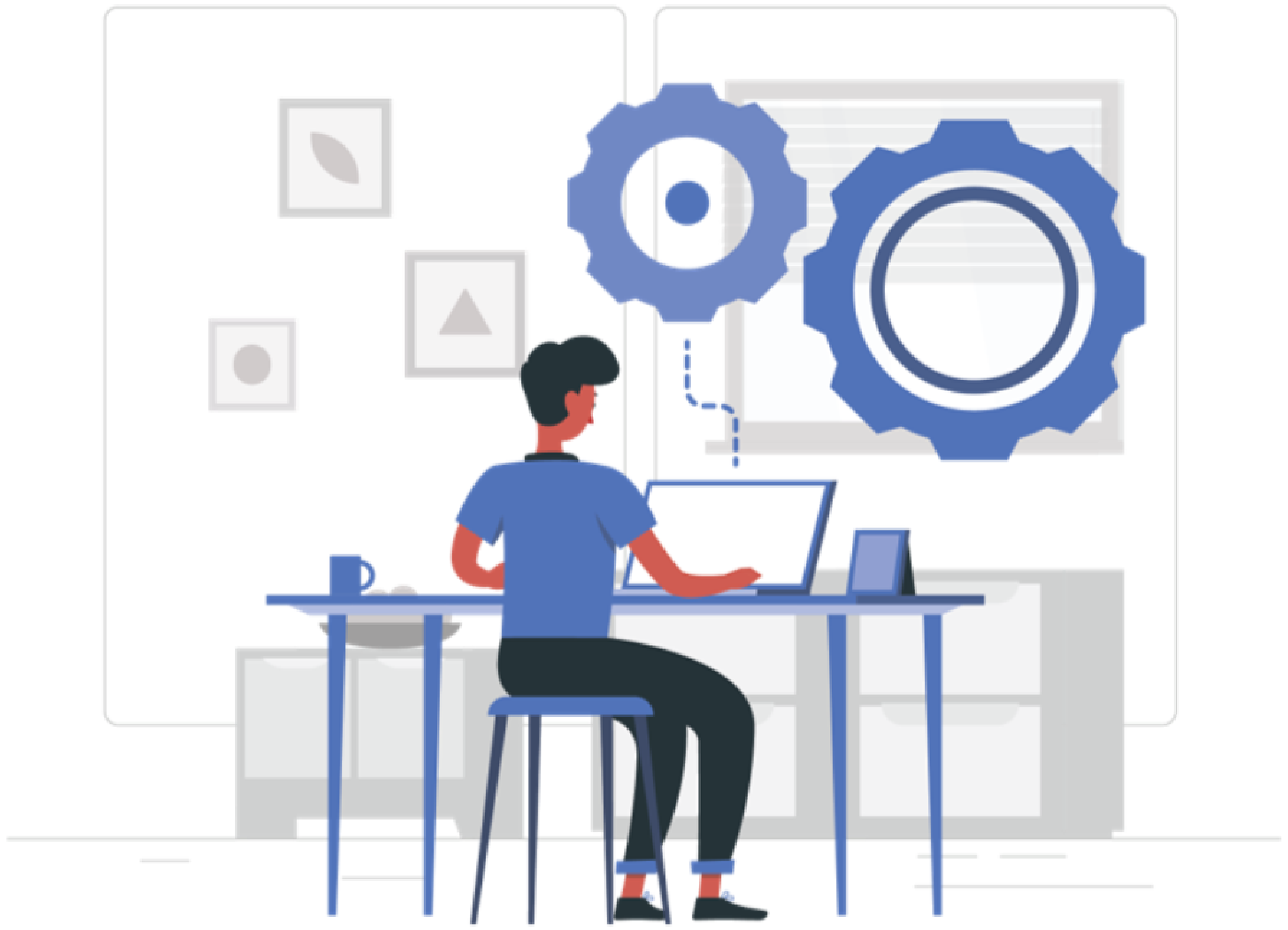


Challenge

Develop RESTful Services by Using Spring Boot by Using JPA

Product Service

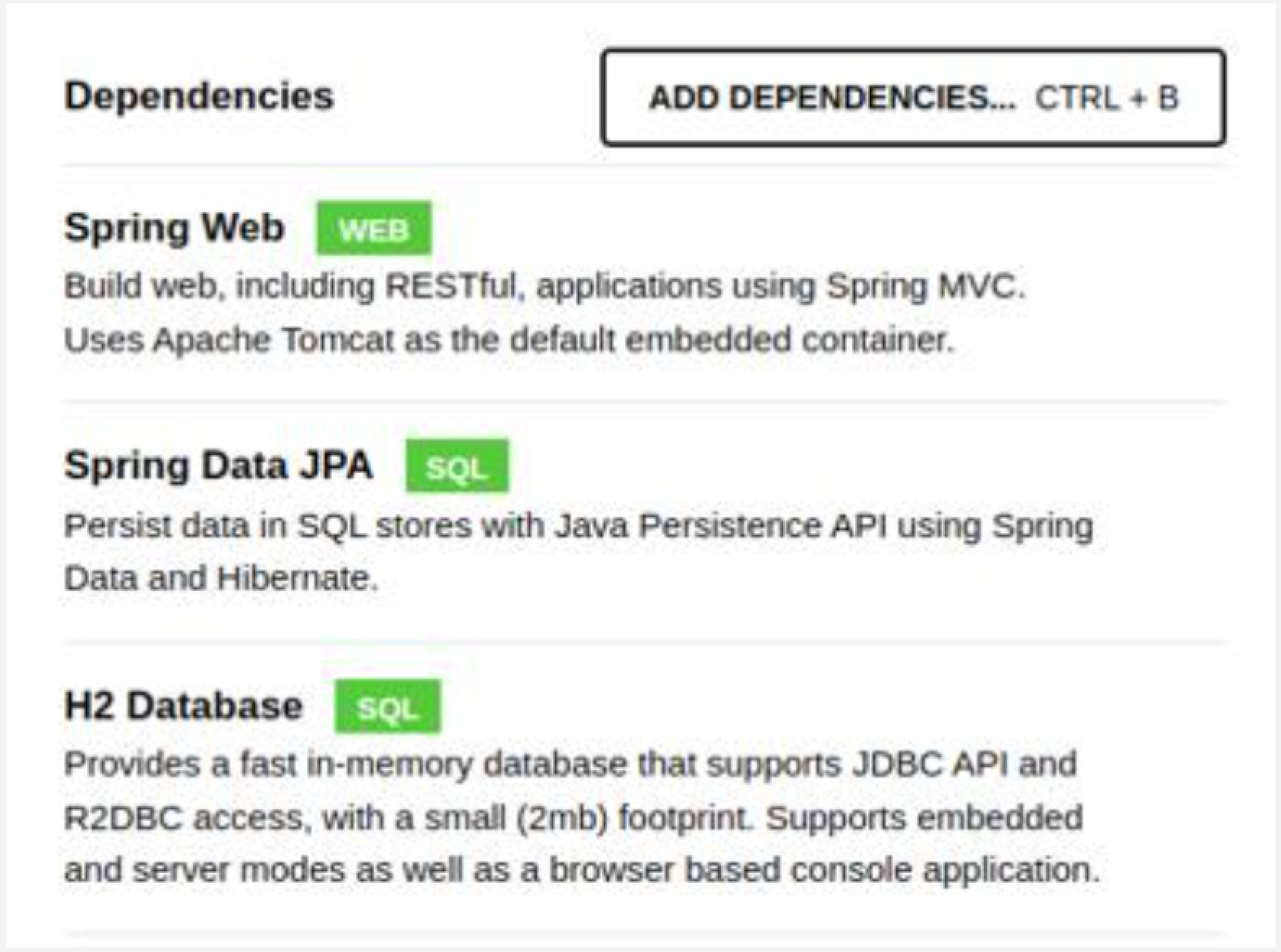
Create a Spring Boot application with one domain class called Product and provide the Service, Controller, and Repository layers.

Save all the Product objects inside the H2 database. Also, retrieve all the Product objects from the H2 database. Test all the REST endpoints on the Postman.

CHALLENGE

Implementation Environment

- Create a Spring Boot application from the Spring Initializr.
- Add the necessary dependencies to pom.xml
- Download the project to your local machine.
- Extract the zip file.
- Export the project in your local IDE.



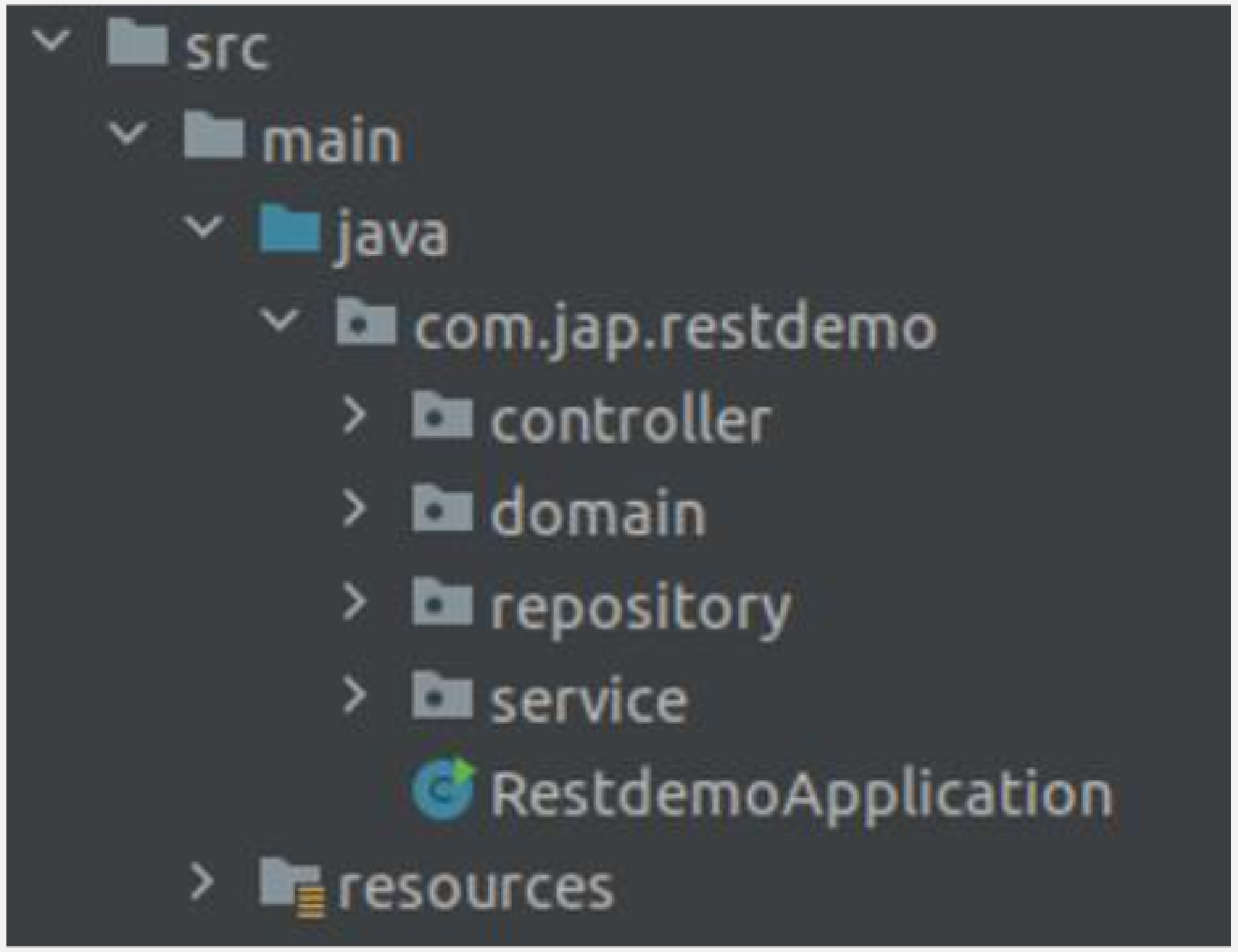
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

H2 Database SQL
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Challenge: Task 1

- The structure of the project is given for your reference.
- Create the domain class `Product` with the following attributes: `productName`, `productId`, `manufacturer`.
- Annotate the `Product` domain class with @`Entity` and `productId` with `@Id`.
- Create getters and setters for all the attributes.
- Create the Repository Interface that extends `CrudRepository` past two parameters: first, the entity class name, and second, the data type of the @Id attribute.



```
src
  main
    java
      com.jap.restdemo
        controller
        domain
        repository
        service
        RestdemoApplication
    resources
```

Challenge: Task 2

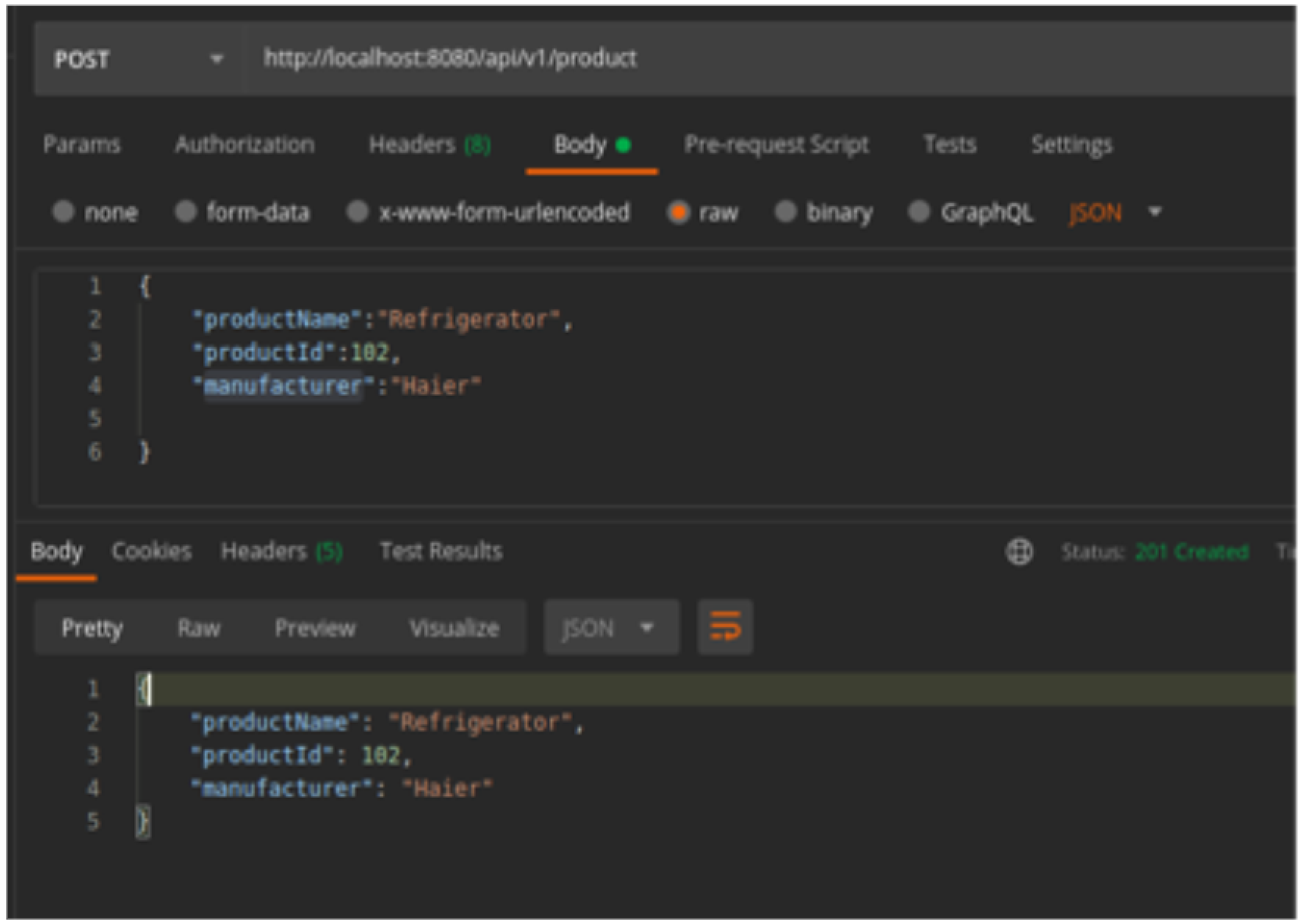
- Inside the Service package, create two Java files: `ProductService` Interface and `ProductServiceImpl` class.
- Annotate the `ProductServiceImpl` class with @Service.
- Inside the `ProductService` Interface, create the method to save the product object and get all the product objects.
- Implement this interface inside the `ProductServiceImpl` class and override the methods.
- Autowire `ProductRepository` inside the service layer.
- Call the `ProductRepository` save() method inside the service class to save the product object in the H2 database.
- Call the `ProductRepository` findAll() method inside the service class to get all the product objects from the H2 database.

Challenge: Task 3

- Inside the Controller package, create the `ProductController` class.
- Annotate this class with @`RestController` and @`RequestMapping`
- Autowire `ProductService` inside the controller layer.
- Create the handler method to save the product data by calling the service save method.
- Annotate this handler method with @`PostMapping`.
- Create the handler method to get all the product data from the service method by calling getAllProducts().
- Annotate this handler method with @`GetMapping`.
- Set up the H2 database configuration details in the `application.properties` file.
- Run the boot application by using the Spring way of execution.
- Open Postman and call the REST API.
- Open the H2 console and check that the tables are created.

Postman Output

Post the Product Data



POST http://localhost:8080/api/v1/product

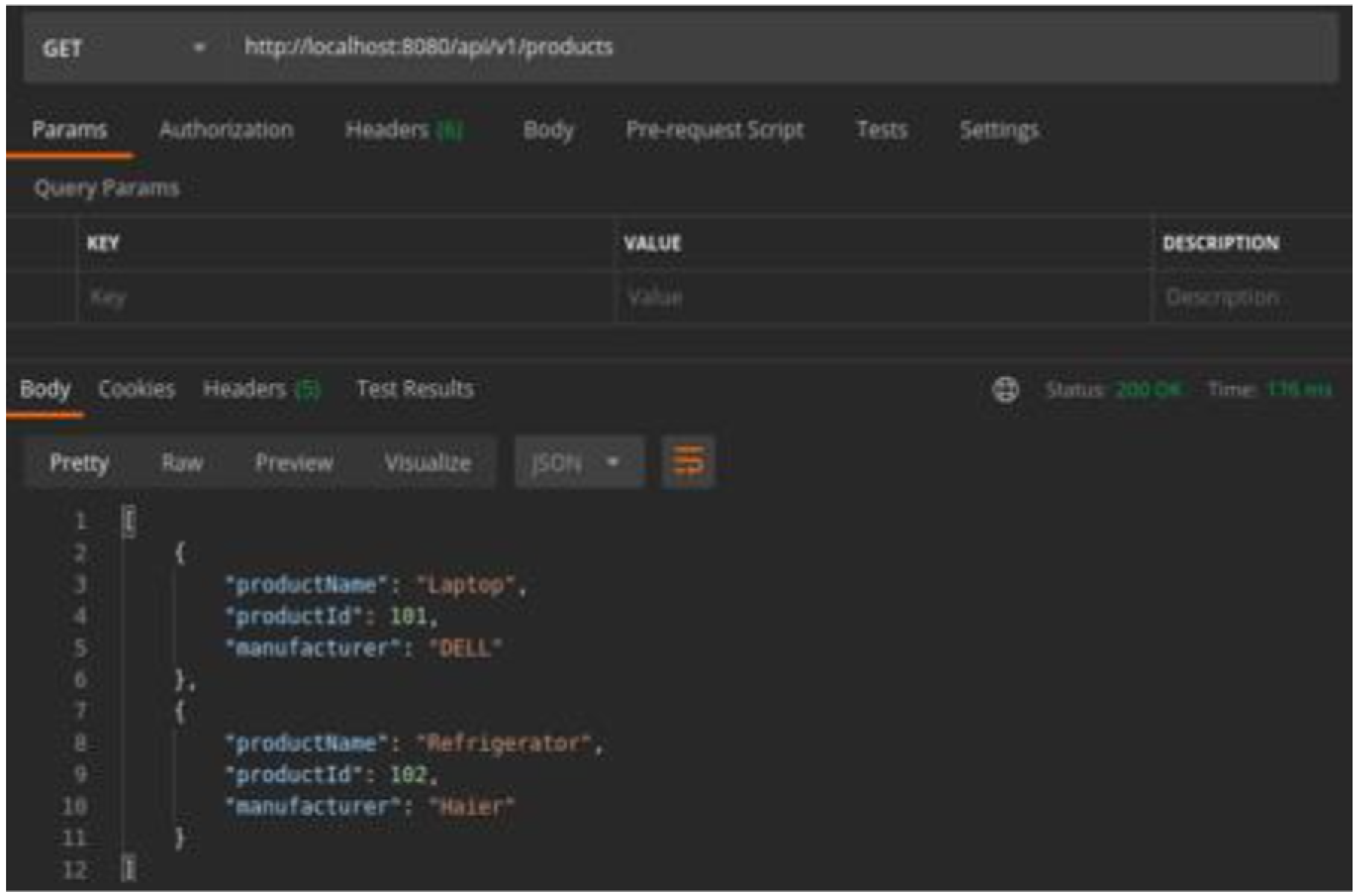
```
{
    "productName":"Refrigerator",
    "productId":102,
    "manufacturer":"Haier"

}
```

Status: 201 Created

```
{
    "productName": "Refrigerator",
    "productId": 102,
    "manufacturer": "Haier"
}
```

Get the Product Data



GET http://localhost:8080/api/v1/products

Status: 200 OK

```
[
    {
        "productName": "Laptop",
        "productId": 101,
        "manufacturer": "DELL"
    },
    {
        "productName": "Refrigerator",
        "productId": 102,
        "manufacturer": "Haier"
    }
]
```

Submission Instructions

- There is no boilerplate for the practice.
- Create a Git repository named **BEJ_C1_S5_REST_API_MC_1.**
- After completing the challenge, push the code back to git using the below commands.

```
git init
git remote add origin <url>
git add .
git commit –m “comments on the push”
git push –u origin master
```

- Submit it for review.