

[https://www.youtube.com/playlist?list=PL62G310vn6nF3gssjqfCKLpTK2sZJ\\_a\\_1](https://www.youtube.com/playlist?list=PL62G310vn6nF3gssjqfCKLpTK2sZJ_a_1)

<https://github.com/devdojobr/springboot-essentials>

Resumo: Para rodar aplicação precisa criar uma Schema school dentro de um Workbench do MySQL(Aula 9)(As tables dentro desta database school o aplicativo cria.

## Aula 1: Spring Boot Essentials 00 - Apresentação do curso

## Aula 2: Spring Boot Essentials 01: Setup do projeto

Realizado configuração do zero manualmente(Cria Classe Student retornando dentro da propria classe os dados de student para servir como dados):

```
@RequestMapping(method = RequestMethod.GET, path = "/list")
public List<Student> listAll() {
    return asList(new Student("Deku"), new Student("Todokori"));
}
```

Obs.: Tem detalhes no arquivo abaixo na parte de Spring via IntelliJ

<https://onedrive.live.com/edit.aspx?cid=b441b70f8e9d4017&page=view&resid=B441B70F8E9D4017!998&parId=B441B70F8E9D4017!990&app=Word>

## Aula 3: Spring Boot Essentials 02 - @Component, @Autowired e @SpringBootApplication

Para usar o DateUtil (mostrar a data no console cada vez que roda o endpoint) adiciona configuração no pom para usar a versão 8 do Java.

Movido a ApplicationStart para a raiz(para não precisar definir no @ComponentScan o endereço dos pacotes)

Adicionado a anotação `@SpringBootApplication` que agrupa as `@EnableAutoConfiguration` `@ComponentScan` `@Configuration`

## Aula 4: Spring Boot Essentials 03 - Configurando hot swap

O hot swap altera o comportamento do IntelliJ para reiniciar automaticamente a aplicação(sobe a aplicação a cada alteração).

Precisa acrescentar esta dependencia no pom:

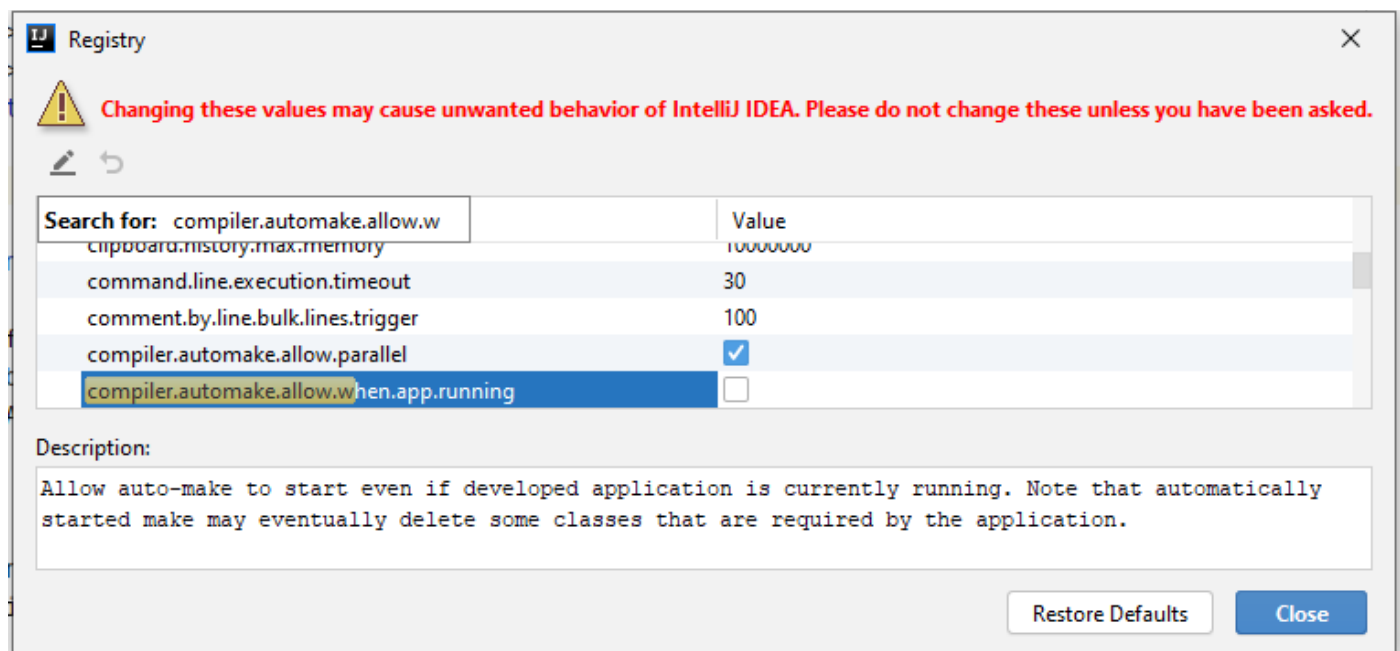
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <optional>true</optional>
</dependency>
```

+ File + Settings + Build, Execution, Deployment + Compiler +

☒ Automatically show first error in editor

+ ok + (Ctrl-Shift+A) + Registry +

compiler.automake.allow.when.app.running



+ tem que reiniciar o IntelliJ.

## Aula 5: Spring Boot Essentials 04 - Setup do projeto com Spring Initializr

Utilizou a versão paga do IntelliJ para criar de dentro do IntelliJ utilizando o Spring Initializer

Como na versão Community 2020.1 não tinha esta opção eu alterei manualmente o arquivo para poder ficar igual(semelhante) ao do curso

1)Removi o diretorio Awesome precisando refatorar as classes

2)Adicionei a dependencia 'para testes

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

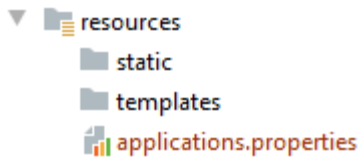
3) Criei os diretorios br.com.devdojo e a classe de testes

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringEssentialsApplicationsTests {

    @Test
    public void contextLoads(){
    }

}
```

4) Criei os diretorios static e templates e o arquivo applications.properties



## Aula 6: Spring Boot Essentials 05 - Padrões REST e POST pt 01

### [Spring Boot Essentials 05 - Padrões REST e POST pt 01](#)

Adicionado metodo para listar por id

Alterado classe Student: 1) Adicionado states{studentRepository();}

2) Adicionado lista de Students dentro da propria classe

```
private static void studentRepository(){  
    studentList = new ArrayList<>(asList(new Student(1,"Deku"), new Student(2,"Todokori")));  
}
```

Isto foi feito para testar o endpoint rapidamente, depois sera adicionado banco de dados

3) Adicionado metodo

```
@RequestMapping(method = RequestMethod.GET, path=("/{id}")  
public ResponseEntity<?> getStudentById(@PathVariable("id") int id) {  
    Student student = new Student();  
    student.setId(id);  
    int index = Student.studentList.indexOf(student);  
    if(index == -1)  
        return new ResponseEntity<>(new CustomErrorType("Student not found"),  
HttpStatus.NOT_FOUND);  
    return new ResponseEntity<>(Student.studentList.get(index), HttpStatus.OK);  
}
```

## Aula 7: Spring Boot Essentials 06 - Padrões REST e POST pt 02

### [Spring Boot Essentials 06 - Padrões REST e POST pt 02](#)

Requisição POST + Postman

```
@RequestMapping(method = RequestMethod.POST)  
public ResponseEntity<?> save(@RequestBody Student student){  
    Student.studentList.add(student);  
    return new ResponseEntity<>(student, HttpStatus.OK);  
}
```

## Aula 8: Spring Boot Essentials 07 - Padrões REST e PUT e DELETE pt 03

### [Spring Boot Essentials 07 - Padrões REST e PUT e DELETE pt 03](#)

Adicionado Delete e PUT

```

@RequestMapping(method = RequestMethod.DELETE)
public ResponseEntity<?> delete(@RequestBody Student student){
    Student.studentList.remove(student);
    return new ResponseEntity<>(HttpStatus.OK);
}

```

```

@RequestMapping(method = RequestMethod.PUT)
public ResponseEntity<?> update(@RequestBody Student student){
    Student.studentList.remove(student);
    Student.studentList.add(student);
    return new ResponseEntity<>(HttpStatus.OK);
}

```

Melhorado a nomenclatura dos metodos Mapping

`@RequestMapping(method = RequestMethod.PUT)` --> `@PutMapping`

`@RequestMapping(method = RequestMethod.DELETE)` --> `@DeleteMapping`

`@RequestMapping(method = RequestMethod.POST)` --> `@PostMapping`

`@RequestMapping(method = RequestMethod.GET, path=("/{id}"))` --> `@GetMapping(path=("/{id}"))`

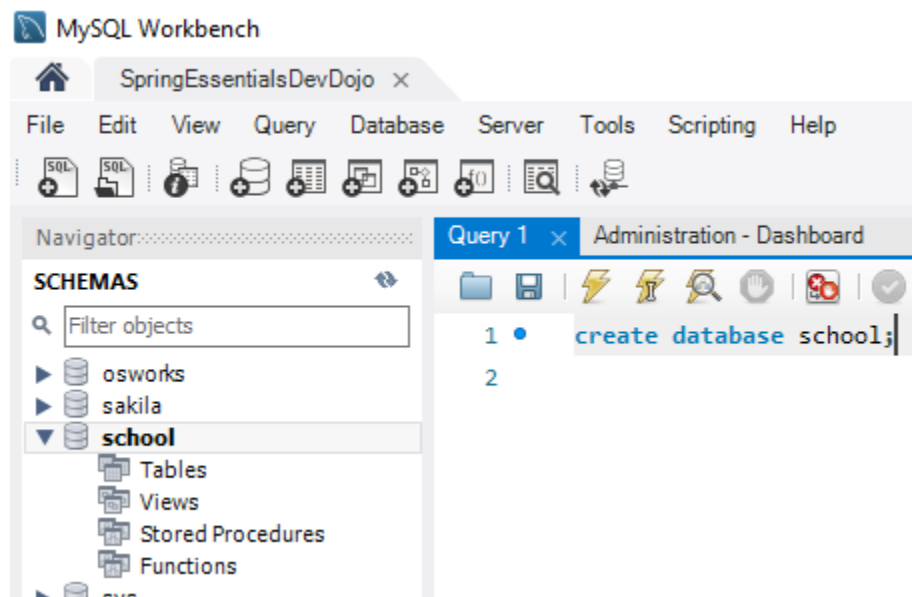
`@RequestMapping(method = RequestMethod.GET)` --> `@GetMapping`

## Aula 9: Spring Boot Essentials 08 - Adicionando Spring Data JPA com MySQL pt 01

### [Spring Boot Essentials 08 - Adicionando Spring Data JPA com MySQL pt 01](#)

Adicionado JPA para trabalhar com BD MySQL

1º) Criado Schema

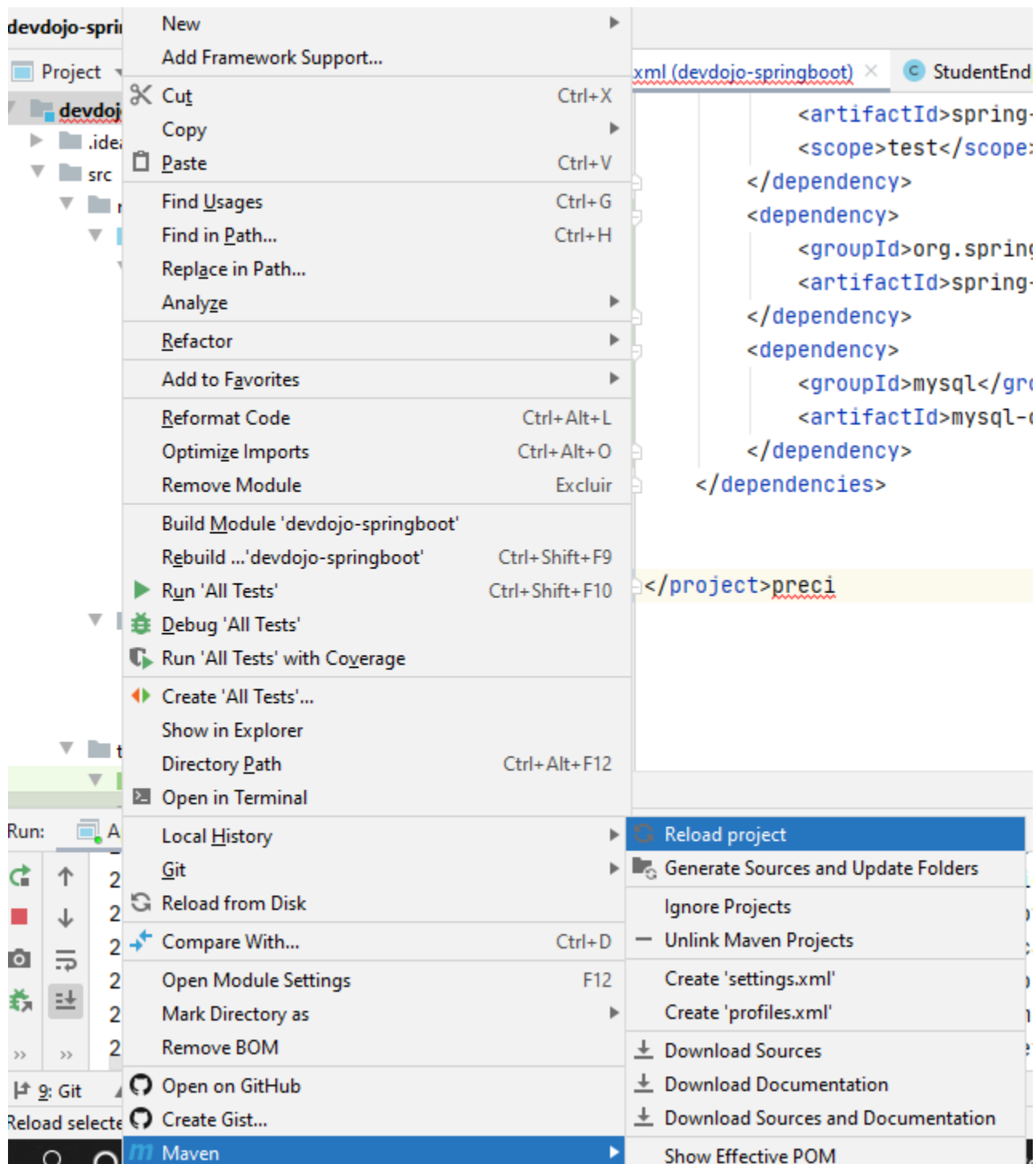


2) Adicionado dependencia

```

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

```



### 3) Configurado application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/school?useSSL=false
spring.datasource.username=root
spring.datasource.password=122026
```

```
spring.datasource.tomcat.test-while-idle=true
spring.datasource.tomcat.validation-query=SELECT 1
```

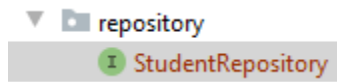
```
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
```

#### 4) Alterado Classe Student

A classe student foi simplificada e extendida de uma nova Classe chamada AbstractEntity que foi criada para ter o Id e os hascode e equals.

Também colocado a anotação @Entity

#### 5) Criado Package e Classe Repository



```
public interface StudentRepository extends CrudRepository<Student, Long> {  
  
    List<Student> findByName(String name);  
  
}
```

#### 6) Alterado EndPoint

7) Ao rodar deu erros e precisei acrescentar em application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

E alterar

```
spring.datasource.url=jdbc:mysql://localhost:3306/school?allowPublicKeyRetrieval=true&  
useSSL=false&useTimezone=true&serverTimezone=UTC
```

E adicionado no pom dependency mysql

```
<scope>runtime</scope>  
<version>8.0.13</version>
```

### Aula 10: Spring Boot Essentials 09 - Adicionando Spring Data JPA com MySQL pt 02

[Spring Boot Essentials 09 - Adicionando Spring Data JPA com MySQL pt 02](#)

Utilizado o metodo findByName alterando para pegar por parte do nome + alterado endpoint de PUT para resposta CREATED que retorna 201

### Aula 11: Spring Boot Essentials 10 - Tratamento de erros em REST pt 01

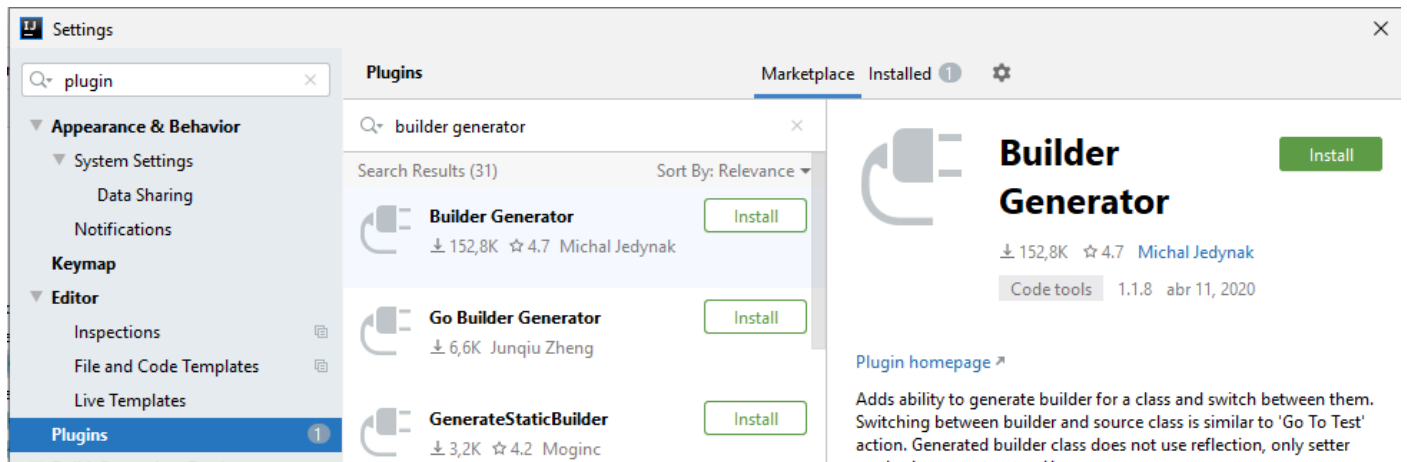
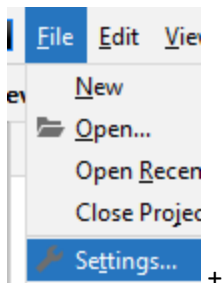
[Spring Boot Essentials 10 - Tratamento de erros em REST pt 01](#)

Adicionado metodo para tratar quando não encontra o id no metodo de GET,PUT ou DELETE

### Aula 12: Spring Boot Essentials 11 - Tratamento de erros em REST pt 02

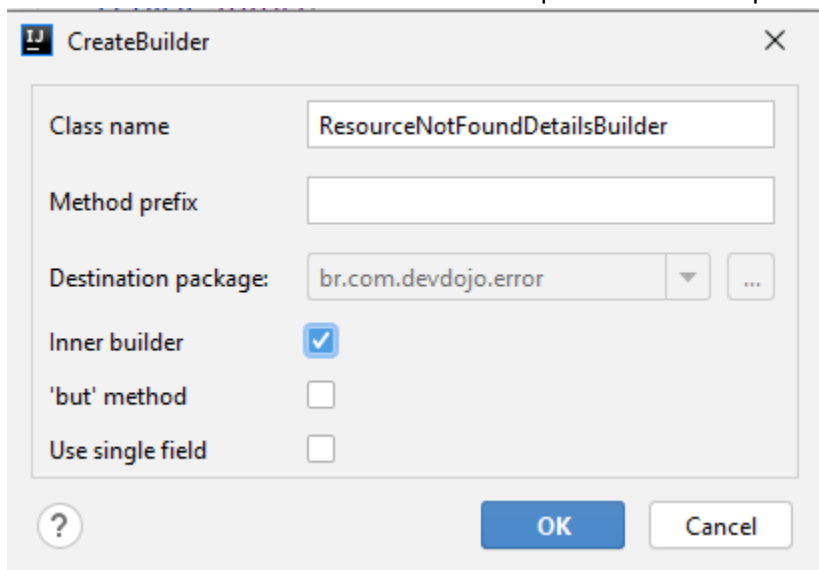
[Spring Boot Essentials 11 - Tratamento de erros em REST pt 02 - Exception Handler](#)

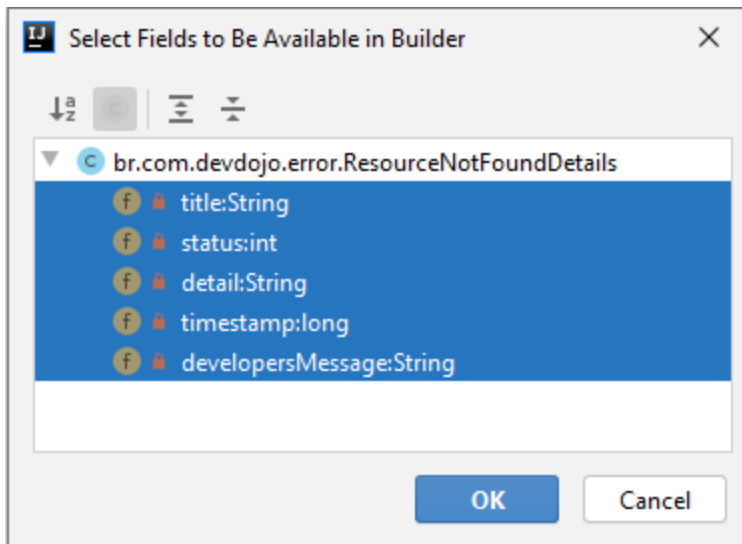
Personalizando/Alterando o corpo da mensagem de erro com handler



+ Reinicializa o IntelliJ

Ao criar a Classe ResourceNotFoundDetails aperta Alt+Shift+B para utilizar o plugin





```
public static ResourceNotFoundDetailsBuilder aResourceNotFoundDetails() {
    return new ResourceNotFoundDetailsBuilder();
}
```

renomeia

para newBuilder --> `public static ResourceNotFoundDetailsBuilder newBuilder() {`

```
1 {
2     "title": "Resource not found",
3     "status": 404,
4     "detail": "Student not found for ID: 10",
5     "timestamp": 1604780423286,
6     "developersMessage": "br.com.devdojo.error.ResourceNotFoundException"
7 }
```

## Aula 13: Spring Boot Essentials 12 - Tratamento de erros em REST pt 03

### Spring Boot Essentials 12 - Tratamento de erros em REST pt 03 - Transações

Se a aplicação for pouquinho maior do que pequeno porte precisa desta transação

Colocar em Application.properties InnoDB para garantir que tabela seja criado com esta propriedade:

```
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
```

Após isto o video virou uma confusão: NÃO FICOU CLARO NO VIDEO COMO E O QUE FAZER e para que fazer

## Aula 14: Spring Boot Essentials 13 - Tratamento de erros em REST pt 04

### Spring Boot Essentials 13 - Tratamento de erros em REST pt 04 - Validação de campos

Adicionado email na classe student

@NotEmpty

@Email



@Valid

Adicionado metodo em ResExceptionHandler e criado Classe ErrorDetail

## Aula 15: Spring Boot Essentials 14 - Tratamento de erros em REST pt 05

[Spring Boot Essentials 14 - Tratamento de erros em REST pt 05 - Padronizando todos os erros](#)

Alt+Shift+B para criar novo Builder

Aula bem densa que foi criado um padrão para todas as msg de erro

## Aula 16 – Paginação em requisições REST

[Spring Boot Essentials 15 - Paginação em requisições REST](#)

Alterado Repository:

```
public interface StudentRepository extends  
PagingAndSortingRepository<Student, Long> {
```

Alterado o metodo

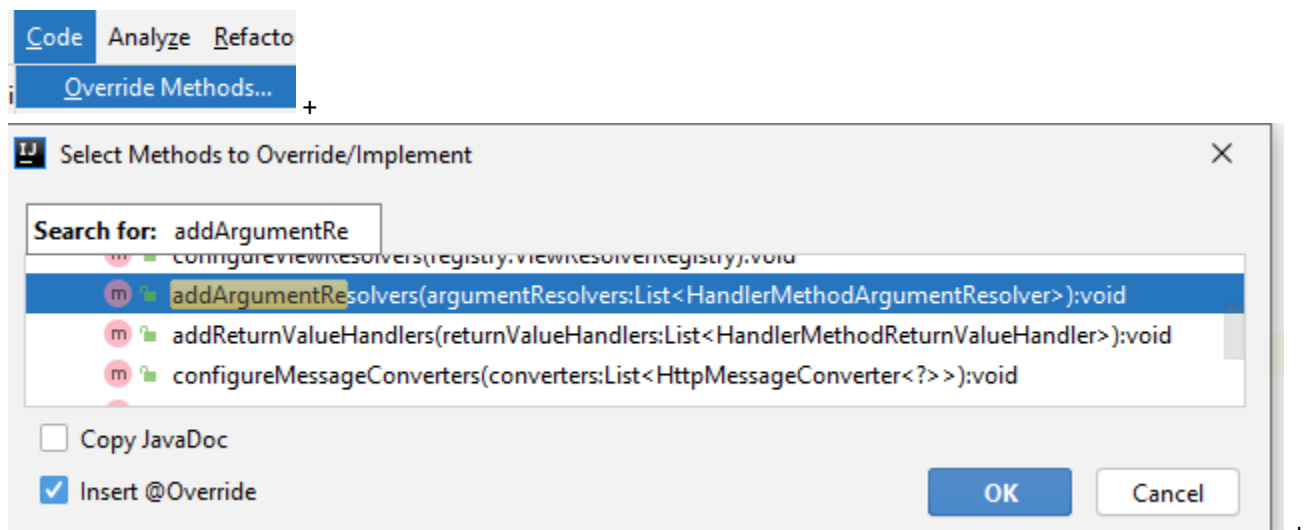
@GetMapping

```
public ResponseEntity<?> listAll(Pageable pageable) {  
    return new ResponseEntity<>(studentDAO.findAll(pageable),  
HttpStatus.OK);  
}
```

localhost:8080/students?page=0

localhost:8080/students?page=1&size=2

Para personalizar a configuração da quantidade por pagina:



## Aula 17 – Ordenação em requisições REST

### [Spring Boot Essentials 16 - Ordenação em requisições REST](#)

Para classificar usa-se somente a requisição no postman com o parametro e ,asc ou ,des  
localhost:8080/students?sort=name,asc

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/009de13625f018f2c984d7ec9e27e88fa199496e>

## Aula 18 – Spring Security Parte 1 – Autenticação e autorização

### [Spring Boot Essentials 17 - Spring Security pt 01 - Autenticação e autorização](#)

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

Ao colocar a dependencia de segurança o spring gera uma senha no log de inicialização que pode ser usado para acessar os end point

The screenshot shows the Postman interface for a GET request to `localhost:8080/students`. The 'Authorization' tab is selected, showing 'Basic Auth' as the type. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The 'Username' field contains 'user' and the 'Password' field contains 'bb13bbfe-bb78-424b-8c28-d374a1434929'. The 'Show Password' checkbox is checked.

Mas desta forma não é segura então um primeira forma é colocar usuario e senha em uma classe do java

3:44 Adicionado novo package e Classe

Senha usada no commit

```
@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
    auth.inMemoryAuthentication() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
        .withUser( username: "willian").password("devdojo").roles("USER") UserDetailsManagerConfigurer<B, C>.UserDel
        .and() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
        .withUser( username: "admin").password("devdojo").roles("USER", "ADMIN");
}
```

GET localhost:8080/students Send Save

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

**TYPE**  
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username admin

Password devdojo

☒ Show Password

Body Cookies (1) Headers (10) Test Results 200 OK 520 ms 675 B Save Response

Commit: [Adicionado Autenticação e Autorização](#)

<https://github.com/eliseusbrito/SpringEssentialsDevDojo/commit/0b337eea28946fa2cb1f76c931dab035cca73a5a>

## Aula 19 – Spring Security Parte 2 – Autenticação e autorização com Spring Data

[Spring Boot Essentials 18 - Spring Security pt 02 - Autenticação e autorização com Spring Data](#)

Validação das credenciais com uma tabela no bando de dados

1:15 Annotations para User

4:02 Criptografando a senha

6:17 coloca valor criptografado no BD

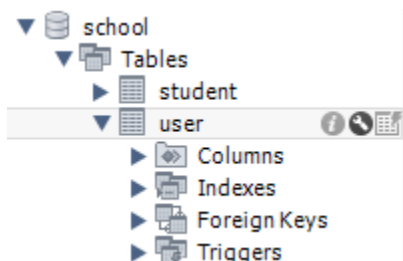
7:13 conclui inserção usuarios do BD


7:30 Cria User Repository

8:40 Cria um Custom Detail Service

15:40 Configura o Security config

O java cria conforme abaixo



No banco de dados precisei utilizar o  +  para editar a Table

user - Table x

Table Name:  Schema: **school**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
admin	TINYINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
name	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
password	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
user_name	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name:  Data Type:

Charset/Collation:

Comments:

Default:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Principal dificuldade foi no campo admin onde precisei utilizar TINYINT que faz o papel de BOOLEAN do java.

Para editar o user usei o  ficando conforme abaixo

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

	id	admin	name	password	user_name
▶	1	1	Shimazu Toyohisa	\$2a\$10\$1xXM4WMLKQ.nsCruPwk.A.aoUpwZnkRqIFKpagR...	toyo
▶	2	0	Oda Nobunaga	\$2a\$10\$1xXM4WMLKQ.nsCruPwk.A.aoUpwZnkRqIFKpagR...	oda
*	NULL	NULL	NULL	NULL	NULL

user 1 x

Apply Revert

O user name toyo por exemplo consegue deletar já o oda não consegue.

Links pesquisados por mim durante a aula:

<https://www.logicbig.com/how-to/intellij/intellij-community-edition-connecting-database.html>

Condição final deste commit relativo a senhas e endpoint

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content:

	id	admin	name	password	user_name
▶	1	1	Shimazu Toyohisa	\$2a\$10\$0AZJx1ZBobWZl2CksvP5le1JwEilvXo2oEguZHjj3FNtv0Rs1W.NW	toyo
▶	2	0	Oda Nobunaga	\$2a\$10\$0AZJx1ZBobWZl2CksvP5le1JwEilvXo2oEguZHjj3FNtv0Rs1W.NW	oda
*	NULL	NULL	NULL	NULL	NULL

\$2a\$10\$0AZJx1ZBobWZl2CksvP5le1JwEilvXo2oEguZHjj3FNtv0Rs1W.NW

GET localhost:8080/students Send Save

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

**TYPE**  
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username: oda  
Password: devdojo  
☒ Show Password

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Body Cookies (1) Headers (9) Test Results 200 OK 34 ms 600 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "content": [
3     {
4       "id": 1,
5       "name": "Eliseu",
6       "email": "eliseu@gmail"
    }
  ]
}
```

Commit: **Autenticação e autorização com SpringData com User em Banco de Dados**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/91957fb02ed366d6b67053d5a694b4e7219baf3>

## Aula 20 – Spring Security Parte 3 – Proteção das URLs com AntMatcher

[Spring Boot Essentials 19 - Spring Security pt 03 - Proteção das URLs com AntMatcher](#)

Outra forma para validar as urls e colocado versão na aplicação

Commit: **Proteção das URL com AntMatcher e Adicionado versão**

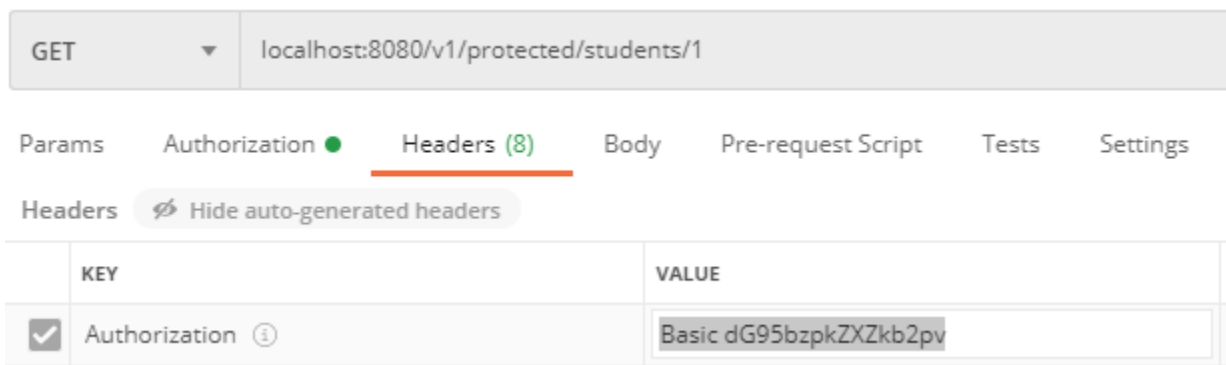
<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/46976bb505c567dcda2522d7a1f846bcc49819f8>

## Aula 21 – Exemplo de conexão com Java puro

[Spring Boot Essentials 20 - Exemplo de conexão com Java Puro](#)

Exemplo somente do get(esta conexao faz o mesmo trabalho do Postman)

8:50 como fazer para gerar a autenticação que o postman tem no Headers



Adiciona metodo

```
36 @ private static String encodeUsernamePassword(String user, String password) {  
37     String userPassword = user + ":" + password;  
38     return new String(Base64.encodeBase64(userPassword.getBytes()));  
39 }
```

e substitui

```
19     connection.setRequestProperty("Authorization", "Basic dG95bzpkZXZkb2pv");
```

Por

```
connection.setRequestProperty("Authorization", "Basic " + encodeUsernamePassword(user, password));
```

Resultado: Inprimi no terminal o usuario que tem no BD

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...  
dG95bzpkZXZkb2pv  
{ "id": 1, "name": "Eliseu", "email": "eliseu@gmail" }
```

Commit: [Adicionado conexao Java Puro metodo get #SBE20](#)

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/43cd21af037ad61d94fc0370d52dfd9119fda876>

## Aula 22 – Cliente Spring com RestTemplate – Parte 1

[Spring Boot Essentials 21 - Cliente Spring com RestTemplate pt 01 GET](#)

Conexão que faz o mesmo trabalho do Postman(busca os dados dentro do banco de dados via uma url)

Obs.: Retirado o pageble para tornar a aula mais curta

Resultado:

```
09:57:22.485 [main] DEBUG org.springframework.web.client.RestTemplate - Reading [class [Lbr.com.devdojo.model.Student;] as "application/json;charset=UTF-8"  
[Student{name='Eliseu', email='eliseu@gmail'}, Student{name='Lucas', email='lucas@gmail'}, Student{name='Katsuki3333333333', email='qualquercoisa@gmail'}]  
09:57:22.489 [main] DEBUG org.springframework.web.client.RestTemplate - Created GET request for "http://localhost:8080/v1/protected/students/"  
09:57:22.504 [main] DEBUG org.springframework.web.client.RestTemplate - Setting request Accept header to [application/json, application/*+json]  
09:57:22.654 [main] DEBUG org.springframework.web.client.RestTemplate - GET request for "http://localhost:8080/v1/protected/students/" resulted in 200 (null)  
09:57:22.655 [main] DEBUG org.springframework.web.client.RestTemplate - Reading [java.util.List<br.com.devdojo.model.Student>] as "application/json;charset=  
[Student{name='Eliseu', email='eliseu@gmail'}, Student{name='Lucas', email='lucas@gmail'}, Student{name='Katsuki3333333333', email='qualquercoisa@gmail'}]
```

Commit: [Client Spring RestTemplate method GET \\_ busca dados do BD via url #SBE21](#)

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/1792099193e2eec12222194c540d26122bd2f619>

## Aula 23 – Cliente Spring com RestTemplate – Parte 2

### [Spring Boot Essentials 22 - Cliente Spring com RestTemplate pt 02 GET Pageable](#)

Voltou o pageble

#### Result

```
10:41:06.311 [main] DEBUG org.springframework.web.client.RestTemplate - Reading [br.com.devdojo.model.PageableResponse<br.com.devdojo.model.Student>] as "application/json"
<200 OK,Page 1 of 1 containing br.com.devdojo.model.Student instances,{X-Content-Type-Options=[nosniff], X-XSS-Protection=[1; mode=block], Cache-Control=[no-
```

Commit: **Client Spring com RestTemplate GET with Pageable #SBE22**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/5464ae16c883e9b16b93f9db72a14b69133de4ca>

## Aula 24 – Cliente Spring com RestTemplate – Parte 3

### [Spring Boot Essentials 23 - Cliente Spring com RestTemplate pt 03 POST](#)

Requisição POST

Resultado 7:30

```
19:34:55.009 [main] DEBUG org.springframework.web.client.RestTemplate - Reading [class br.com.devdojo.model.Student] as "application/json;charset=UTF-8" using
<201 Created,Student{name='Jonh Wick', email='jonh@pencil.com'},{X-Content-Type-Options=[nosniff], X-XSS-Protection=[1; mode=block], Cache-Control=[no-cache,
Student{name='Jonh Wick', email='jonh@pencil.com'}
<201 Created,Student{name='Jonh Wick', email='jonh@pencil.com'},{X-Content-Type-Options=[nosniff], X-XSS-Protection=[1; mode=block], Cache-Control=[no-cache,
```

Commit parcial: **Cliente Spring com RestTemplate POST #SBE23p1**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/5f9fe680db007ceae01562aa4f5489ed9798cee5>

\$ git stash -u

Saved working directory and index state WIP on master: 5f9fe68 Cliente Spring com RestTemplate POST #SBE23p1

7:38 Organizado o código criando um ClientDAO

12:30 testa o código

13:59 alterei para 20 o numero de itens por pagina para poder mostrar mais itens no print



The screenshot shows an IDE with a project named 'devdojo'. The 'SpringBootEssentialsAdapter' class is selected in the 'config' folder. The code editor shows an override of the 'addArgumentResolvers' method. The change is to set the fallback page size to 20 in the 'PageRequest' constructor. The terminal output shows the application running and making a GET request to 'http://localhost:8080/v1/protected/students/'. The response is a 200 OK with a JSON array of three student objects: Eliseu, Lucas, and Katsuki.

```
@Override
public void addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) {
    PageableHandlerMethodArgumentResolver phmar = new PageableHandlerMethodArgumentResolver();
    phmar.setFallbackPageable(new PageRequest( page: 0, size: 20));
    argumentResolvers.add(phmar);
}
```

```
11:23:26.781 [main] DEBUG org.springframework.web.client.RestTemplate - Reading [class br.com.devdojo.model.Student] as "application/json;charset=UTF-8" using
Student{name='Eliseu', email='eliseu@gmail'}
11:23:27.036 [main] DEBUG org.springframework.web.client.RestTemplate - Created GET request for "http://localhost:8080/v1/protected/students/"
11:23:27.097 [main] DEBUG org.springframework.web.client.RestTemplate - Setting request Accept header to [application/json, application/**json]
11:23:27.211 [main] DEBUG org.springframework.web.client.RestTemplate - GET request for "http://localhost:8080/v1/protected/students/" resulted in 200 (null)
11:23:27.211 [main] DEBUG org.springframework.web.client.RestTemplate - Reading [br.com.devdojo.model.PageableResponse<br.com.devdojo.model.Student>] as "application/json"
[Student{name='Eliseu', email='eliseu@gmail'}, Student{name='Lucas', email='lucas@gmail'}, Student{name='Katsuki333333333333', email='qualquercoisa@gmail'},
Response finished with exit code 0
```

Commit: **Cliente Spring com RestTemplate POST #SBE23final**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/5bb6dfbedf054ba4e8bf4dcda35b1018275f95fa>

## Aula 25 – Cliente Spring com RestTemplate – Parte 4

### Spring Boot Essentials 24 - Cliente Spring com RestTemplate pt 03 PUT e DELETE

03:04 --> testado o update e delete

```
11:56:42.932 [main] DEBUG org.springframework.web.client.RestTemplate - Created PUT request for "http://localhost:8080/v1/admin/students/"
11:56:43.023 [main] DEBUG org.springframework.web.client.RestTemplate - Writing [Student{name='Jonh Wick 2', email='jonh@pencil.com'}] using [org.springframework
```

ID	email	name
15	jonh@pencil.com	Jonh Wick 2

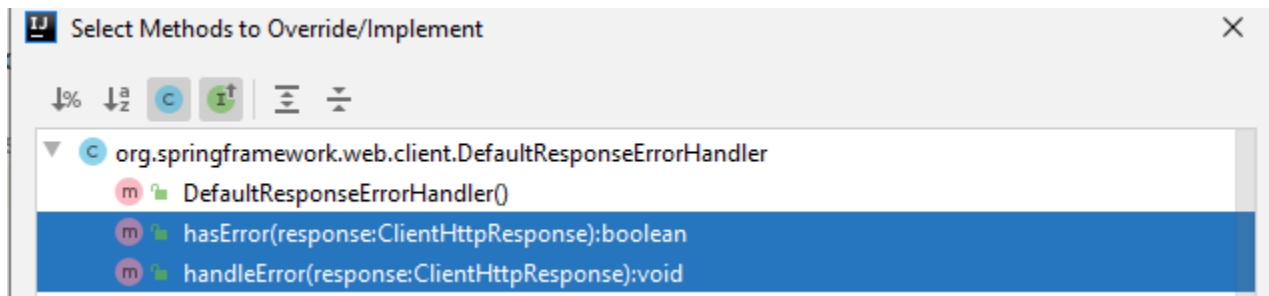
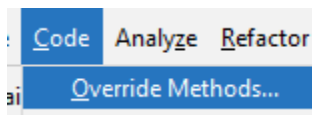
```
12:01:41.757 [main] DEBUG org.springframework.web.client.RestTemplate - Created DELETE request for "http://localhost:8080/v1/admin/students/14"
12:01:42.089 [main] DEBUG org.springframework.web.client.RestTemplate - DELETE request for "http://localhost:8080/v1/admin/students/14" resulted in 200 (null)
```

ID	email	name
13	jonh@pencil.com	Jonh Wick
15	jonh@pencil.com	Jonh Wick 2

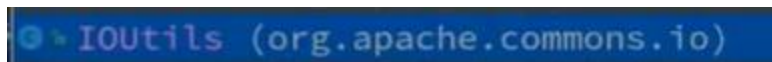
Commit parte 1: **Adicionado PUT e DELETE e CTRL+SHIFT+F #SBE24p1**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/04447671557b9946a43b361875f375074cf7589a>

04:02 --> se tiver excessao



6:50 uso do common I/O para facilitar toString



11:22 exceção lançada conforme programado

```
12:29:31.232 [main] DEBUG org.springframework.web.client.RestTemplate - Created DELETE request for "http://localhost:8080/v1/admin/students/14"
Inside hasError
12:29:31.612 [main] DEBUG org.springframework.web.client.RestTemplate - DELETE request for "http://localhost:8080/v1/admin/students/14" resulted in 404 (null)
Doing something with status code 404
Doing something with status code {"title":"Resource not found","status":404,"detail":"Student not found for ID: 14","timestamp":1605364171600,"developerMessage":"Resource not found"}
Exception in thread "main" org.springframework.web.client.HttpClientErrorException: 404 null
    at org.springframework.web.client.DefaultResponseErrorHandler.handleError(DefaultResponseErrorHandler.java:63)
    at br.com.devdojo.handler.RestResponseExceptionHandler.handleError(RestResponseExceptionHandler.java:22)
    at org.springframework.web.client.RestTemplate.handleResponse(RestTemplate.java:700)
    at org.springframework.web.client.RestTemplate.doExecute(RestTemplate.java:653)
    at org.springframework.web.client.RestTemplate.execute(RestTemplate.java:613)
    at org.springframework.web.client.RestTemplate.delete(RestTemplate.java:485)
    at br.com.devdojo.javaClient.JavaClientDAO.delete(JavaClientDAO.java:48)
    at br.com.devdojo.javaClient.JavaSpringClientTest.main(JavaSpringClientTest.java:16)

Process finished with exit code 1
```

11:57 removendo o super



```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...  
12:32:47.477 [main] DEBUG org.springframework.web.client.RestTemplate - Created DELETE request for "http://localhost:8080/v1/admin/students/14"  
Inside hasError  
12:32:47.720 [main] DEBUG org.springframework.web.client.RestTemplate - DELETE request for "http://localhost:8080/v1/admin/students/14" resulted in 404 (null)  
Doing something with status code 404  
Doing something with status code {"title":"Resource not found","status":404,"detail":"Student not found for ID: 14","timestamp":1605364367710,"developerMessage":null}
```

Commit: **Tratado as exceções de Put e Delete #SBE24final**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/27aa8007567e61360095ef8b398f4545c88ae1e5>

## Aula 26 – Testes unitários para o repositório com @DataJpaTest

[Spring Boot Essentials 25 - Testes Unitários para o Repositório com @DataJpaTest](#)

Testando com banco em memoria

```
<dependency>  
  <groupId>com.h2database</groupId>  
  <artifactId>h2</artifactId>  
  <scope>test</scope>  
</dependency>
```

7:48 Importa staticamente o Assert.that

12:07 metodo para testar o findByNameIgnoreCaseContaining

16:57 metodo para testar se email não é vazio

20:05 criando testes para o proprio banco de dados

Somente um teste falho porque foi definido um numero de dados para o banco em memoria mas o banco real tem uma quantidade diferente de dados. Ficou comentado o comando

```
//@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
```

Commit **Teste unitario para o repositorio com DataJpaTest #SBE25**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/26de72865f659ac4f4c5a14999590d0fded1f858>

## Aula 27 – Atualizando Springboot e resolvendo bugs de importação

[Spring Boot Essentials 26 - Atualizando Springboot e resolvendo bugs de importação](#)

Somente alterei da versao 1.5.3 para 1.5.4 mas não deu nenhum erro no programa

Commit: **Alterado versao do SpringBoot #SBE26**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/a90c8cca0e126eff562f0e68333ce68064be7d80>

Acabou alterando no proximo commit um monte de coisa em função desta troca de versao

## Aula 28 – Teste de integração com MockMvc, Mockito e AsserJ – Parte 1

[Spring Boot Essentials 27 - Testes de integração com MockMvc, Mockito e AssertJ pt 01](#)

6:54 Primeiro test:

`listStudentWhenUsernameAndPasswordAreIncorrectShouldReturnStatusCode401()`

10:30 Segundo test

`getStudentByIdWhenUsernameAndPasswordAreIncorrectShouldReturnStatusCode401()`

11:42 Terceiro test

`listStudentsWhenUsernameAndPasswordAreCorrectShouldReturnStatusCode200()`

16:29 Quarto test

`getStudentByIdWhenUsernameAndPasswordAreCorrectShouldReturnStatusCode200()`

19:29 Quinto Test

Commit: [Adicionado Testes de Integração Endpoints com MockMvc, Mockito e Assert #SBE27](#)

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/6768ed01ef9fdd8ba0379a09d2af870d49180ebb>

Acabou alterando um monte de coisa em função da troca de versão do spring que fiz no commit anterior

## Aula 29 – Teste de integração com MockMvc, Mockito e AssertJ – Parte 2

[Spring Boot Essentials 28 - Testes de integração com MockMvc, Mockito e AssertJ pt 02](#)

4:40 Adicionado Before

6:35 Novo test

`deleteWhenUserHasRoleAdminAndStudentDoesNotExistsShouldReturnStatusCode404()`

8:38 Usando o MockMVC

11:48 importação para uso do @WithMocker

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-test</artifactId>
  <version>4.2.3.RELEASE</version>
  <scope>test</scope>
</dependency>
```

16:23 Novo test

`DeleteWhenUserDoesNotHaveHasRoleAdminReturnStatusCode403`

17:48 Novo test para created

`createWhenNameIsNullShoudReturnStatusCode400BadRequest()`

23:16 `createShouldPersistDataAndReturnStatusCode201()`

Commit **Adicionado mais Testes de Integração #SBE28**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/ea1408f7911e131fe267a5e0ca6524de1d2496d0>

## Aula 30 – Executando Springboot como serviço

[Spring Boot Essentials 29 - Executando Springboot como serviço Init.d](#)

Rodar com serviço no Linux. Então não fiz esta alteração

Também tem uma diferença que o meu pom --> build esta no org.apache.maven e o do devdojo esta com org.springframework.boot

Isto provavelmente se deve que o curso usou o IntelliJ pago e eu usei o community para gerar o start.

## Aula 31 – Spring security – Parte 4

[Spring Boot Essentials 30 - Spring security pt 04 - JWTToken Authentication](#)

Alterado de autenticação httpbasicAuthentication para Token

Com token tem authentication diz quem você é e authorization diz oque vc pode fazer

1:24 cria classe JWTAuthenticationFilter

5:30 adicionar gerador de token no pom

```
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt</artifactId>
  <version>0.7.0</version>
</dependency>
```

7:00 cria classe SecurityConstants

15:30 Classe para autorização

23:15 Modificações no securityconfig

25:20 testes no postman

<https://andreybleme.com/2017-04-01/autenticacao-com-jwt-no-spring-boot/>

Referencia de um mao na massa bem semelhante ao usado aqui

Obs.: SecurityConfig.java toda vez que a aplicação sobe gera um codigo no Header do Response no Postman

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(customUserDetailsService).passwordEncoder(new
BCryptPasswordEncoder());
}
```

\$2a\$10\$ppFmHDeJyoro8FM1iXZI2u5Z6b5t4XVwLk.hHy9GrjhNMSzbeDqii

\$2a\$10\$7D.4V9JtvYn5PG7XwWKos.7fucATYAzibZsty30z86BN1h5CRz9TW

POST

http://localhost:8080/login

Send

Save

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookiesCode

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

BodyCookies (1)Headers (9)Test Results

200 OK273 ms431 BSave Response

KEY	VALUE
X-Content-Type-Options	nosniff
X-XSS-Protection	1; mode=block
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Pragma	no-cache
Expires	0
X-Frame-Options	DENY
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0b3lviwiZXhwIjoxNjA2NjUwODg1fQ.aTh_7OP8AxQmImQJEEhqbGIBuQb5vEPIUPAUUKxnEgenj3kC6a7bXmLb0gLnU4wYV76Kr5o-VXjssoPj1gtDIQ
Content-Length	
Date	

Para fazer o get localhost:8080/v1/protected/students/

ParamsAuthorizationHeaders (8)

Headers7 hidden

Clica em Headers no Request e hidden os Headers

Em seguida acrescenta o um Auth clicando em cima do Key

ParamsAuthorizationHeaders (7)BodyPre-i

Headers7 hidden

KEY	VALUE
Key	Value

+ Digita Au

KEY
<input checked="" type="checkbox"/> Au
Authorization
Proxy-Authorization

e seleciona Authorization + Cola o Value

Params	Authorization	Headers (8)	Body	Pre-request Script	Tests	Settings
Headers <span>7 hidden</span>						
	KEY	VALUE	DESCRIPTION			
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJvZGEiLCJleHAiOiJlMDY2NTI5Mzd9.WpYoQmWetIz2XQ8KAYkZlPJPRsthzOP8JxS2sANp6c4h9wD8_I-R-6CxoHzPXPvsOCCjUdFK3kUD-Wv34xMtA				
	Key					

USER --> {"username": "oda", "password": "devdojo"}

Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJvZGEiLCJleHAiOiJlMDY2NTI5Mzd9.WpYoQmWetIz2XQ8KAYkZlPJPRsthzOP8JxS2sANp6c4h9wD8\_I-R-6CxoHzPXPvsOCCjUdFK3kUD-Wv34xMtA

ADMIN --> {"username": "toyo", "password": "devdojo"}

Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJvZGEiLCJleHAiOiJlMDY2NTI5Mzd9.WpYoQmWetIz2XQ8KAYkZlPJPRsthzOP8JxS2sANp6c4h9wD8\_I-R-6CxoHzPXPvsOCCjUdFK3kUD-Wv34xMtA

Resultado OK

Get Todos

Examples 0 BUILD

GET localhost:8080/v1/protected/students/ Send Save

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

Headers 7 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWliOiJvZGEiLCJleHAiOiJlMDY2NTI5Mzd9.WpYoQmWetIz2XQ8KAYkZlPJPRsthzOP8JxS2sANp6c4h9wD8_I-R-6CxoHzPXPvsOCCjUdFK3kUD-Wv34xMtA				
	Key	Value	Description			

Body Cookies (1) Headers (9) Test Results 200 OK 62 ms 1.07 KB Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "content": [
3     {
4       "id": 1,
5       "name": "Eliseu",
6       "email": "eliseu@gmail"
7     }
8   ]
9 }

```

Em 28/11/2020 deu certo todos os comandos http com USER E ADMIN(com as respostas adequadas de cada um)

Commit **Adicionado Spring security JWTToken Authentication #SBE30**

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/43a0c2e20bfd25012d288eed4ec2f112b0d22216>

Obs.: tive bastante dificuldade de gerar o token em função de ter sido criada uma coluna extra no banco. Quem me ajudou foi o Guilherme da mentoria do maisprati.

Result Grid					
	nin	name	password	user_name	username
		Shimazu Toyohisa	\$2a\$10\$0AZJx1ZBobWZl2CksvP5le1JwEiIvXo2oEguZHjj3FNtv0Rs1W.NW	toyo	toyo
		Oda Nobunaga	\$2a\$10\$0AZJx1ZBobWZl2CksvP5le1JwEiIvXo2oEguZHjj3FNtv0Rs1W.NW	oda	
		NULL	NULL	NULL	NULL

NA coluna username eu precisei adicionar o toyo para conseguir funcionar. Vou acrescentar o oda também. Já a coluna usernameame não sei porque existe.

## Aula 32 – Spring security – Parte 5

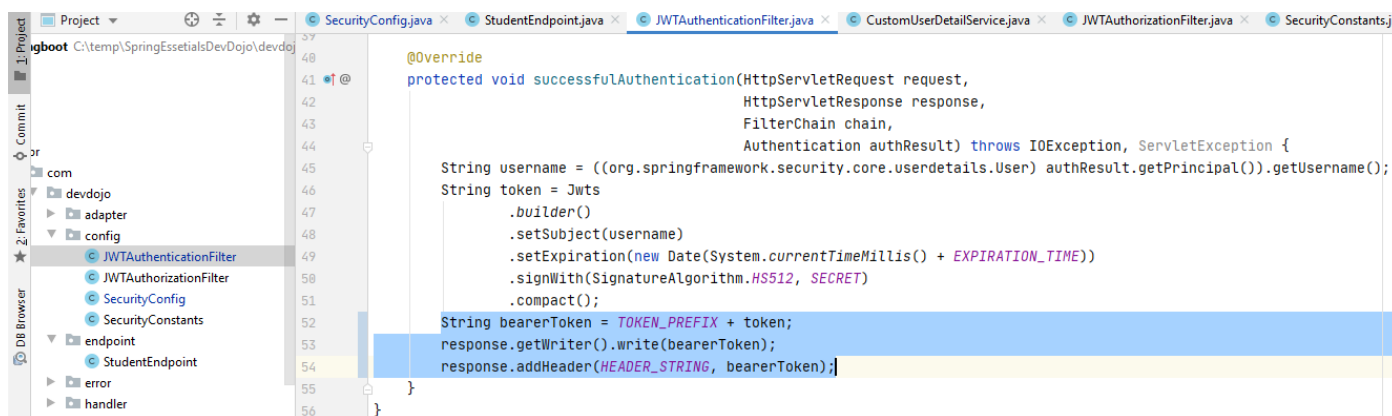
### Spring Boot Essentials 31 - Spring security pt 05 - CORS

3 coisas

1) CORS – Cross Origin Resource Sharing: é uma questão de segurança dos navegadores

Alteração no programa é para permitir requisições que não estejam no mesmo servidor, como por exemplo se a aplicação front-end for feita em javascript.

2) 3:25 Facilitando a vida de quem faz front-end para pega o header



login JWTToken Protected Role User oda

POST http://localhost:8080/login

Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (9) Test Results

Status: 200 OK Time: 4.54 s Size: 586 B

Pretty Raw Preview Visualize Text

```

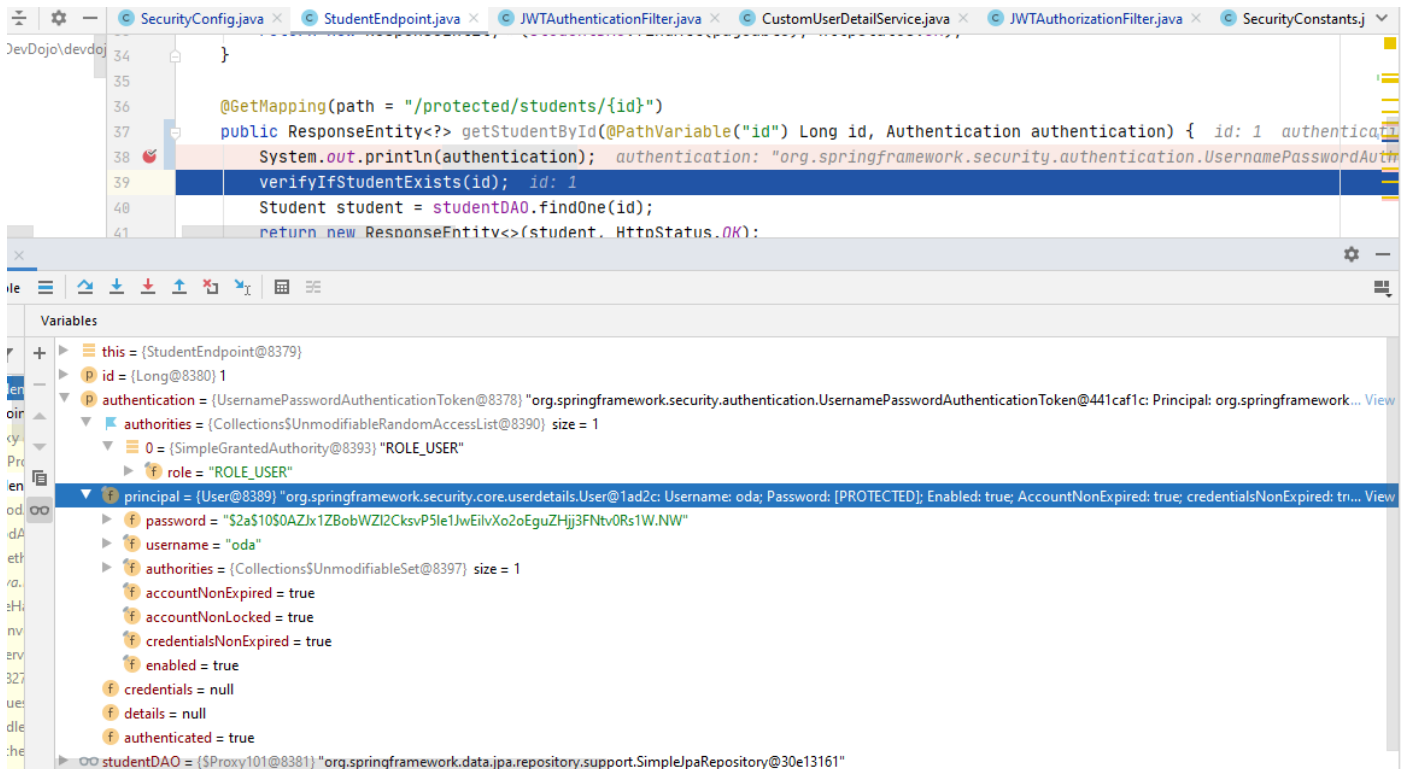
1 Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJvZGEiLCJleHAiOiJlMjMDY2NTcwOTR9.2QVcPTV90OHVJv0N_JoDjuQPhI5AdJT3DN71by05xC6cqkE7VBtL-VzrrTQx1zpaVxxnddROBNmf18Vq4N1FjQ

```

Agora o Token também vem no Body do response

3) 5:00 Como pegar o usuário que temos no token

Alterado codigo e resultado abaixo



\$ git commit -m "Melhorias incluindo CORS, token no Body Response e usuarios no token"

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/c6e47673092463a3b850b711958fe12e3e220003>

## Aula 33 – Refatoração dos testes para executar com Token

### Spring Boot Essentials 32 - Refatoração dos testes para executar com Token

0:37 Copiado a classe StudentEndPointTest para StudentEndPointTokenTest

1:00 apagado @TestConfiguration

1:16 Acrescentado token no header adicionando atributos + 2:05 criado metodos

4:44 Alterado os testes

\$ git commit -m "Adicionado/Refatorado os testes para Token security"

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/56cfb40d1a5c8f5b677370f70e2cfe984dbb25f0>

## Spring Boot Essentials 33 - Documentation with Swagger2 and SpringFox

### Spring Boot Essentials 33 - Documentation with Swagger2 and SpringFox

0:57 Adiciona as dependencias no pom

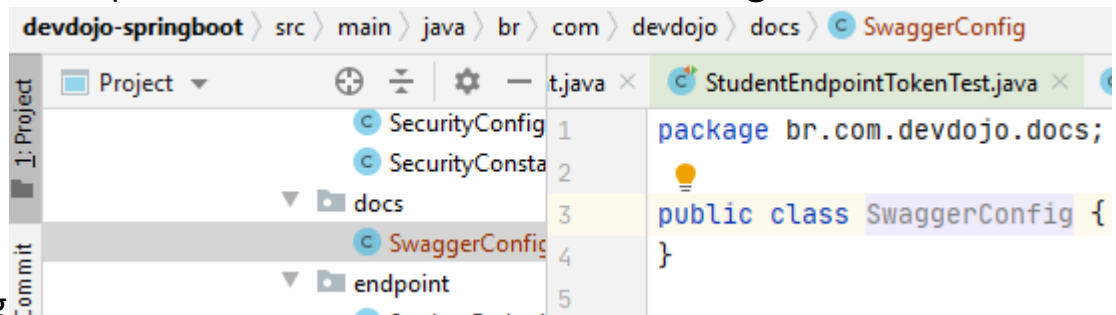
```
<dependency>
  <groupId>io.springfox</groupId>
```

```

        <artifactId>springfox-swagger2</artifactId>
        <version>2.7.0</version>
    </dependency>
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
        <version>2.7.0</version>
    </dependency>

```

2:00 Cria um novo pacote chamado docs e uma configuração chamada



SwaggerConfig

4:55 usa browser em <http://localhost:8080/v2/api-docs> que mostra um json que sera transformado em <http://localhost:8080/swagger-ui.html>

5:38 adiciona configurações através de metodo

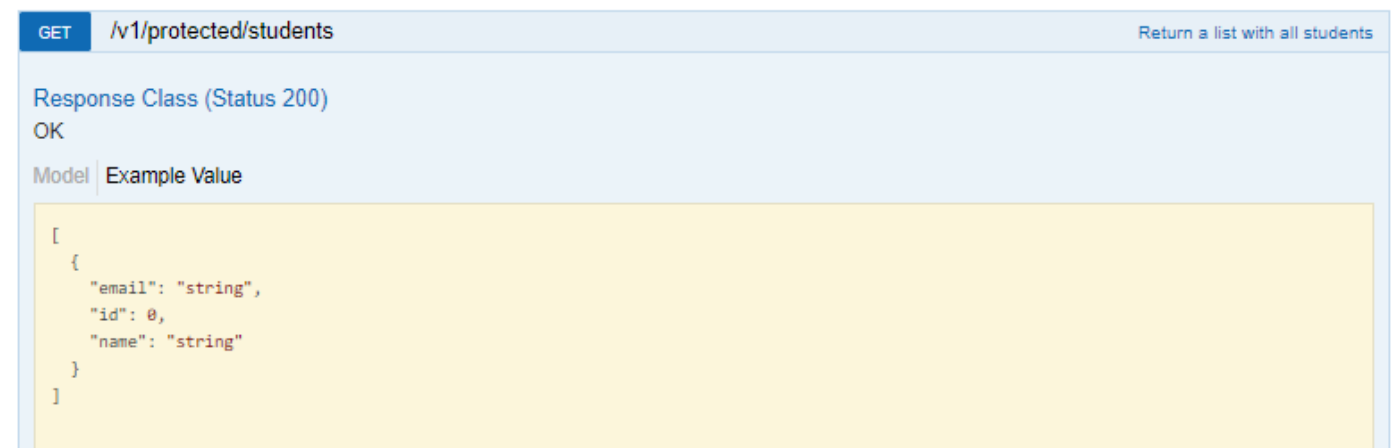
6:35 cuidado na hora de importar o contact ( tem que ser `import springfox.documentation.service.Contact` 😊)

8:26 adicionando informação ao getAll com anotações em StudentEndPoint

```

@GetMapping(path = "/protected/students")
@ApiOperation(value = "Return a list with all students", response= Student[].class)
public ResponseEntity<?> listAll(Pageable pageable) {
    System.out.println(studentDAO.findAll(pageable));
    return new ResponseEntity<>(studentDAO.findAll(pageable), HttpStatus.OK);
}

```



10:00 Adicionado Token para poder dar um TryOut

Com anotações `@ApiOperation` and `@ApiImplicitParam(s)`



```

@GetMapping(path = "/protected/students")
@ApiOperation(value = "Return a list with all students", response = Student[].class)
@ApiImplicitParams({
    @ApiImplicitParam(name = "Authorization", value = "Bearer token",
        required = true, dataType = "string", paramType = "header")
})
public ResponseEntity<?> listAll(Pageable pageable) {
    System.out.println(studentDAO.findAll(pageable));
    return new ResponseEntity<>(studentDAO.findAll(pageable), HttpStatus.OK);
}

```

Com isto é necessário colocar uma authorization no swagger

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	7VBtL-VzrrTQxlzpaVxxnddROBNmf18Vq4NIFjQ	Bearer token	header	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

-->

Response Body	
<pre> {   "content": [     {       "id": 1,       "name": "Eliseu",       "email": "eliseu@gmail"     },     {       "id": 2,       "name": "Lucas",       "email": "lucas@gmail"     }   ], } </pre>	

12:00 "Sem condições de fazer isto em todos os metodos" então vai SwaggerConfiguration e adiciona

```

.globalOperationParameters(Collections.singletonList(new ParameterBuilder()
    .name("Authorization")
    .description("Bearer token")
    .modelRef(new ModelRef("string"))
    .parameterType("header")
    .required(true)
    .build()))

```

```

@ApiImplicitParams({
    @ApiImplicitParam(name = "Authorization", value = "Bearer token",
        required = true, dataType = "string", paramType = "header")
})

```

E apaga o  
da configuração individual

Resultado: Authorization em todos os metodos que solicita. Exemplo do PUT

Parameter	Value	Description
Authorization	<input type="text" value="(required)"/>	Bearer token
student	<input type="text" value="(required)"/>	student

\$ git commit -m "Adicionado documentação da API com Swagger"

<https://github.com/eliseusbrito/SpringEssetialsDevDojo/commit/5d11adaaf2ed9c50d49a6ab8e993e10325f79cad>