EducaCiência FastCode

Fala Galera,

- Artigo: 47/2021 Data: Fevereiro/2021
- Público-alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: Artigo 47 - SpringBoot JPA Mysql Completo
- Link: https://github.com/perucello/DevFP

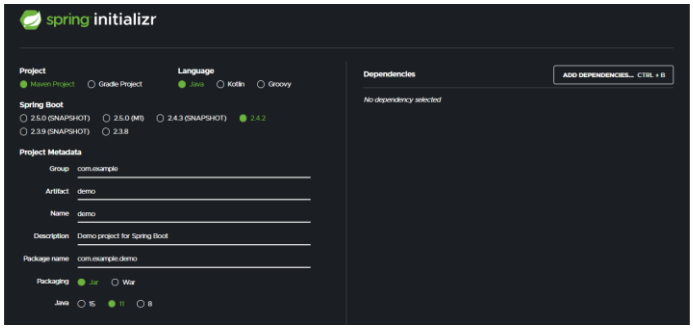
Neste artigo, abordaremos Spring Boot e iremos mapear o CRUD com repositório JPA Repository e Banco de Dados MySql.

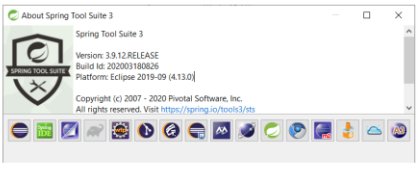
- Post
- Put
- Get e Get ID
- Delete

Lembrando que os fins são didáticos !

Para este ambiente , utilizaremos do Banco de Dados MySql , deixaremos um script de criação do Banco anexado ao nosso artigo.

Criaremos nosso projeto utilizando do link - https://start.spring.io/ e abriremos na IDE Spring Tool Suíte 3 “STS”.





Com nosso Projeto já aberto no STS , iremos preparar nosso arquivo “pom”, ou seja, nossas dependências que iremos trabalhar no projeto.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.4.2</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.project.jpa.mysql</groupId>
        <artifactId>SpringBoot-JPA-Mysql</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>SpringBoot-JPA-Mysql</name>
        <description>Demo project for Spring Boot</description>
        <properties>
                <java.version>1.8</java.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-devtools</artifactId>
                        <scope>runtime</scope>
                        <optional>true</optional>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <scope>runtime</scope>
                </dependency>

                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                </dependency>

                <dependency>
                        <groupId>javax.validation</groupId>
                        <artifactId>validation-api</artifactId>
                        <version>1.1.0.Final</version>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
```

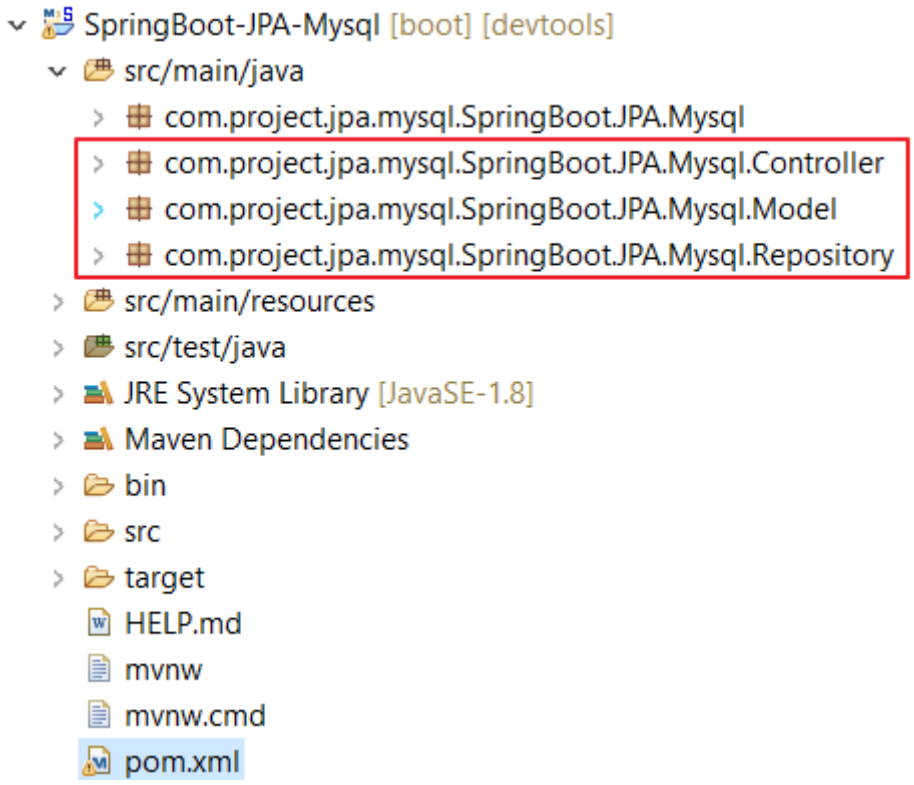
```
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Feito isso, vamos atualizar nosso Maven – para isso, basta executar “Maven Install”.

Vamos preparar agora nossos “pacotes” como demonstrados abaixo:



Feito isso, vamos criar nossas classes sendo:

Vamos detalhar:

a) **Controller** – neste pacote teremos nossa classe "ClientesController" onde será escrito nosso código que manipularemos os métodos CRUD e nosso endpoint;
b) **Model** – neste pacote teremos nossa classe "Cliente" onde criaremos a estrutura da nossa tabela Cliente no Banco de Dados
c) **Repository** – teremos nossa "interface" que se estenderá da classe "Cliente" e receberá nosso Repository JPA.

Primeiramente, vamos criar a estrutura da nossa tabela, como a seguir:

```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model;


import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.NotNull;



@Entity
public class Cliente {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;


	private String nome;

	@NotNull
	private String email;

	public Cliente() {
		super();
	}

	public Cliente(Long id, String nome, String email) {
```

```
		super();
		this.id = id;
		this.nome = nome;
		this.email = email;
	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Cliente other = (Cliente) obj;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (!id.equals(other.id))
			return false;
		return true;
	}

}
```

Agora, podemos estender nossa interface, para isso , abra o pacote Repositório e clique em Clientes.java e insira o seguinte código:

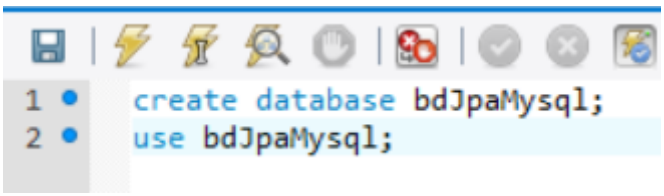
```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model.Cliente;


public interface Clientes extends JpaRepository<Cliente, Long>  {
```

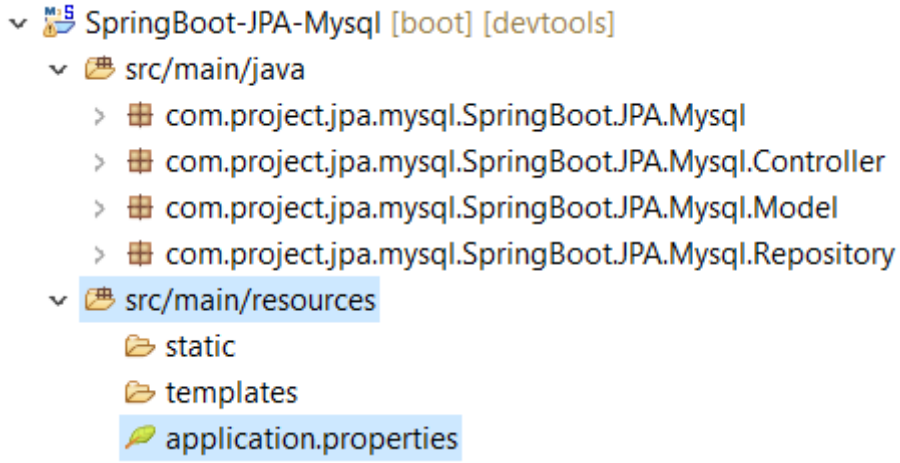
}

Feito isso, temos a estrutura do nosso banco de dados (tabela) pronta e agora , agora precisamos criar nosso Banco de Dados no Mysql, para isso, abra seu Workbench e insira os seguintes códigos e execute.



```
1  create database bdJpaMysql;
2  use bdJpaMysql;
```

Com o Banco de Dados criado, vamos preparar a conexão do nosso Projeto, sendo assim, acesse application.properties pela rota src/main/resources e insira os seguintes códigos:




```
spring.datasource.url=jdbc:mysql://localhost:3306/bdJpaMysql?useTimezone=true&serverTimezone=UTC
&useSSL=false
spring.datasource.username=root
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

Com a nossa estrutura do Banco de Dados criado, (entidade e repositório) agora, podemos focar em nosso CRUD.

Para este trabalho, abra o Script **ClientesController** que está no pacote Controller e insira os seguintes códigos:

Com isso , temos nosso Post do CRUD pronto onde os métodos que manipularemos será o seguinte:

```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Controller;

import java.util.List;
import java.util.Optional;

import javax.validation.Valid;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
```


```
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model.Cliente;
import com.project.jpa.mysql.SpringBoot.JPA.Mysql.Repository.Clientes;


@RestController
@RequestMapping("api/JPA/Mysql/clientes")
public class ClientesController {

	@Autowired
	private Clientes clientes;

	@GetMapping
	public List<Cliente> listar(){
		System.out.println("Quantidade de Registros de Clientes : " + clientes.count());
		return (clientes.findAll());
	}

	@PostMapping("/add")
	public Cliente adicionar(@Valid @RequestBody Cliente cliente) {
		return clientes.save(cliente);
	}

	@GetMapping("/{id}")
	public ResponseEntity<Optional<Cliente>> buscar(@PathVariable Long id){
		Optional<Cliente> cliente = clientes.findById(id);
		if (clientes == null) {
			return ResponseEntity.notFound().build();
		}
		return ResponseEntity.ok(cliente);
	}


	@PutMapping("/{id}")
	public ResponseEntity<Object> atualizar(@PathVariable Long id, @Valid @RequestBody
Cliente cliente)
	{
		Object atualizar = clientes.findById(id);
		if (atualizar == null) {
			return ResponseEntity.notFound().build();
		}
		BeanUtils.copyProperties(cliente, atualizar, "id");
		atualizar = clientes.save(cliente);
		return ResponseEntity.ok(atualizar);
	}

	@DeleteMapping("/{id}")
	public ResponseEntity<Void> deletar(@PathVariable Long id){
		Optional<Cliente> cliente = clientes.findById(id);
		if(cliente != null) {
			clientes.deleteById(id);
		}
		return ResponseEntity.noContent().build();
	}



}
```

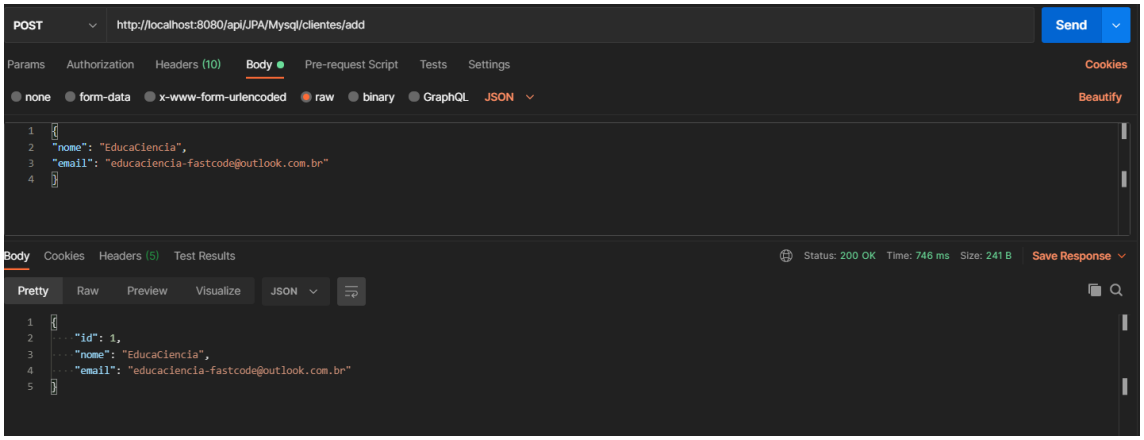
Agora, basta iniciarmos nossa “Aplicação”

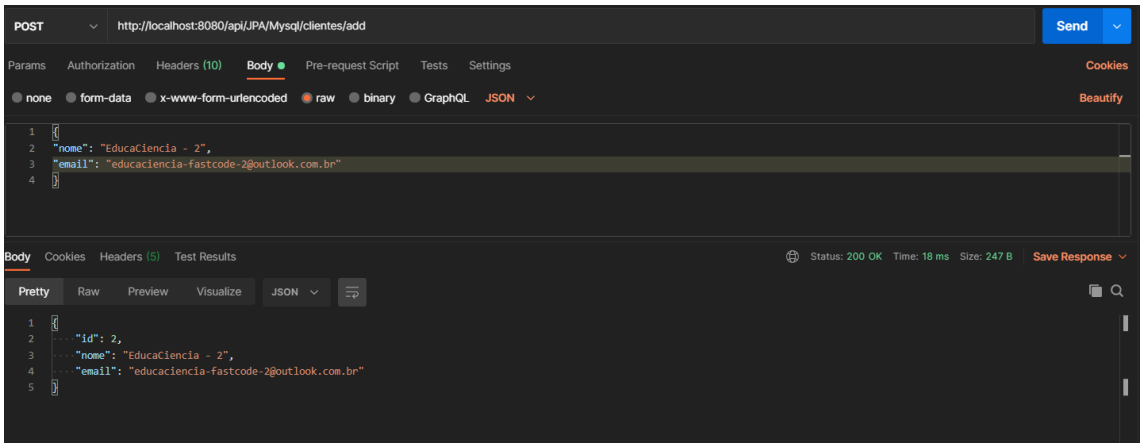
⇨ Run As \ Spring Boot App

Nossa aplicação se iniciou como esperado.

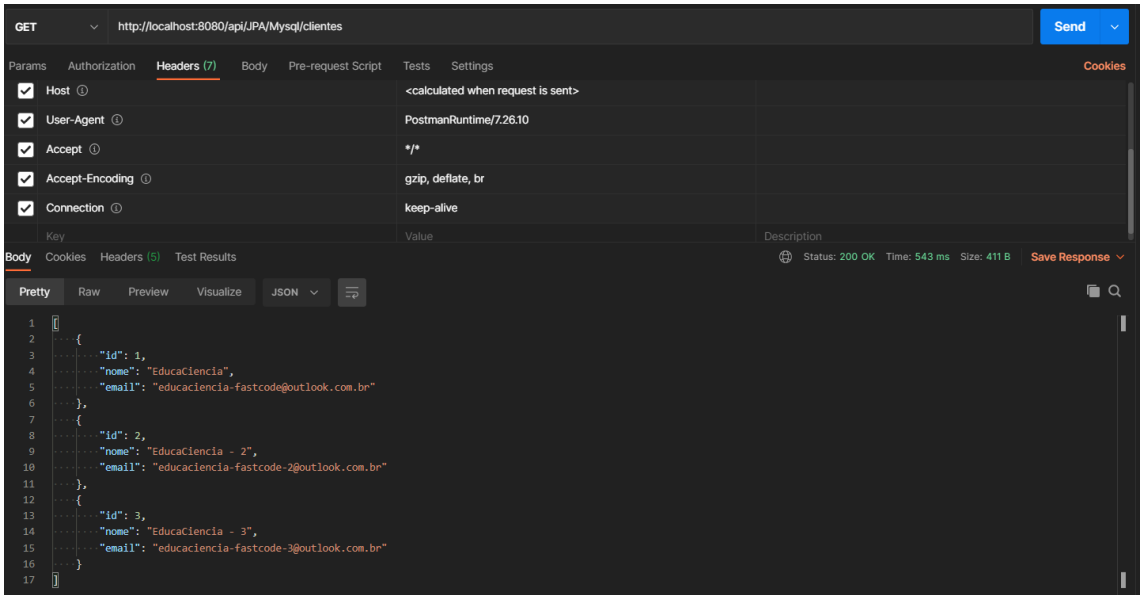
Agora podemos testar , manipulando via Postman, para isso abra seu aplicativo e vamos testar os métodos que criamos em nossa API.

Método Post – Iremos inserir três registros

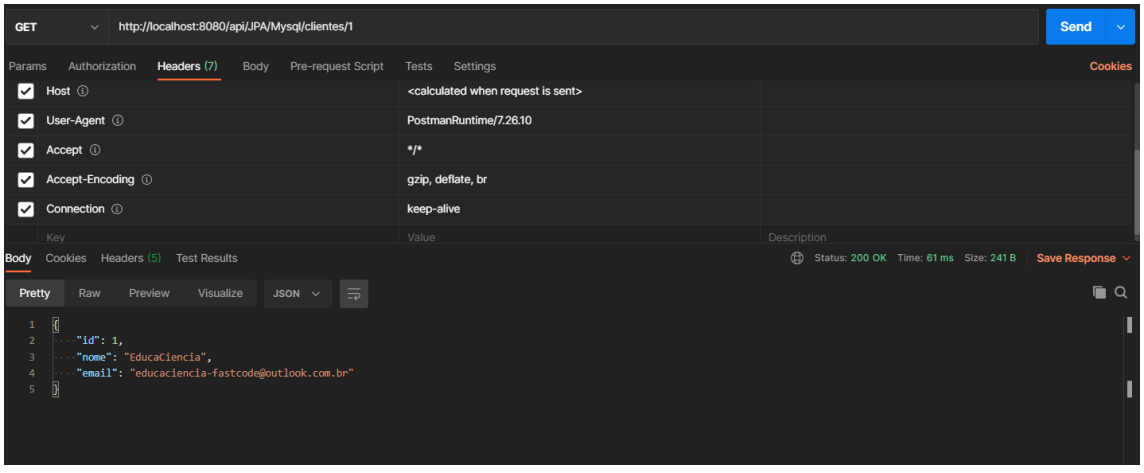


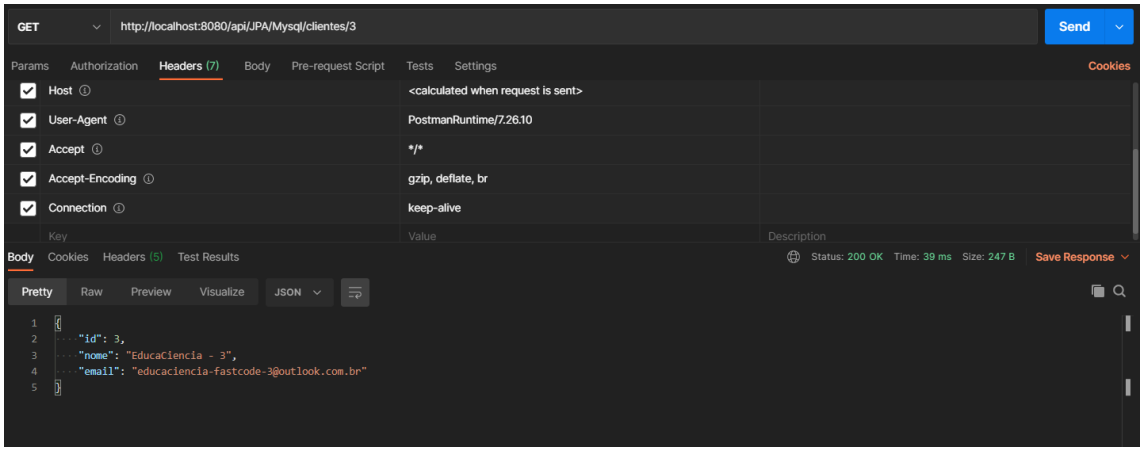


Método Get

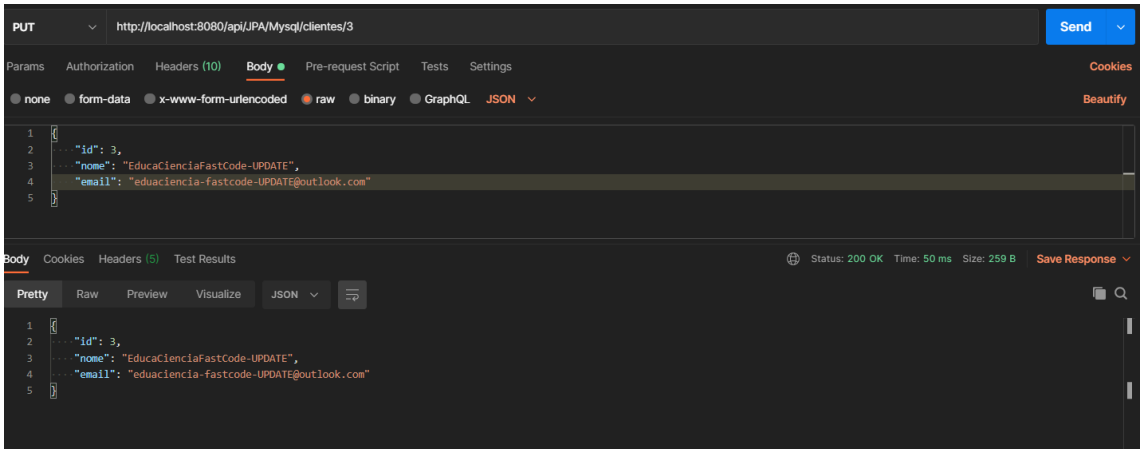


Método Get ID – Select passando argumento ID “1” e depois “3”.

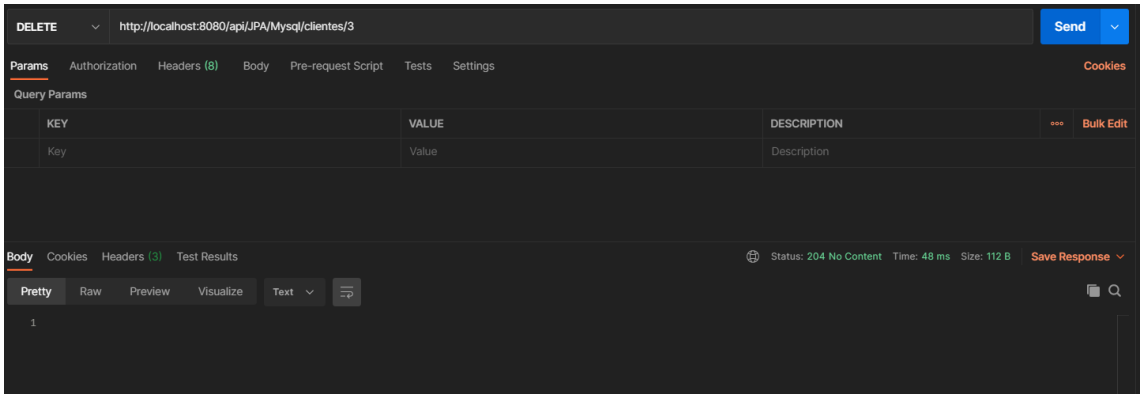




Método Put – atualizaremos registro ID“3”



Método Delete – deletando registro passando ID “3”



Nossa API funcionou como nossa proposta, sendo assim finalizamos este artigo, onde os códigos estão disponíveis no GitHub para consumo.

Até mais !
Espero ter ajudado !