligible for auto-detection through component scanning.
2. @Controller: Used to mark a class as a web controller, responsible for handling incoming HTTP requests.
3. @Service: Used to mark a class as a service component, representing a business service in the application.
4. @Repository: Used to mark a class as a data access component, responsible for managing the persistence of data.
5. @Autowired: Used to specify a constructor, setter method, or field that should be wired with a bean from the application context.
6. @Value: Used to set a property value directly, typically from a configuration file.
7. @RequestMapping: Used to map a URL pattern to a specific method in a controller, allowing it to handle HTTP requests.
8. @ResponseBody: Used to specify that a method return value should be bound to the response body of an HTTP response.
9. @RestController: A combination of @Controller and @ResponseBody, used to create a RESTful web service controller.

If you want to create entity, controller, repository, and service classes, the general order in which you should create these classes is:

1. Entity class: Start by creating the entity class, which represents the data model for your application. This class should define the attributes, relationships, and constraints of the data you want to store.
2. Repository interface: Next, create the repository interface, which defines the methods you will use to interact with your data store. This interface should extend **JpaRepository** or another appropriate repository interface, and should declare the methods you need to perform CRUD operations on your entity.
3. Service class: After the repository interface, create the service class, which implements the business logic for your application. The service class should depend on the repository interface, and should use the methods declared by the repository to perform operations on your entity.
4. Controller class: Finally, create the controller class, which handles incoming HTTP requests and returns the appropriate responses. The controller should depend on the service class, and should use the methods provided by the service to handle incoming requests.

This order of creating classes allows you to first define the data model for your application, then the way you want to interact with that data, then the business logic that should be performed on the data, and finally the way the data should be exposed to the outside world through HTTP endpoints.

Once you have created all the classes, you can then test and refine your implementation as needed.