



EducaCiência FastCode

Fala Galera,

- Artigo: 55/2021 Data: Maio/2021
- Público-alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: Artigo 55 - API SpringBoot com Autenticação Basic Auth - USER/ADMIN
- Link: <https://github.com/perucello/DevFP>

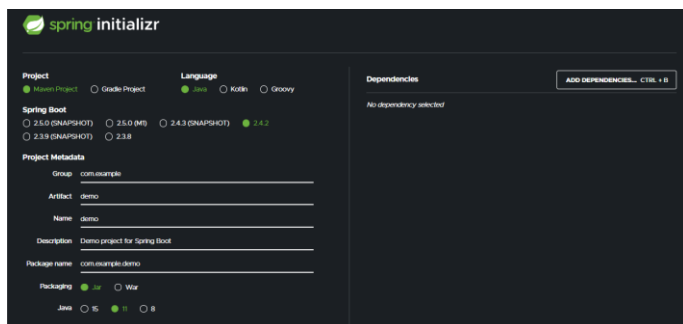
Neste artigo, utilizaremos da API desenvolvida no Artigo 47 e adicionaremos a ela uma autenticação ADMIN e USER, lembrando que este artigo é continuação do Artigo 54 porém, traremos ele reproduzido também mais abaixo.

Lembrando que os fins são didáticos !

Vamos partir da premissa que será desenvolvido, portanto, trazemos os passos para criar a API.

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>
```



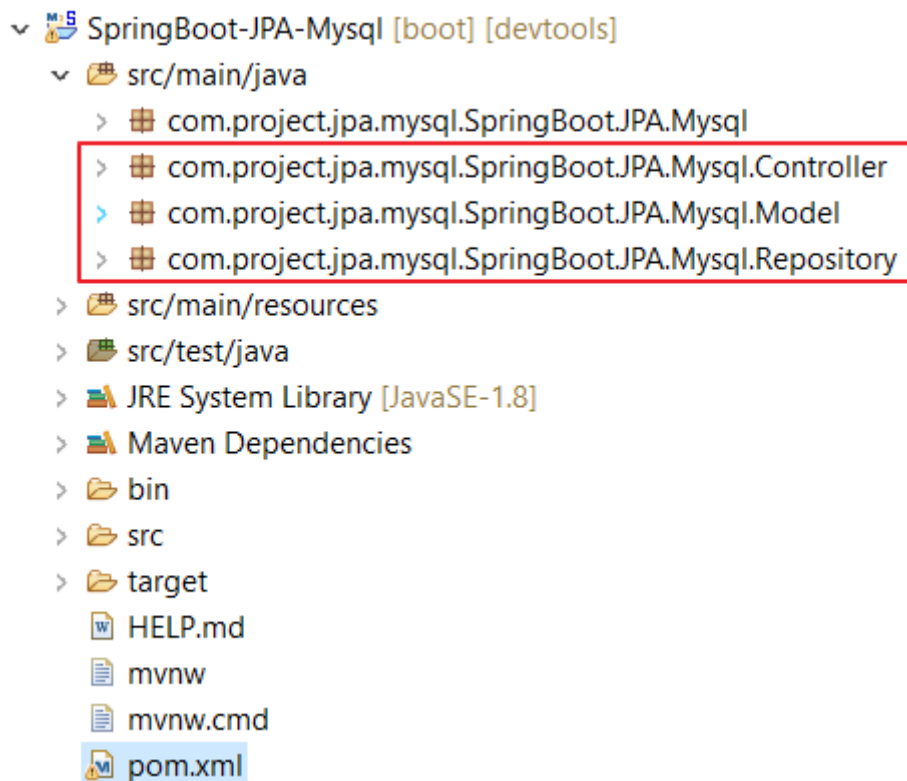
```
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>

<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:

```

v SpringBoot-JPA-Mysql [boot] [devtools]
  v src/main/java
    > com.project.jpa.mysql.SpringBoot.JPA.Mysql
    > com.project.jpa.mysql.SpringBoot.JPA.Mysql.Controller
    > com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model
    > com.project.jpa.mysql.SpringBoot.JPA.Mysql.Repository
  v src/main/resources
    static
    templates
    application.properties

```

```

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.Cientes;

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

    @Autowired
    private Cientes 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

```

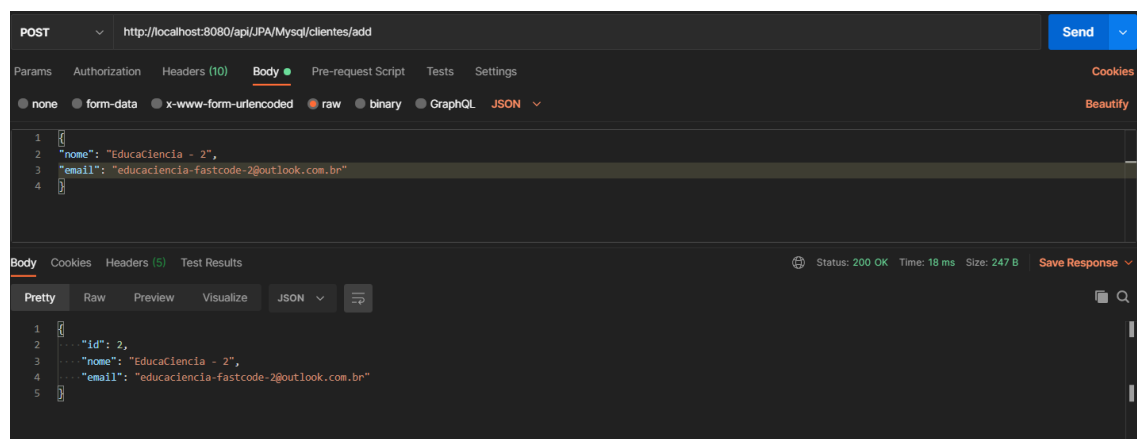
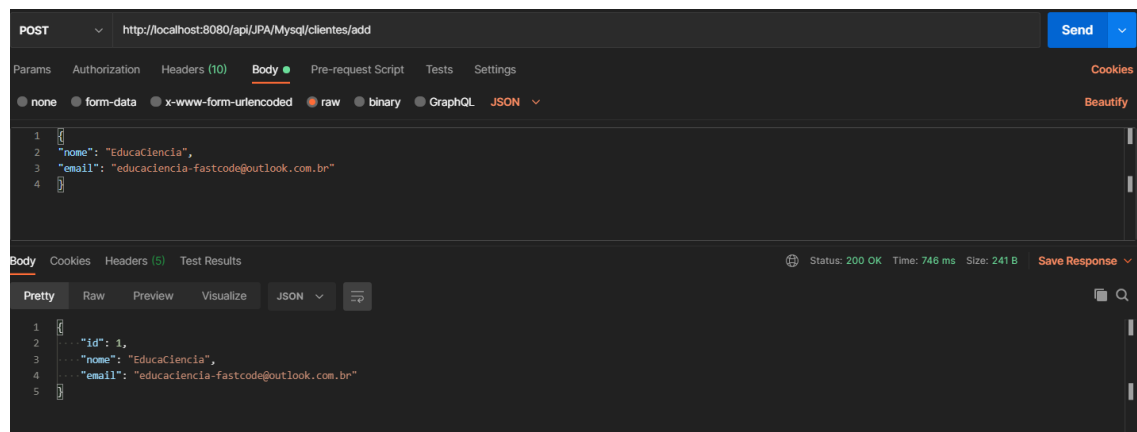
  1  (v2.4.2)
  2  :: Spring Boot ::
  3
  4  2021-02-21 13:42:25.216 INFO 13444 -- restartedMain] j.-m.s.v.M.SpringBootPaPaSysApplication : Starting SpringBootPaPaSysApplication using Java 1.8.0_212-3-redhat on DESKTOP-F3A10P2 with PID 13444
  5  2021-02-21 13:42:25.223 INFO 13444 -- restartedMain] j.-m.s.v.M.SpringBootPaPaSysApplication : No active profile set, falling back to default profiles: default
  6  2021-02-21 13:42:25.356 INFO 13444 -- restartedMain] o.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
  7  2021-02-21 13:42:25.356 INFO 13444 -- restartedMain] o.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
  8  2021-02-21 13:42:27.096 INFO 13444 -- restartedMain] s.s.d.c.RepositoryConfigurationDelegate : Bootstrapping Spring data JPA repositories in DEFAULT mode.
  9  2021-02-21 13:42:27.228 INFO 13444 -- restartedMain] s.s.d.c.RepositoryConfigurationDelegate : Finished Spring data repository scanning (84 ms). Found 1 JPA repository interfaces.
  10 2021-02-21 13:42:28.334 INFO 13444 -- restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
  11 2021-02-21 13:42:28.408 INFO 13444 -- restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
  12 2021-02-21 13:42:28.408 INFO 13444 -- restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.41]
  13 2021-02-21 13:42:28.421 INFO 13444 -- restartedMain] o.a.c.c.[/].[localhost:8080] : Initializing Spring embedded WebApplicationContext
  14 2021-02-21 13:42:28.667 INFO 13444 -- restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Root WebApplicationContext: initialization completed in 3309 ms
  15 2021-02-21 13:42:29.043 INFO 13444 -- restartedMain] o.hibernate.jpa.internal.util.LogHelper : HH0000204: Processing PersistenceUnitInfo [name: default]
  16 2021-02-21 13:42:29.184 INFO 13444 -- restartedMain] o.hibernate.Version : HH0000412: Hibernate ORM core version 5.4.27.Final
  17 2021-02-21 13:42:29.291 INFO 13444 -- restartedMain] o.s.b.c.ServletContextCustomizerWebMvc : o.Hibernate Commons Annotations (5.1.2.Final)
  18 2021-02-21 13:42:29.755 INFO 13444 -- restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 Starting...
  19 2021-02-21 13:42:30.282 INFO 13444 -- restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 Start completed.
  20 2021-02-21 13:42:30.236 INFO 13444 -- restartedMain] o.hibernate.dialect.Dialect : HH0000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
  21 2021-02-21 13:42:30.448 INFO 13444 -- restartedMain] o.h.e.j.t.jta.JtaPlatformInitiator : HH0000400: Using JtaPlatform implementation: org.hibernate.engine.transaction.jta.platform.internal.J
  22 2021-02-21 13:42:31.469 INFO 13444 -- restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
  23 2021-02-21 13:42:31.562 INFO 13444 -- restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
  24 2021-02-21 13:42:32.141 WARN 13444 -- restartedMain] jPaaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view
  25 2021-02-21 13:42:32.141 INFO 13444 -- restartedMain] o.s.b.c.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
  26 2021-02-21 13:42:32.842 INFO 13444 -- restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/'
  27 2021-02-21 13:42:32.869 INFO 13444 -- restartedMain] j.-m.s.v.M.SpringBootPaPaSysApplication : Started SpringBootPaPaSysApplication in 8.429 seconds (JVM running for 10.522)

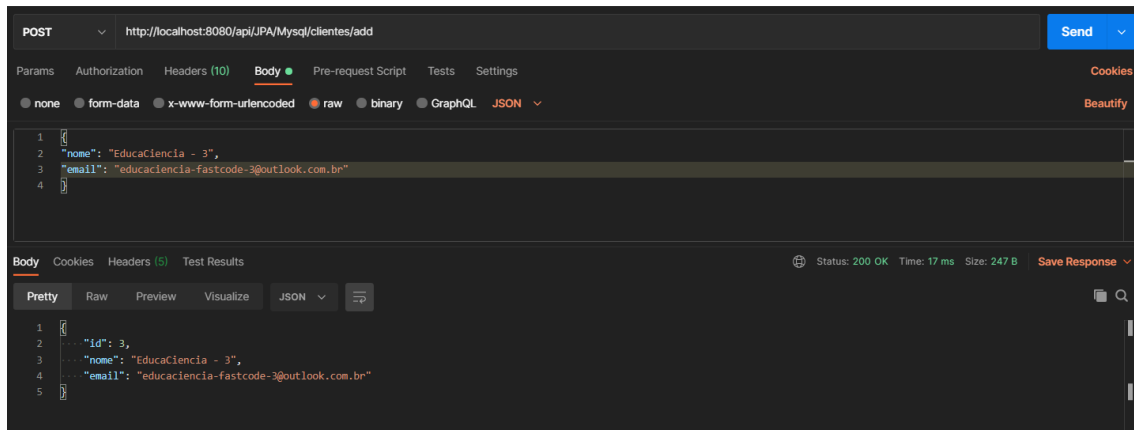
```

Nossa aplicação se iniciou como esperado.

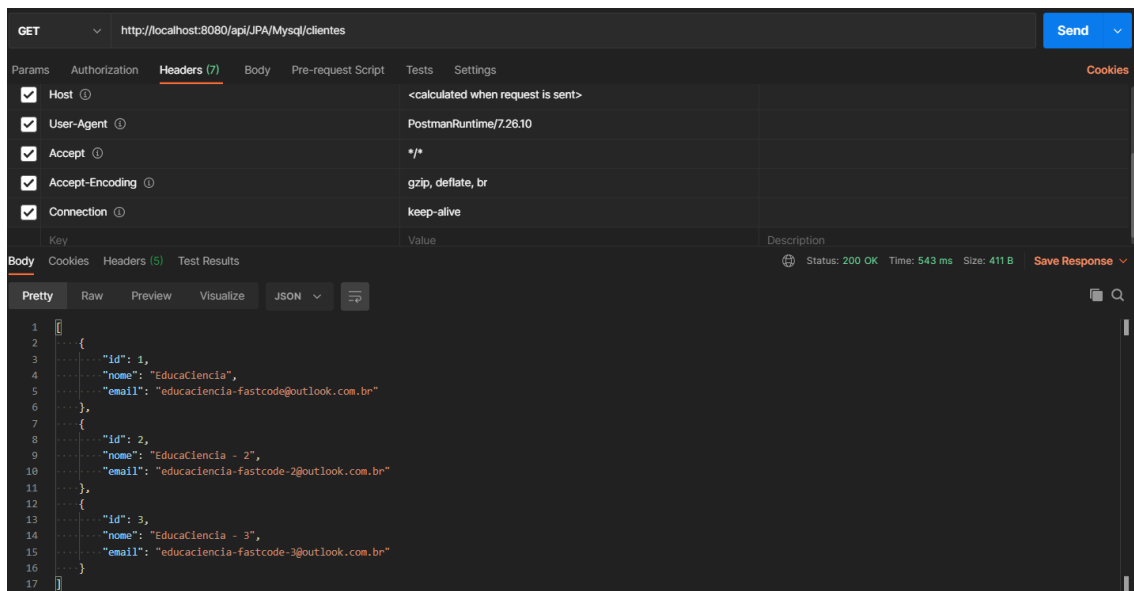
Agora podemos testar , manipulando via Postman, para isso abra seu aplicativo e vamos testar os métodos que criamos em nossa API.

Método Post – Iremos inserir três registros

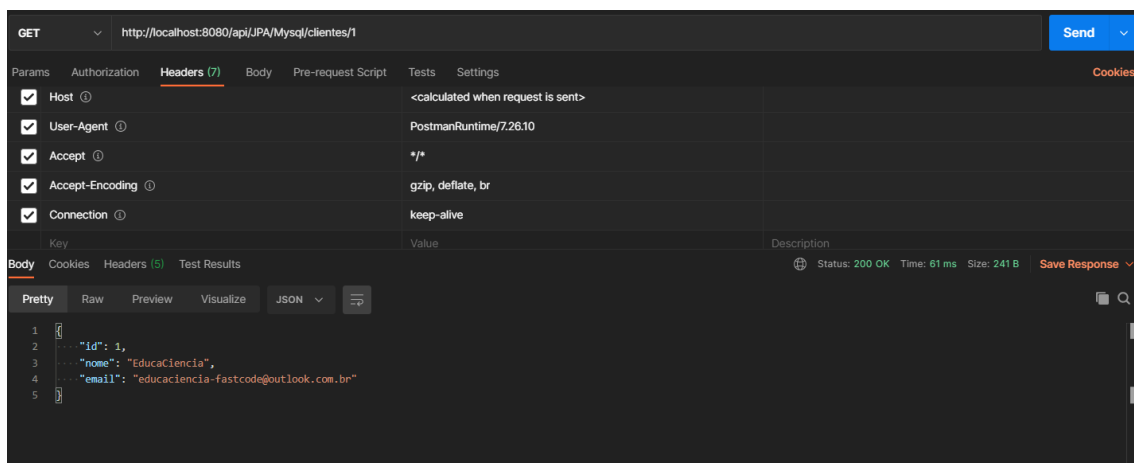




Método Get



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





GET http://localhost:8080/api/JPA/Mysql/clientes/3 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Host ☒ <calculated when request is sent>

User-Agent ☒ PostmanRuntime/7.26.10

Accept ☒ */*

Accept-Encoding ☒ gzip, deflate, br

Connection ☒ keep-alive

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "nome": "EducaCiência - 3",
4   "email": "educaciencia-fastcode-3@outlook.com.br"
5 }
```

Status: 200 OK Time: 39 ms Size: 247 B Save Response

Método Put – atualizaremos registro ID“3”

PUT http://localhost:8080/api/JPA/Mysql/clientes/3 Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "nome": "EducaCiênciaFastCode-UPDATE",
4   "email": "educaciencia-fastcode-UPDATE@outlook.com"
5 }
```

Status: 200 OK Time: 50 ms Size: 259 B Save Response

Método Delete – deletando registro passando ID “3”

DELETE http://localhost:8080/api/JPA/Mysql/clientes/3 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize Text

```
1
```

Status: 204 No Content Time: 48 ms Size: 112 B Save Response

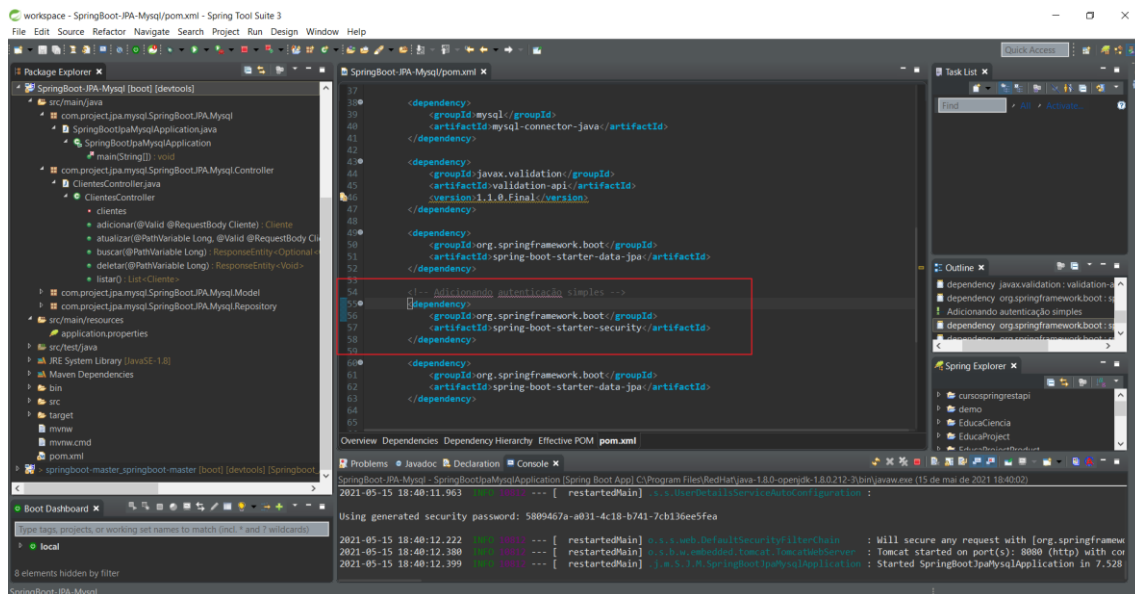
Nossa API funcionou como nossa proposta, sendo assim finalizamos este artigo, onde os códigos estão disponíveis no GitHub para consumo.



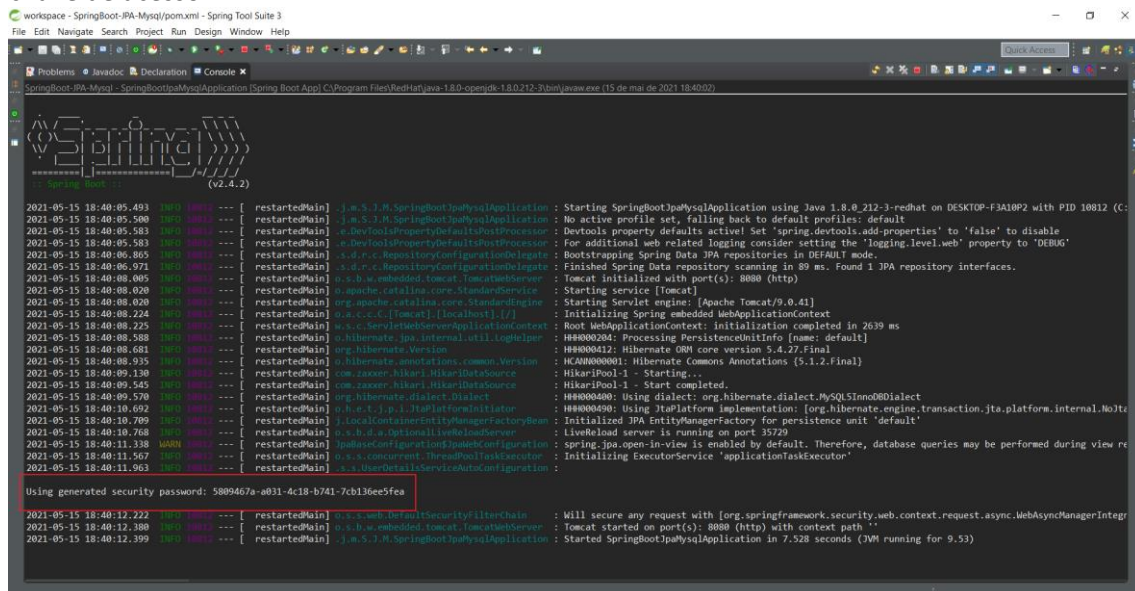


Com seu projeto startado, iremos, no entanto, no nosso pom.xml e adicionaremos a dependência referente à Autenticação que iremos manipular.

```
<!-- Adicionando autenticação simples -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```



Note que após adicionarmos a dependência, e startar nosso Sistema temos já no Output uma chave de acesso.

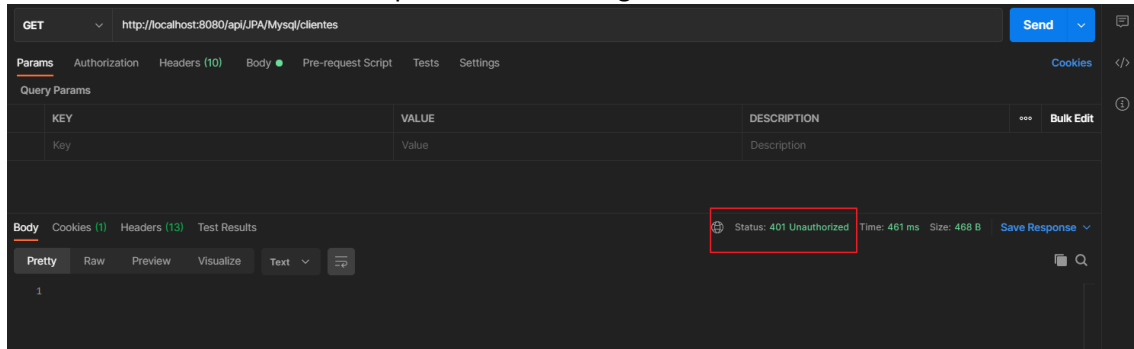


Apenas com esta ação, já temos uma segurança com relação aos nossos Metodos. Sendo assim, vamos testar no Postman nosso método GET



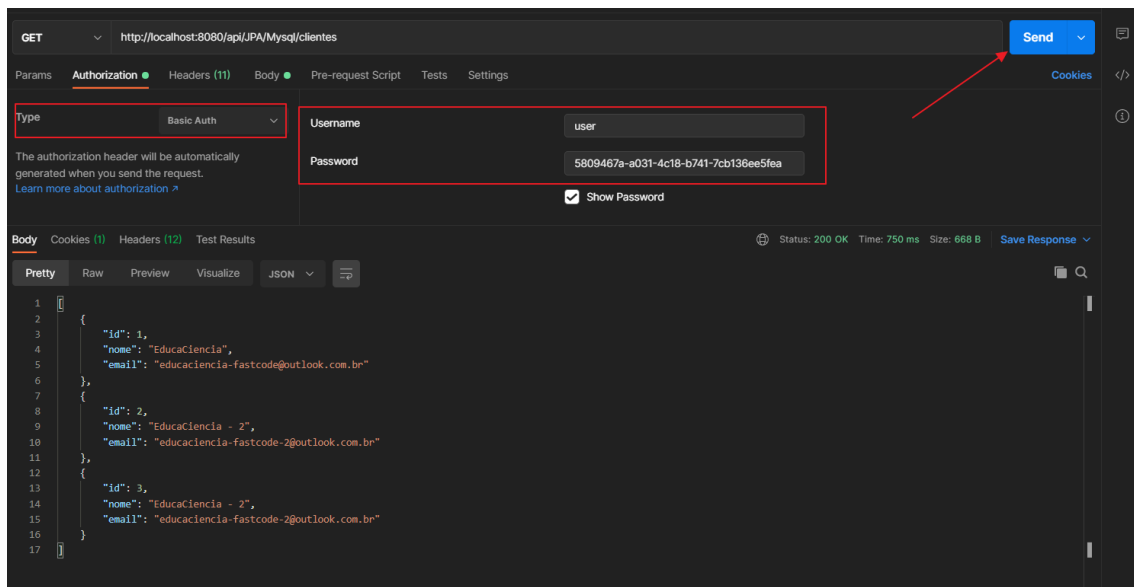


Abra o Postman e execute o endpoint como na imagem abaixo:



Note que ao executarmos o endpoint, recebemos retorno de que não temos autorização !
Exatamente isso que queríamos, agora iremos validar.

Portanto localize a aba “AUTHORIZATION”, selecione o tipo “Basic Auth” e insira as credenciais como na imagem abaixo, e clique em executar.

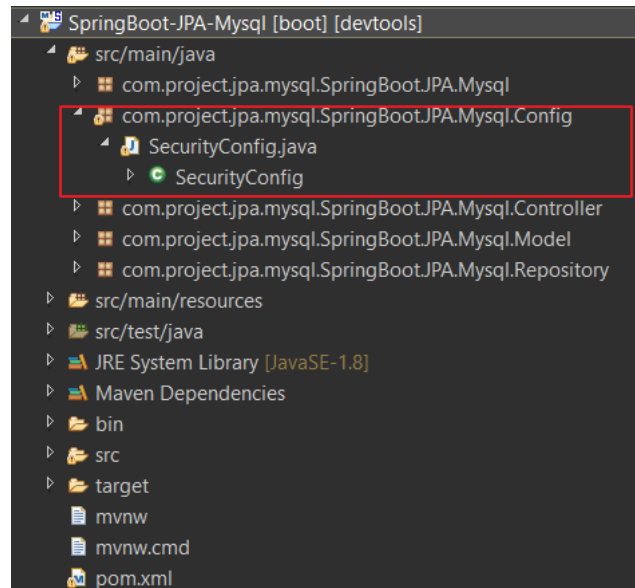


Com isso , temos já uma segurança pra execução dos métodos GET.



Agora, iremos dar início na proposta do nosso artigo que é adicionarmos uma AUTENTICAÇÃO ADMIN e USER.

Para isso, criaremos um pacote chamado “Config” e nesse pacote uma classe chamada “SecurityConfig”, portanto crie o pacote e a classe e insira o código abaixo.



```
SecurityConfig.java x
1 package com.project.jpa.mysql.SpringBoot.JPA.Mysql.config;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
6 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
7 import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
8
9 @Configuration
10 public class SecurityConfig extends WebSecurityConfigurerAdapter
11 {
12     @Override
13     protected void configure(HttpSecurity http) throws Exception
14     {
15         http
16             .csrf().disable()
17             .authorizeRequests().anyRequest().authenticated()
18             .and()
19             .httpBasic();
20     }
21
22     @Autowired
23     public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception
24     {
25         auth.inMemoryAuthentication()
26             .withUser("user").password("{noop}1234").roles("USER")
27             .and()
28             .withUser("admin").password("{noop}1234").roles("ADMIN");
29     }
30 }
```

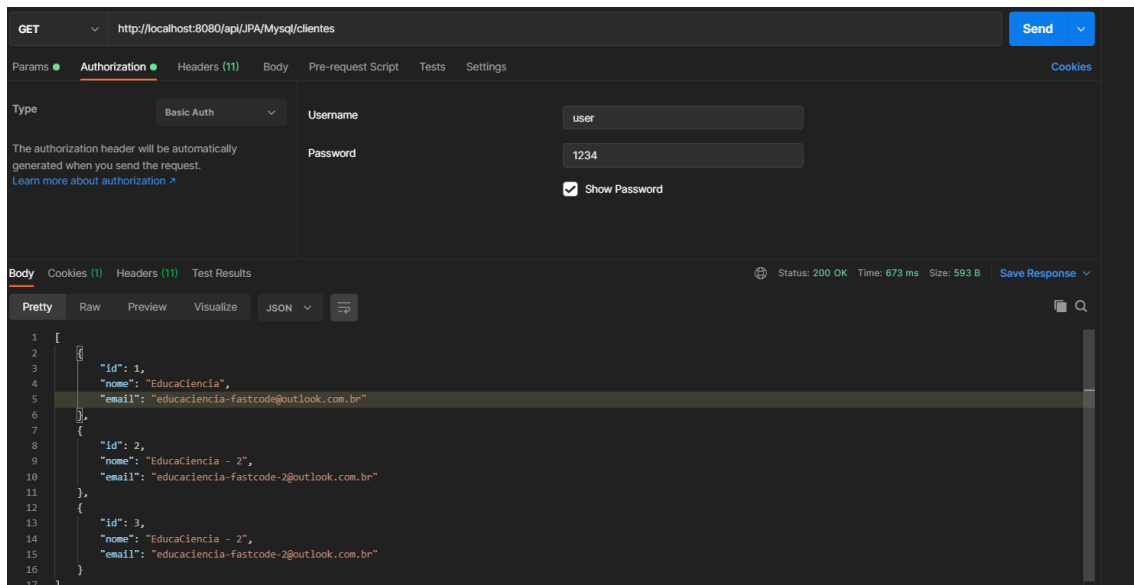


Criamos nessa classe um autenticador para usuário e um para administrador.
Agora, basta iniciar o serviço para testarmos.

Vamos ao nosso Postman e insira as credenciais que acabamos de criar.

Username: user

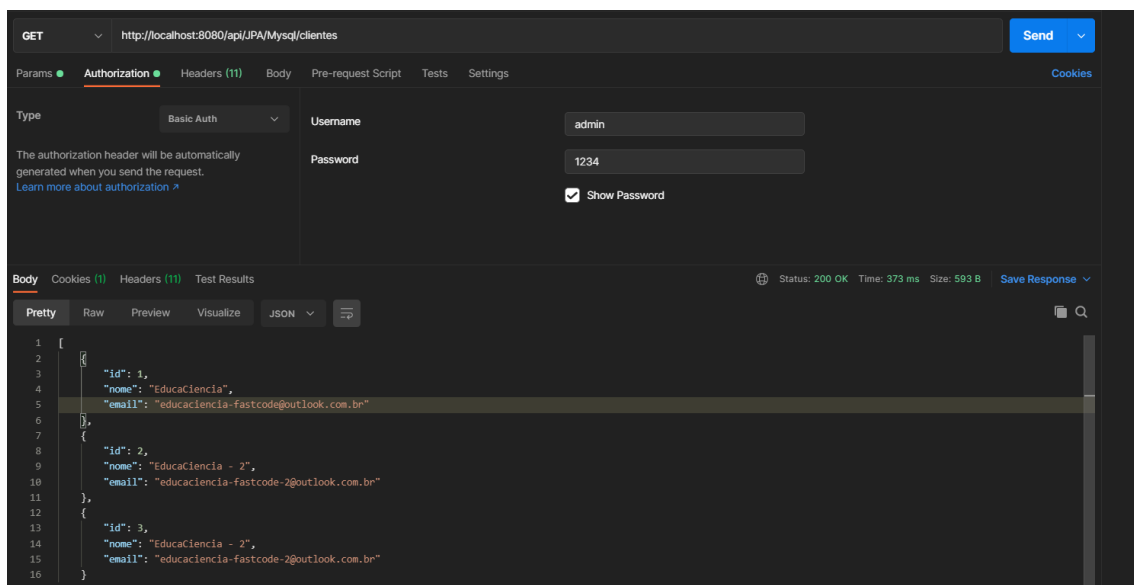
Password: 1234



Agora iremos testar nosso administrador

Username: admin

Password: 1234





No console tem nosso registro do Get para “user” e “admin”

```
Problems Javadoc Declaration Console x
SpringBoot-JPA-Mysql - SpringBootJpaMysqlApplication [Spring Boot App] C:\Program Files\RedHat\java-1.8.0-openjdk-1.8.0.212-3\bin\javaw.exe (15 de mai de 2021 22:54:13)
2021-05-15 22:54:21.936 INFO 10364 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context pat
2021-05-15 22:54:21.954 INFO 10364 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMysqlApplication : Started SpringBootJpaMysqlApplication in 7.142 seconds
2021-05-15 22:56:20.425 INFO 10364 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServle
2021-05-15 22:56:20.425 INFO 10364 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-05-15 22:56:20.427 INFO 10364 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
Quantidade de Registros de Clientes : 3
Quantidade de Registros de Clientes : 3
```

Testando Post

```
POST http://localhost:8080/api/JPA/Mysql/clientes/add
Body
[{"nome": "EducaCiencia",
  "email": "educaciencia-fastcode@outlook.com.br"}]
Body
{"id": 4,
  "nome": "EducaCiencia",
  "email": "educaciencia-fastcode@outlook.com.br"}
Status: 200 OK Time: 125 ms Size: 423 B Save Response
```

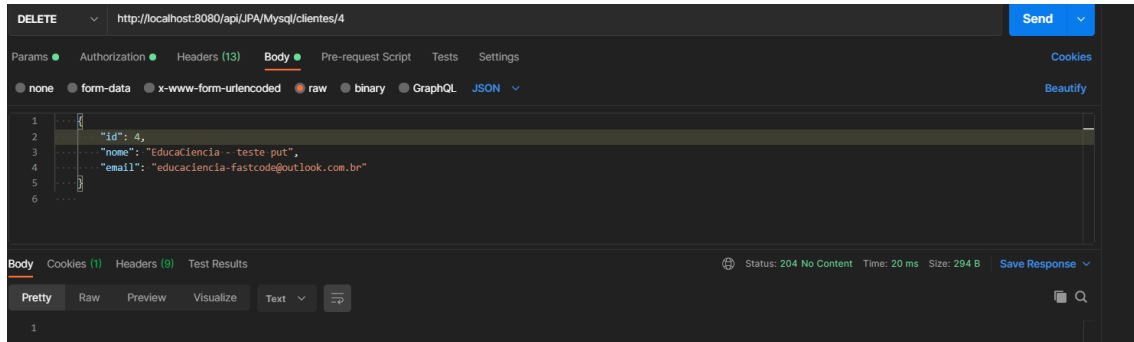
Teste Put

```
PUT http://localhost:8080/api/JPA/Mysql/clientes/4
Body
{"id": 4,
  "nome": "EducaCiencia - teste put",
  "email": "educaciencia-fastcode@outlook.com.br"}
Body
{"id": 4,
  "nome": "EducaCiencia - teste put",
  "email": "educaciencia-fastcode@outlook.com.br"}
Status: 200 OK Time: 47 ms Size: 435 B Save Response
```

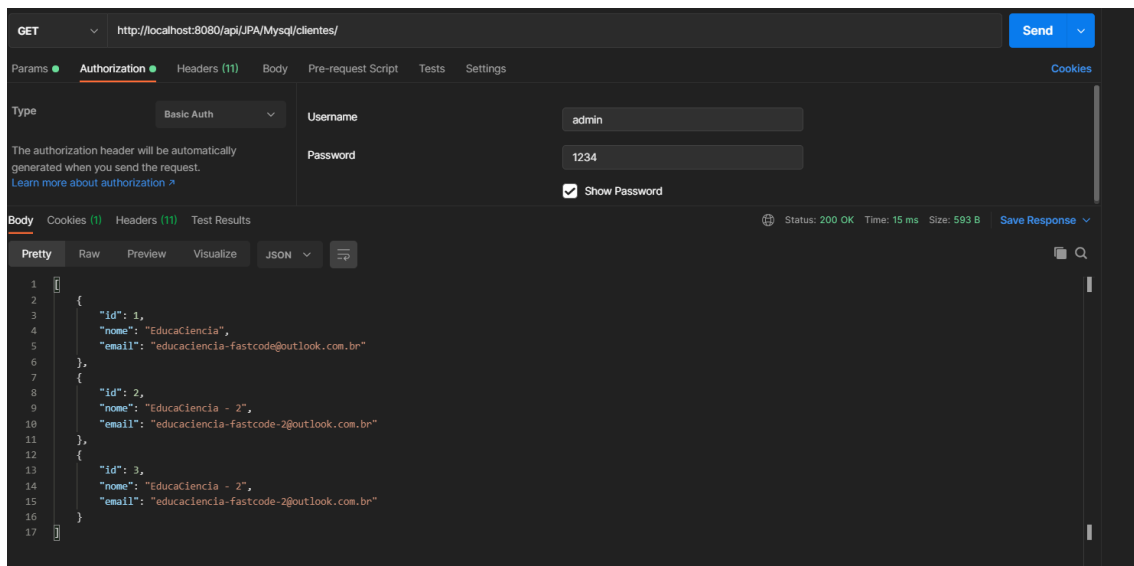




Teste Delete



Novamente GET – após atualizações acima



Com isso concluímos nossa proposta.

Até mais !

Espero ter ajudado !

