

Requirement: To create RESTful Web services implementation using Java, Spring Boot, JPA, Hibernate and MySQL as back-end database.

Create the CRUD operation methods to perform actions like Save new Employee, Read existing Employees, Update existing Employee Information and finally Delete an existing Employee

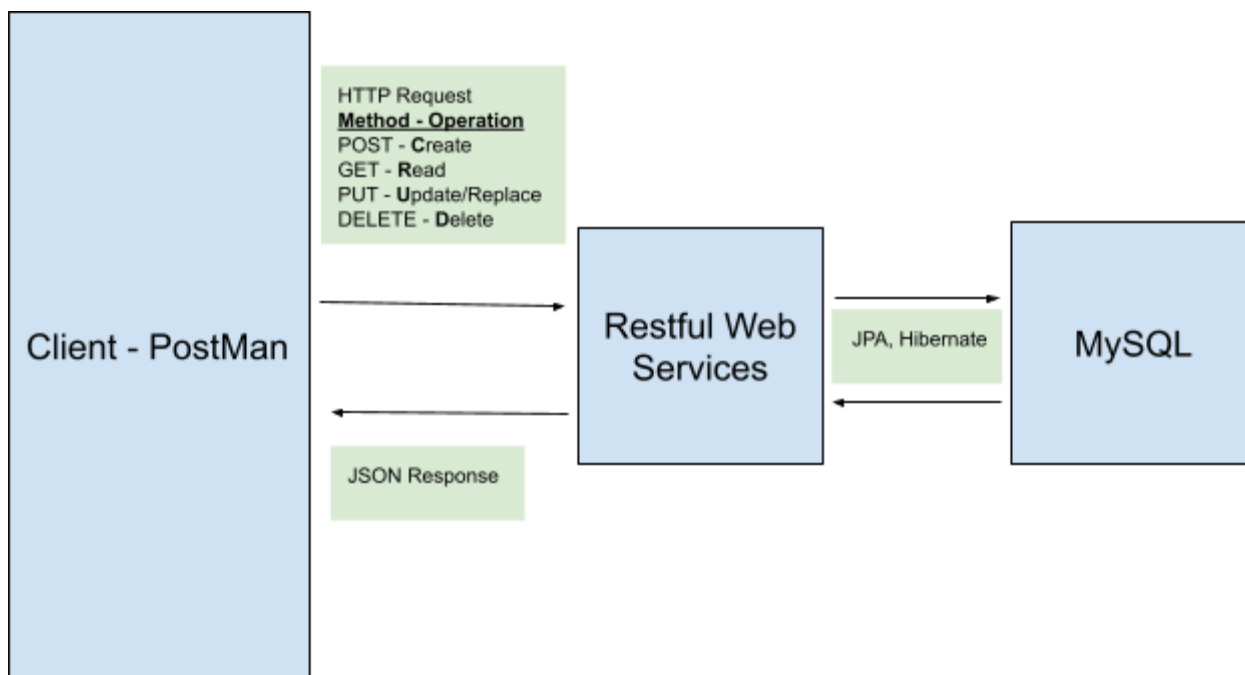
Solution Specifications

For the POC solution to be accessible over the internet, both the application and the database should be hosted on a cloud environment. I have preferred to use **Amazon AWS** to get the benefits of **Elastic Beanstalk** for hosting the Spring boot application as JAR and the **AWS RDS** to host the MySQL database. The AWS Free Trial tier is being used to avoid any costs for hosting the POC solution.

Following are the components and the steps required to implement the solution.

Step 1

Solution Definition



Step 2

Create a **MySQL** database on AWS RDS

- DB identifier : ilinesoft-database-1
- Database: EmployeeDb
- Table : employee
- Endpoint - ilinesoft-database-1.c0vklmcetpe.us-east-2.rds.amazonaws.com/

Add sample data manually using SQLStatement or MySql Client (HeidiSQL)

Host: ilinesoft-database-1.c0vkclmctpe.us-east-2.rds.amazonaws.com				Database: EmployeeDb		Table: employee	
Data		Query*			Query #2*		
EmployeeDb.employee: 3 rows total (approximately)				Next		Show all	
				Sorting		Columns (8/8) Filter	
id	first_name	last_name	work_location_city	technology	department	salary	date_of_joining
1,001	Ahmed	Shahab	Houston	Java, Spring Boot, MySql	Technology	40,000.67	2020-02-10
1,002	Raghu	Kalyan	Banglore	Java, Spring Boot, Angular, AWS	Technology	50,000.9	2020-02-26
1,003	Mohammed	Salman	Hyderabad	N/A	Human Resource	6,000,190	2014-01-25

Step 3

Use **Spring Initializer** to generate spring boot projects by selecting Maven Project, Java, Spring Boot 2.2.4. Enter Project Metadata Group as *com.ilinesoft.springboot.employee* and Artifact *employee.poc*

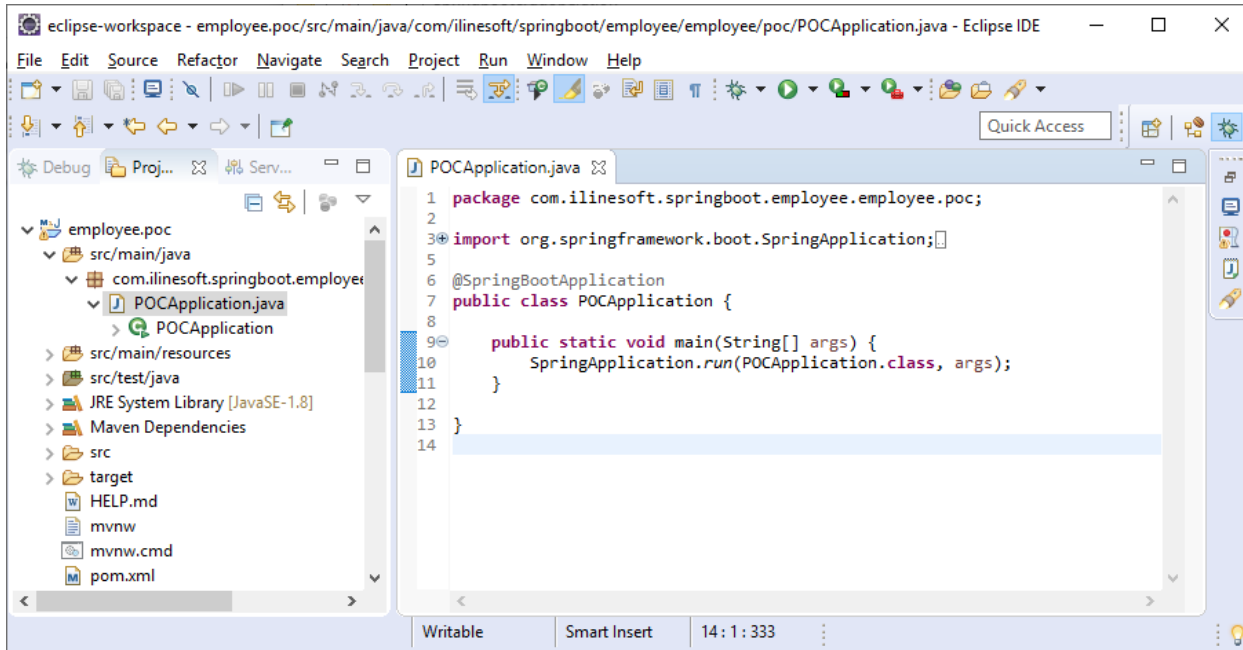
The screenshot shows the Spring Initializer web application interface. The left sidebar contains navigation links: Project, Language, Spring Boot, Project Metadata, and Dependencies. The main content area is configured as follows:

- Project:** Maven Project (selected), Gradle Project
- Language:** Java (selected), Kotlin, Groovy
- Spring Boot:** 2.3.0 M1, 2.3.0 (SNAPSHOT), 2.2.5 (SNAPSHOT), **2.2.4** (selected), 2.1.13 (SNAPSHOT), 2.1.12
- Project Metadata:**
 - Group: *com.ilinesoft.springboot.employee*
 - Artifact: *employee.poc*
 - Options: (empty)
- Dependencies:**
 - Search dependencies to add: Web, Security, JPA, Actuator, Devtools...
 - Selected dependencies: **Spring Web** (Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container. [checked])

At the bottom, there are buttons for "Generate - Ctrl + G", "Explore - Ctrl + Space", and "Share...". The footer mentions "© 2013-2020 Pivotal Software" and "start.spring.io is powered by Spring Initializer and Pivotal Web Services".

Remember to select the dependency **Spring Web**, **Spring Data JPA** and **MySQL Driver**

Download the project and open in Eclipse IDE or IntelliJ IDEA Community Edition. I am comfortable using **Eclipse IDE**



The Main class already exists as part of the Spring Initializer

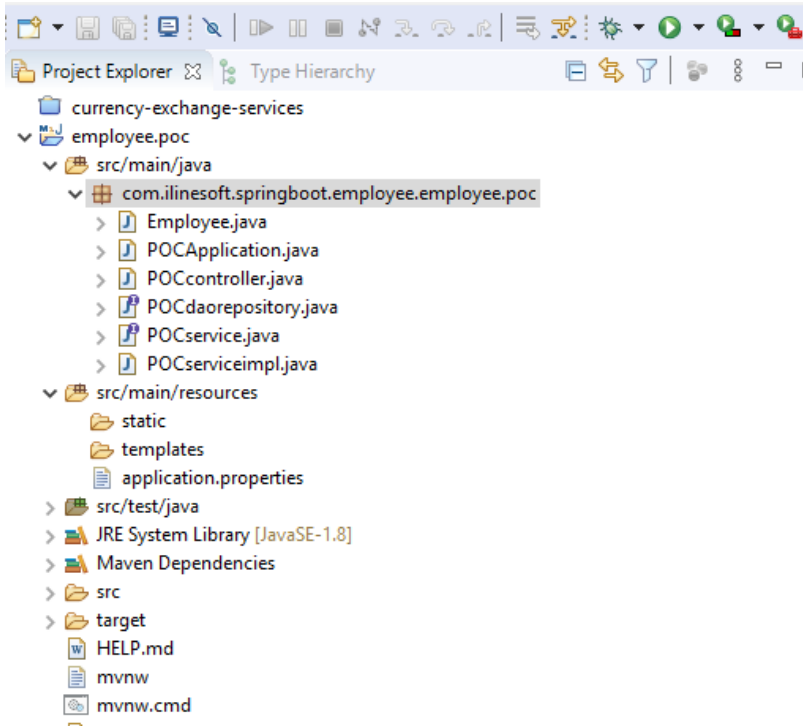
Now create the below .java class files in the sequence order as shown below

- a) Modal **Class** - Employee.java
- b) Data-Access-Object **interface** - POCdaorepository.java
- c) Service **interface** - POCservice.java
- d) Service **Class** - POCserviceimpl.java
- e) Controller **class** - POCcontroller.java

Class designed to handle incoming HTTP requests

Class is annotated with @RestController

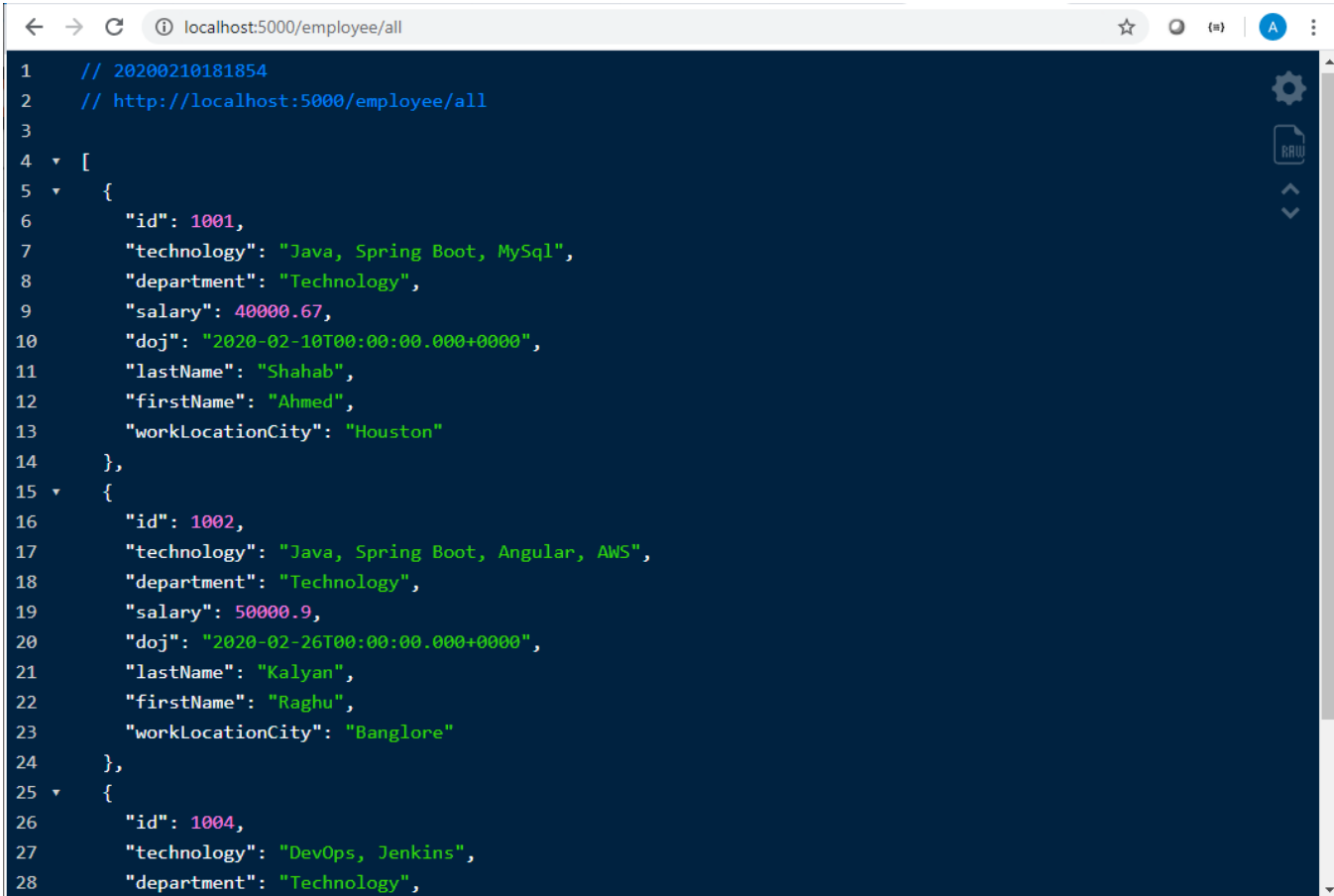
Below screen shows files in the Project Explorer once the .java files are created



Step 4

Run the project using eclipse locally to test POC application is launching successfully by using a browser and typing localhost and port 5000

Note: We will test the remaining Rest API methods using postman tool in the subsequent steps after the application is deployed on AWS



The screenshot shows a web browser window with the address bar displaying `localhost:5000/employee/all`. The main content area shows a REST client interface with a JSON array of employee data. The JSON is as follows:

```
1 // 20200210181854
2 // http://localhost:5000/employee/all
3
4 [
5   {
6     "id": 1001,
7     "technology": "Java, Spring Boot, MySql",
8     "department": "Technology",
9     "salary": 40000.67,
10    "doj": "2020-02-10T00:00:00.000+0000",
11    "lastName": "Shahab",
12    "firstName": "Ahmed",
13    "workLocationCity": "Houston"
14  },
15  {
16    "id": 1002,
17    "technology": "Java, Spring Boot, Angular, AWS",
18    "department": "Technology",
19    "salary": 50000.9,
20    "doj": "2020-02-26T00:00:00.000+0000",
21    "lastName": "Kalyan",
22    "firstName": "Raghu",
23    "workLocationCity": "Bangalore"
24  },
25  {
26    "id": 1004,
27    "technology": "DevOps, Jenkins",
28    "department": "Technology",
```

Step 5

Export the project as JAR file and upload on AWS Elastic Beanstalk

The screenshot shows the AWS Elastic Beanstalk console. The browser tabs include 'RDS - AWS', 'rest-api-ii', 'restapiiilinesoft', 'Search re...', 'POC-REST', and 'Customer'. The address bar shows 'us-east-2.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-2#/application/over...'. The console header shows 'aws', 'Services', 'Resource Groups', and a user profile 'MyAWSLearningAC' from 'Ohio'. The main navigation bar has 'Elastic Beanstalk' and 'rest-api-iiinesoft-poc' selected, with a 'Create New Application' button. The breadcrumb is 'All Applications > rest-api-iiinesoft-poc'. On the left, a sidebar lists 'Environments', 'Application versions', and 'Saved configurations'. The main content area shows details for the 'RestApiIlinesoftPoc-env' environment:

- Environment tier: Web Server
- Platform: Java 8 running on 64bit Amazon Linux/2.10.2
- Running versions: rest-api-iiinesoft-poc-v1
- Last modified: 2020-02-10 17:32:23 UTC-0600
- URL: RestApiIlinesoftPoc-env.mid2kpk5nq.us-east-2.elasticbe...
- Health status: Ok

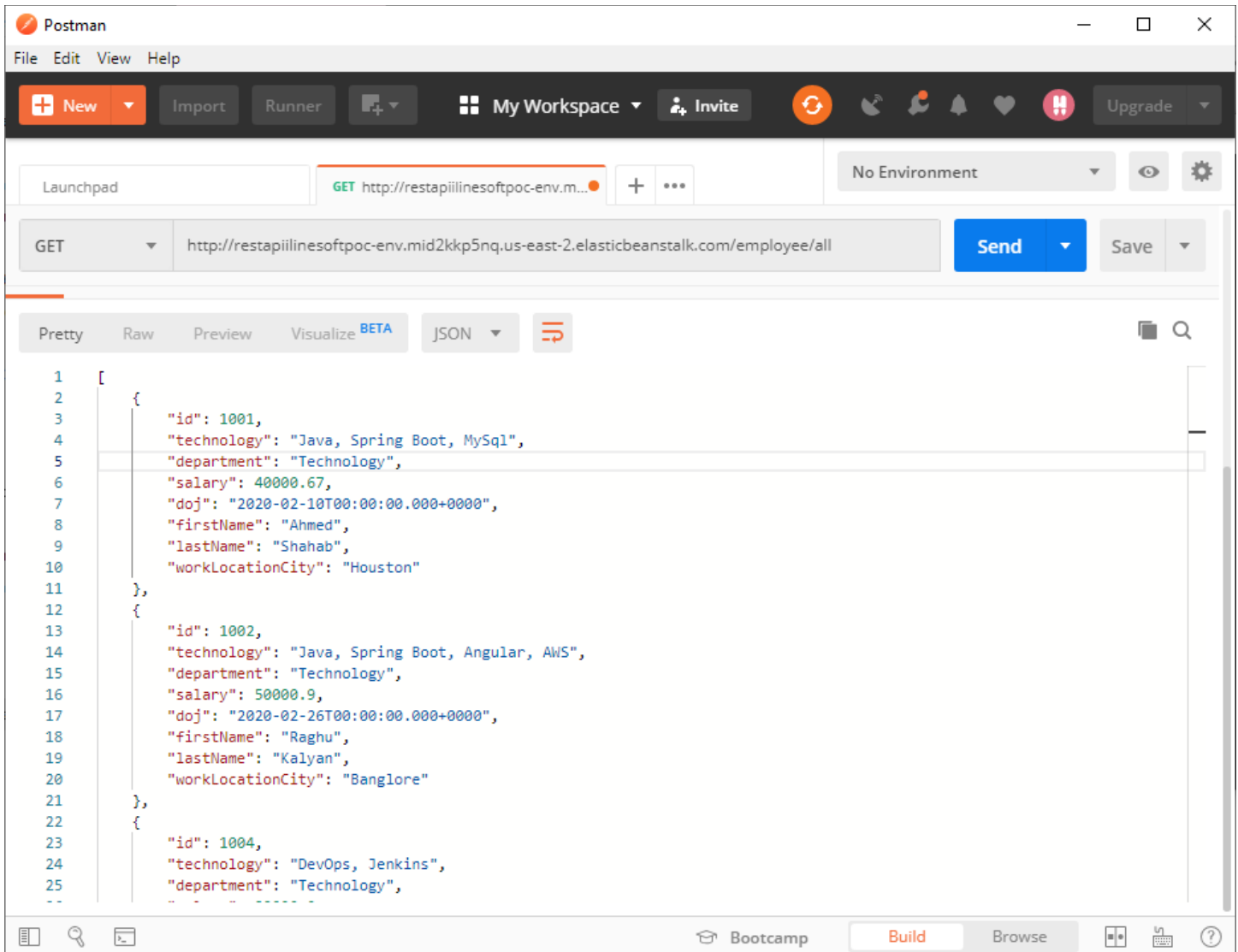
The footer contains 'Feedback', 'English (US)', copyright information '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.', and links for 'Privacy Policy' and 'Terms of Use'.

Step 6

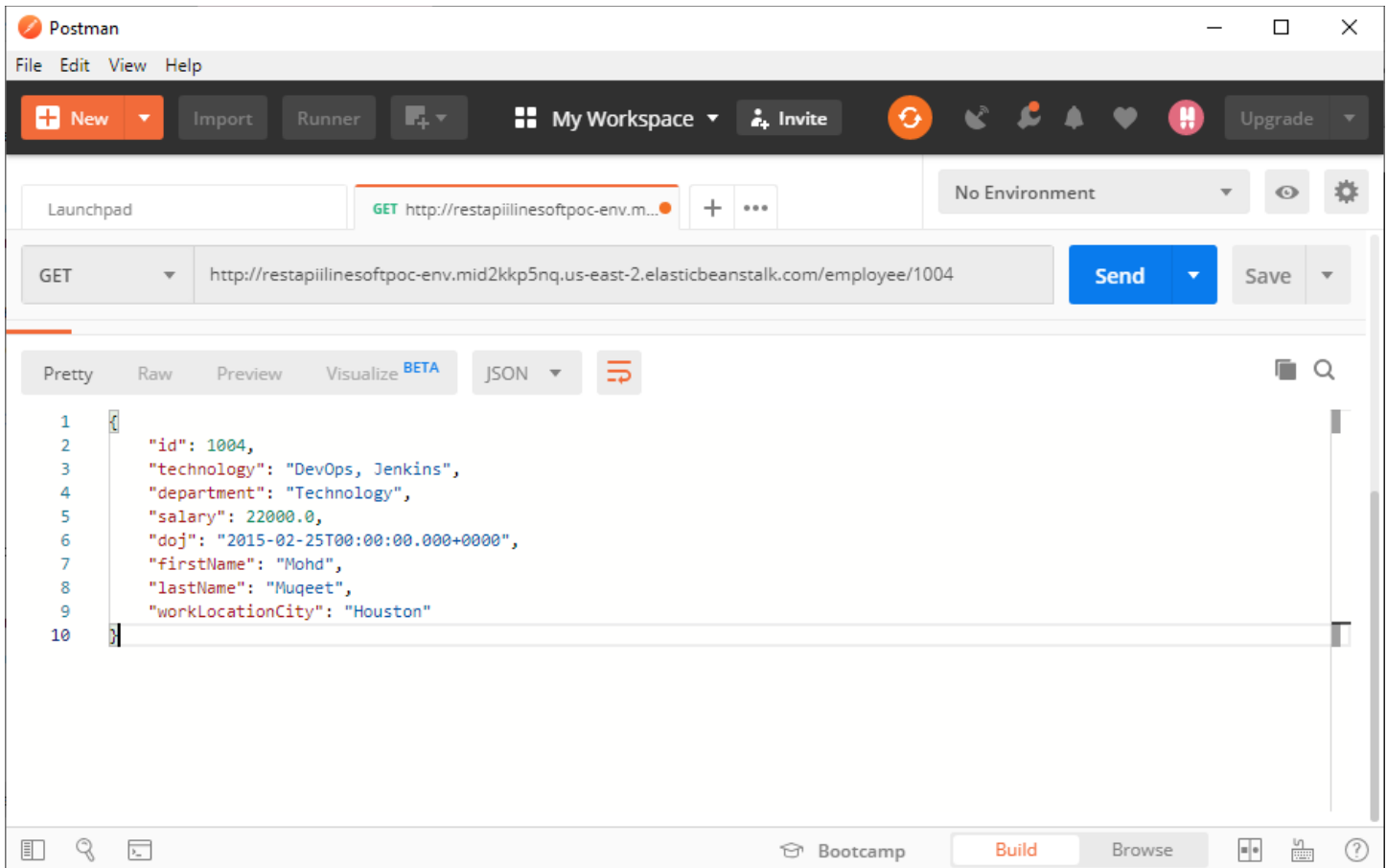
Use the postman tool to test all the CRUD operations.

This step tests the **Read** operation

- Testing the **GET method** to **Read** all employees using the Restful API
- URL `http://restapiiilinesoftpoc-env.mid2kpk5nq.us-east-2.elasticbeanstalk.com/employee/all`
- Method **Get**



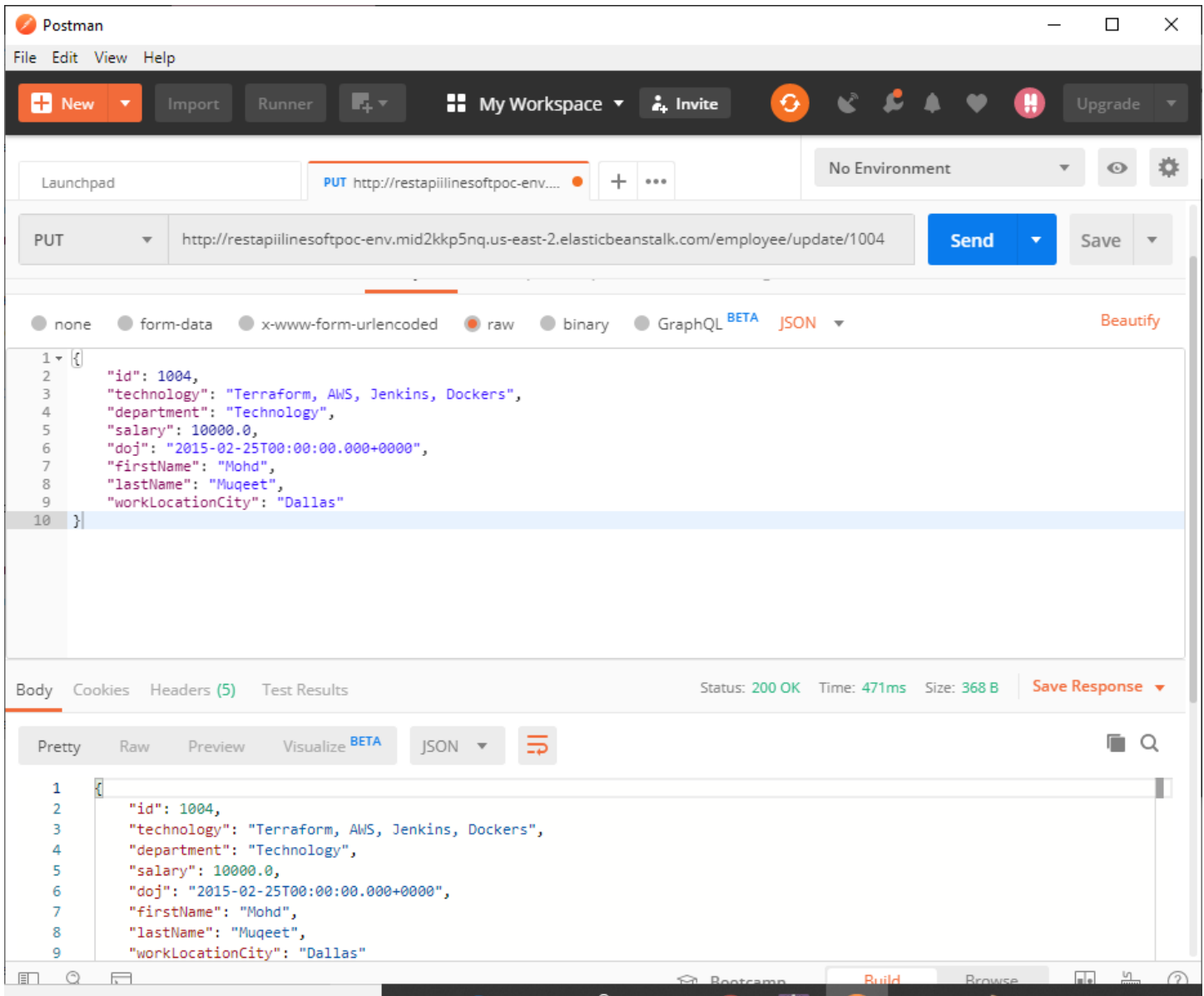
- d) Testing the **GET method** with **parameter** to retrieve specific employee using the Restful API
- e) URL `http://restapiilinesoftpoc-env.mid2knp5nq.us-east-2.elasticbeanstalk.com/employee/1004`
- f) Method Get



Step 7

This step tests the **Update** operation

- g) Testing the **PUT RequestMethod** update specific employees using the Restful API
- h) URL `http://restapiilinesoftpoc-env.mid2knp5nq.us-east-2.elasticbeanstalk.com/employee/update/1004`
- i) Method **PUT**



Step 8

This step tests the **Add** operation

- j) Testing the **POST RequestMethod** add an employee detail using the Restful API
- k) URL <http://restapiilinesoftpoc-env.mid2kkp5nq.us-east-2.elasticbeanstalk.com/employee/update/1004>
- l) Method **Post**

The image shows the Postman application window. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and 'Upgrade'. The main area is divided into sections. The top section shows the request details: 'POST http://restapiilinesoftpoc-env....'. Below this is the 'Untitled Request' section with tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected, showing a JSON payload. The bottom section shows the response details: 'Status: 200 OK', 'Time: 195ms', 'Size: 359 B', and 'Save Response'. The response body is also in JSON format.

Postman

File Edit View Help

New Import Runner My Workspace Invite Upgrade

Launchpad POST http://restapiilinesoftpoc-env.... No Environment

Untitled Request Comments (0)

POST http://restapiilinesoftpoc-env.mid2kpk5nq.us-east-2.elasticbeanstalk.com/employee/add Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON Beautify

```
1 {
2   "id": 1006,
3   "technology": "Java Core, Spring Boot",
4   "department": "Technology",
5   "salary": 2500.00,
6   "doj": "2017-04-28T00:00:00.000+0000",
7   "firstName": "Nisar",
8   "lastName": "Ahmed",
9   "workLocationCity": "St Louis"
10 }
```

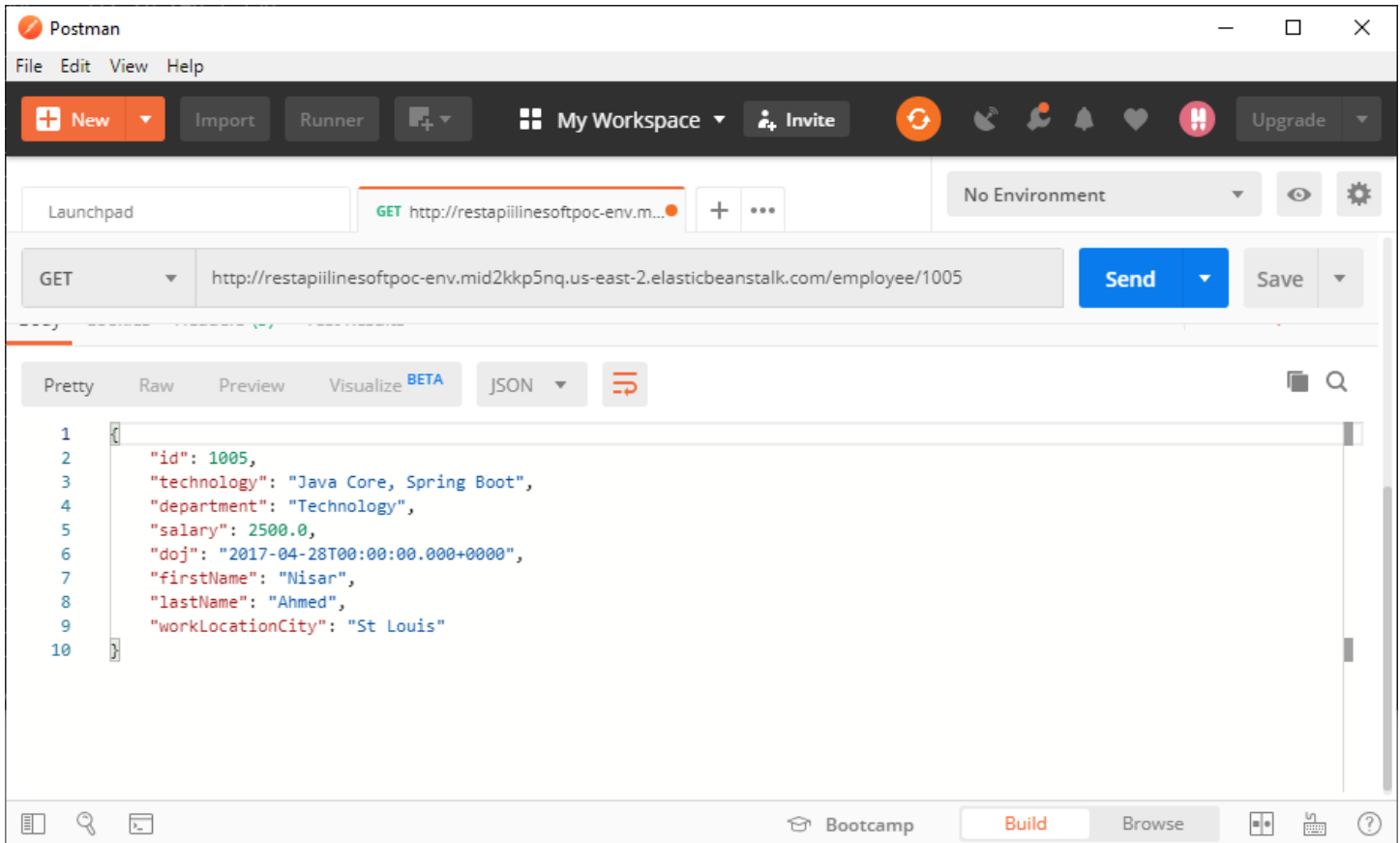
Body Cookies Headers (5) Test Results Status: 200 OK Time: 195ms Size: 359 B Save Response

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "id": 1005,
3   "technology": "Java Core, Spring Boot",
4   "department": "Technology"
```

Bootcamp Build Browse

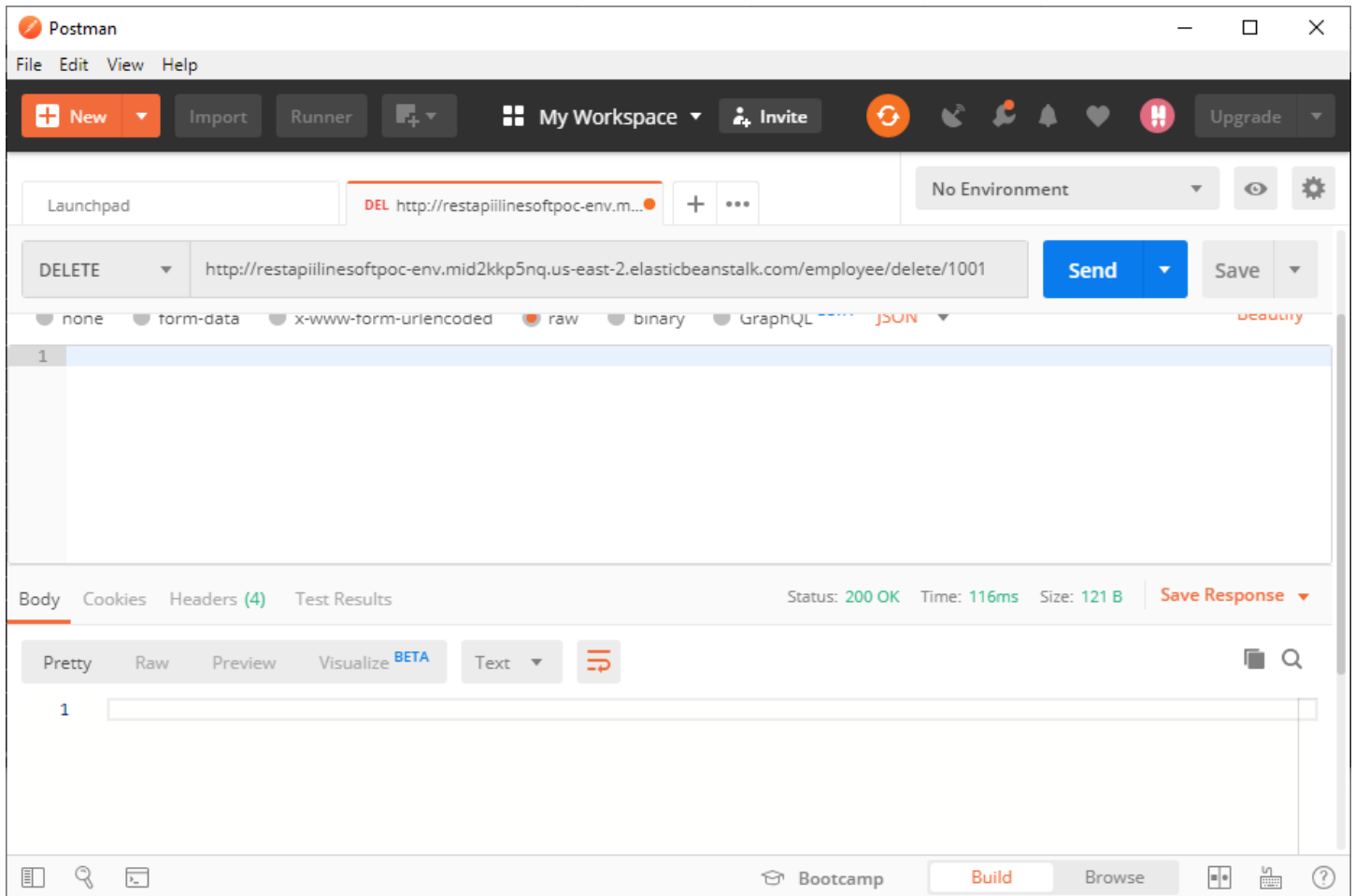
Use the GET method again to see the newly added Employee is pulled from the server



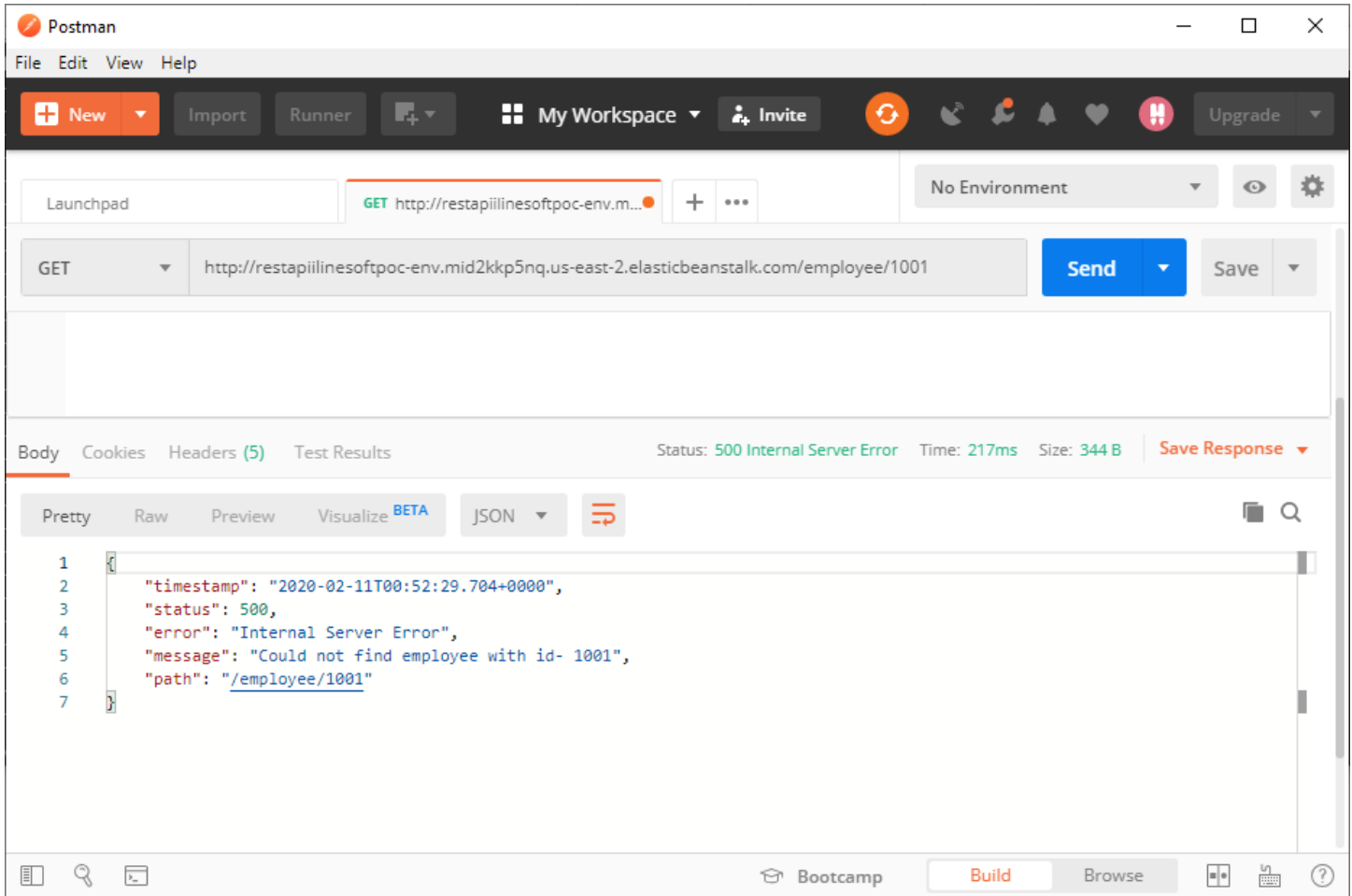
Step 9

This step tests the **Delete** operation

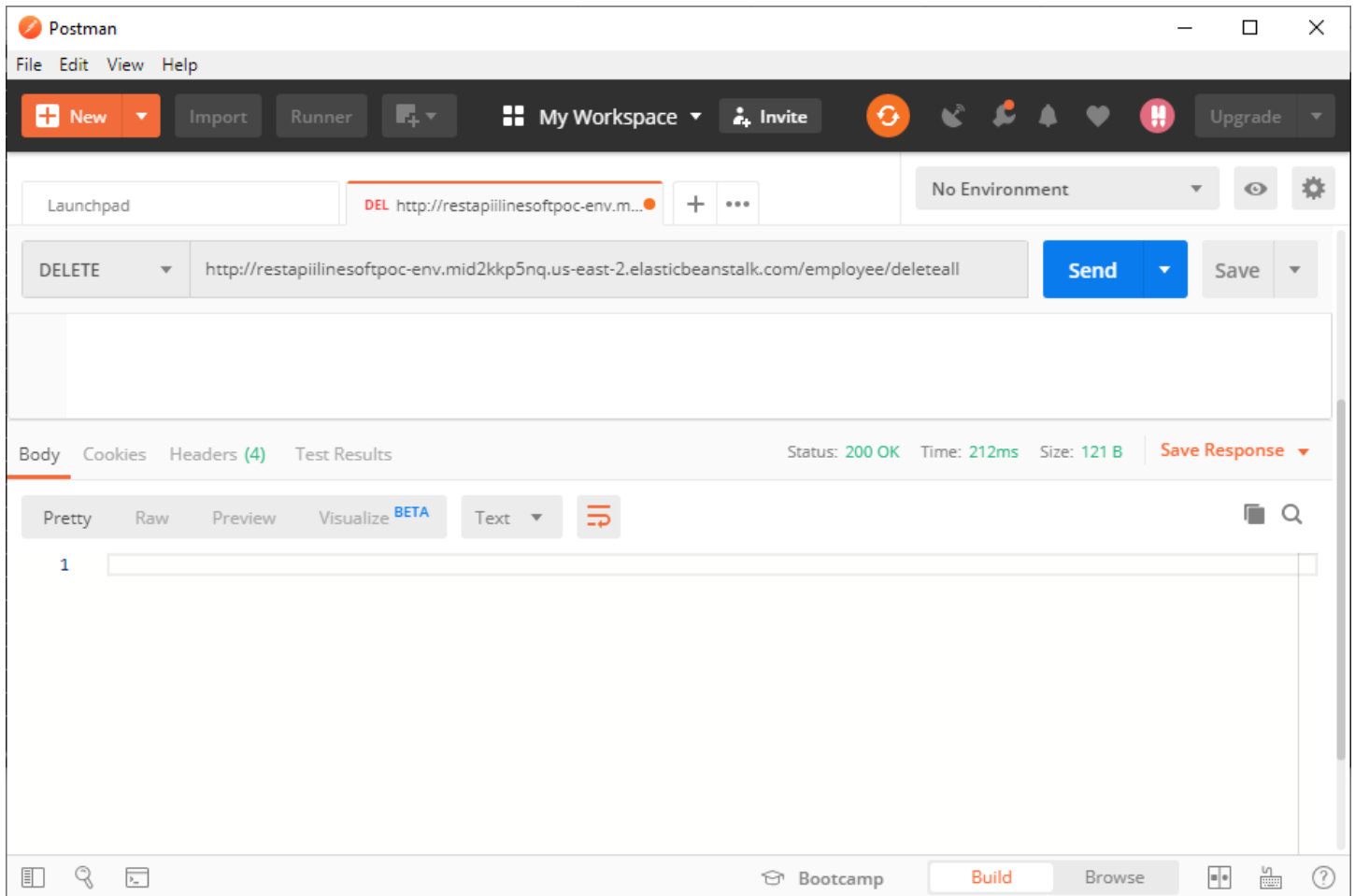
- m) Testing the **DELETE RequestMethod** to delete an specific employee using the Restful API
- n) URL <http://restapiilinesoftpoc-env.mid2kkp5nq.us-east-2.elasticbeanstalk.com/employee/delete/1001>
- o) Method Delete



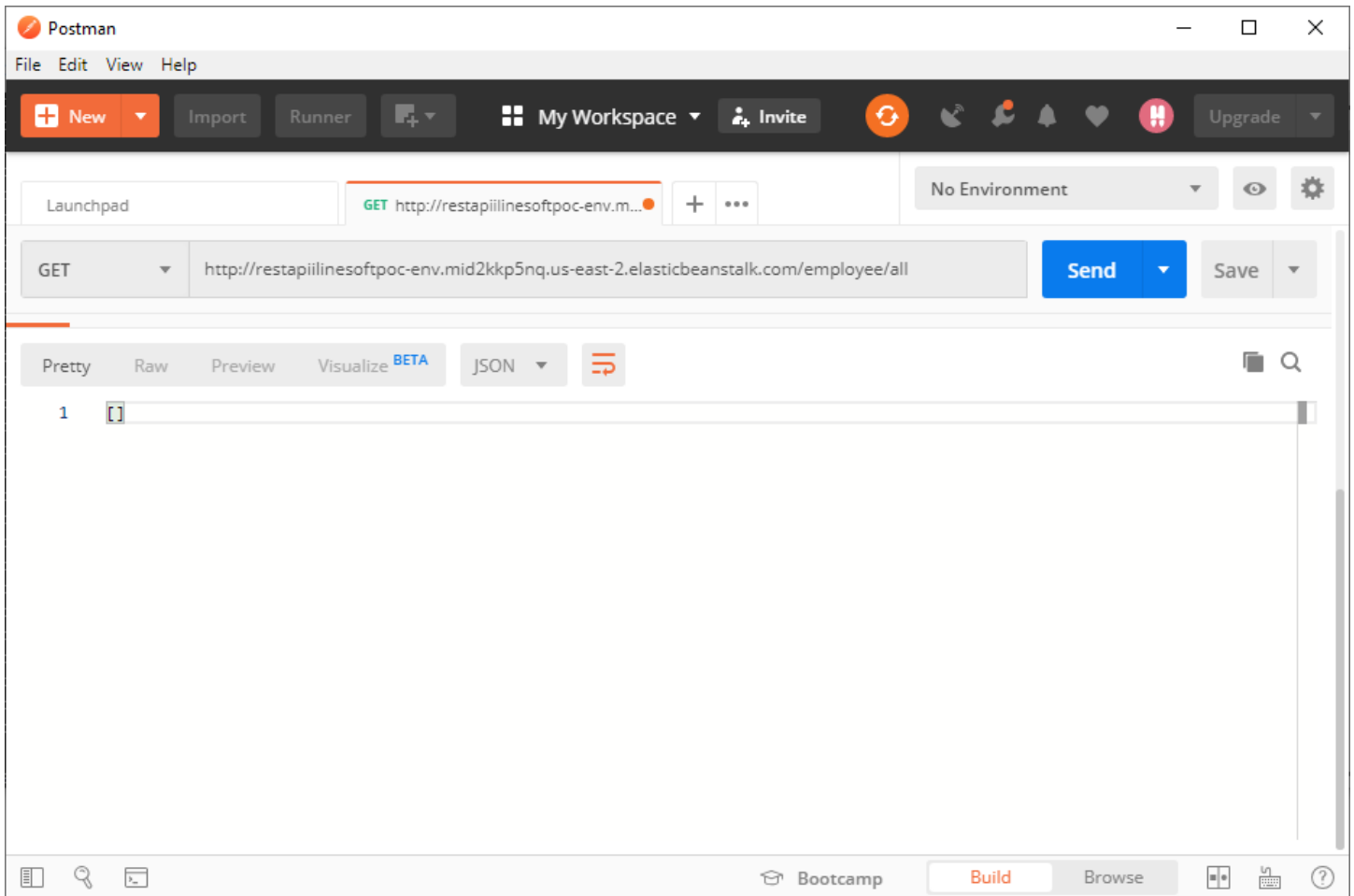
Check if the Employee (1001) has actually been deleted



- p) Testing the **DELETE RequestMethod** to delete **all employees** using the Restful API
- q) URL `http://restapiilinesoftpoc-env.mid2kkp5nq.us-east-2.elasticbeanstalk.com/employee/deleteall`
- r) Method Delete



Executing the GET method again to see any employee record exists.



Conclusion

In the above steps, Restful API CRUD operations have been tested using the Postman tool to demonstrate a Restful application running Java and Spring Boot.

----- End of document-----