



11^a
Emisión

DIPLOMADO
Desarrollo de Sistemas
con Tecnología Java

Módulo 7
Persistencia con Spring Data

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC



Educación
Continua
1971 - 2021

REST

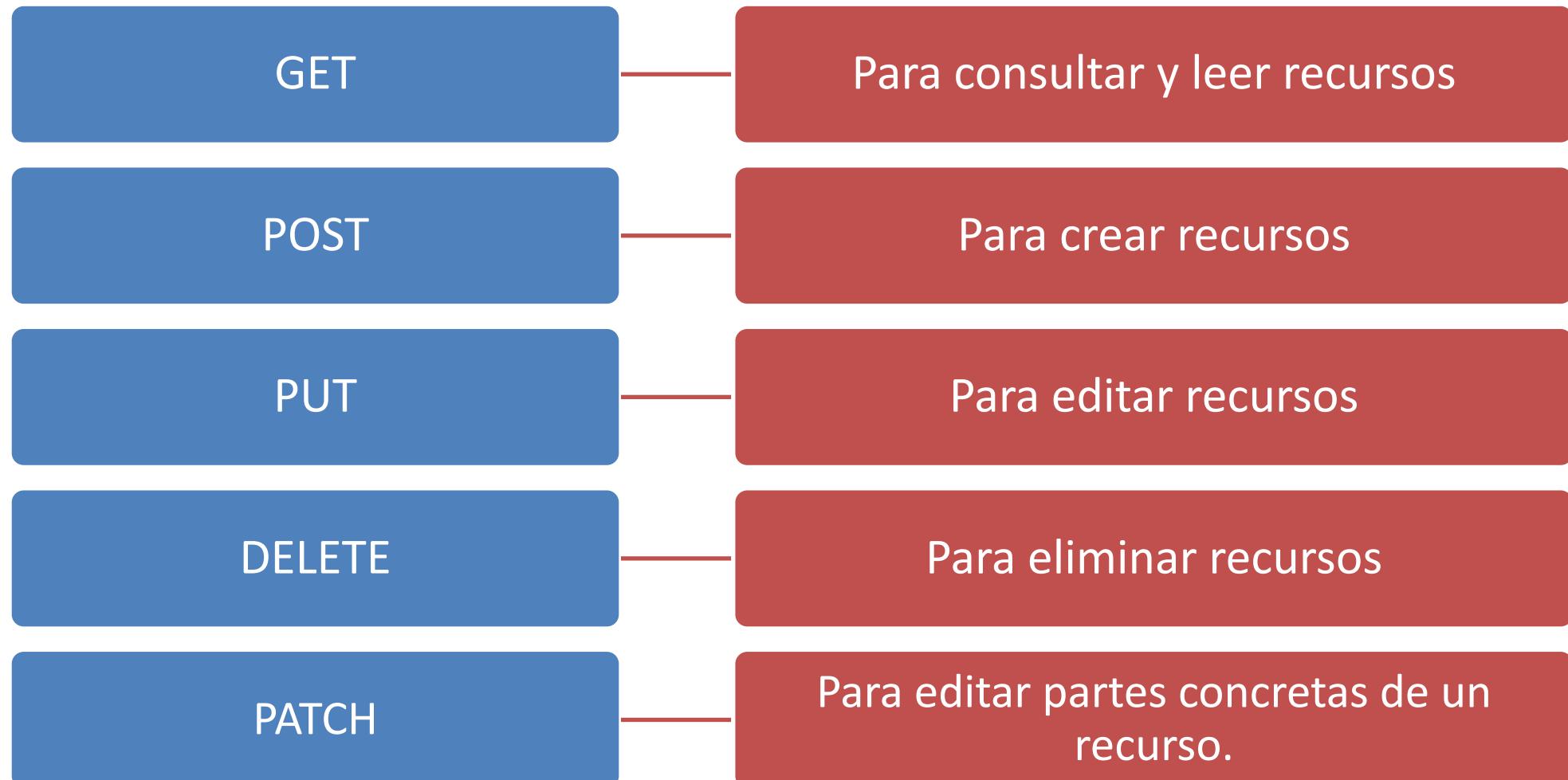
- REST, REpresentational State Transfer, es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP.
- REST permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP.



REST

- Características
 - Independencia de plataforma.
 - Independencia de lenguaje de programación.
 - Basado en estándares.
 - Puede ser usado fácilmente en presencia de firewall.
 - Protocolo cliente/servidor sin estado
 - Sólo utiliza HTTP

Verbos HTTP



Códigos de estado HTTP

HTTP Status Codes

Level 200 (Success)

200 : OK

201 : Created

**203 : Non-Authoritative
Information**

204 : No Content

Level 400

400 : Bad Request

401 : Unauthorized

403 : Forbidden

404 : Not Found

409 : Conflict

Level 500

500 : Internal Server Error

503 : Service Unavailable

501 : Not Implemented

504 : Gateway Timeout

599 : Network timeout

502 : Bad Gateway



Spring Data REST

- Spring Data REST es parte del proyecto general Spring Data y facilita la creación de servicios web REST sobre los repositorios de Spring Data.
- Se basa en dichos repositorios, analiza el modelo de dominio que se tenga definido en las entidades y expone estos recursos HTTP controlados por hipermedia, es decir, bajo la arquitectura HATEOAS (Hypermedia as the Engine of Application State)

Spring Data REST

- Aprovecha los standards HYPERMEDIA & Internet
- HAL (Hypertext Application Language,)
- ALPS (Application-Level Profile Semantics)
- JSON Schema
- URI Templates (RFC 6570) text/uri-list mediatype (RFC 2483)
- profile link relation (RFC 6906)



Json

```
{  
    "matricula": "7B",  
    "nombre": "Gema",  
    "paterno": null,  
    "fnac": "2001-03-20T06:00:00.000+00:00",  
    "estatura": 1.53,  
    "calificaciones": [  
        {  
            "id": 7,  
            "materia": "BDII",  
            "calificacion": 8  
        },  
        {  
            "id": 8,  
            "materia": "POO",  
            "calificacion": 5  
        }  
    ]  
}
```

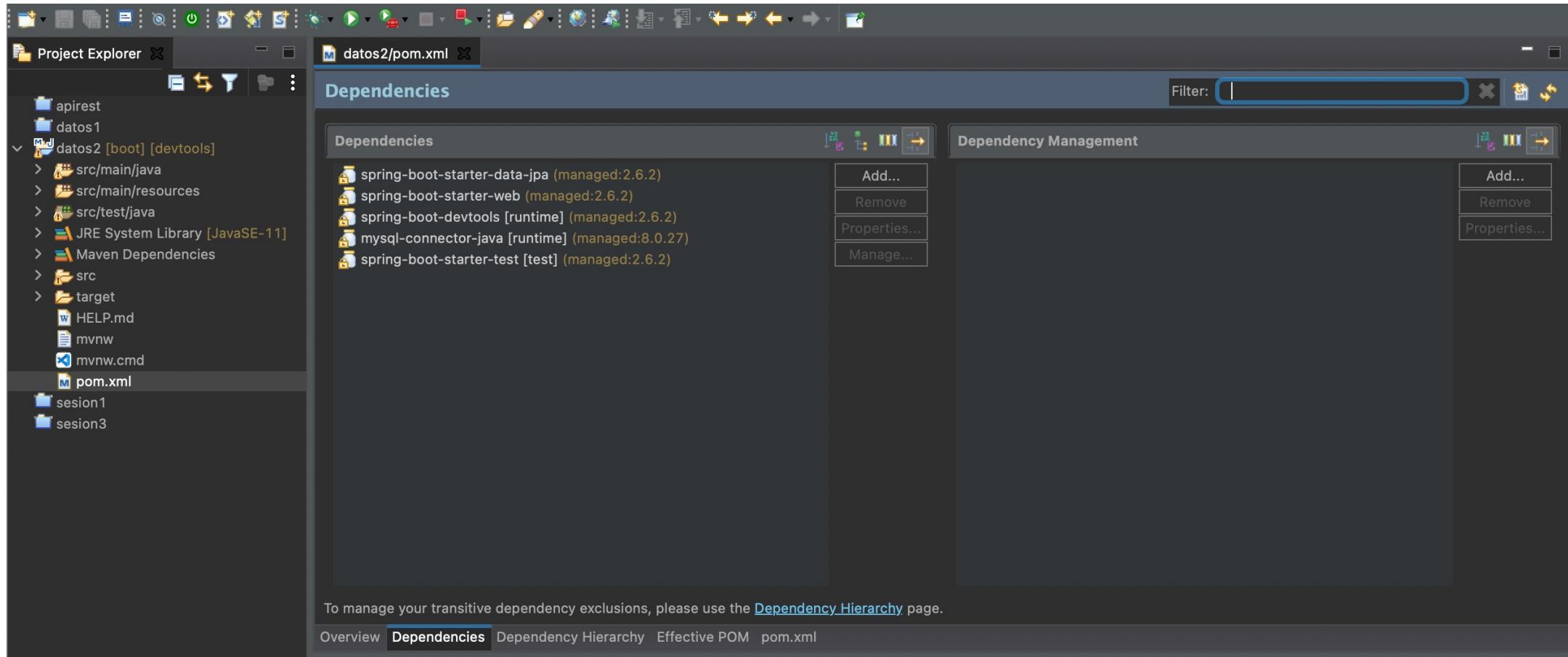


pom.xml

The screenshot shows a Java development environment with the following details:

- Project Explorer:** Shows the project structure with a folder named "datos2 [boot] [devtools]" containing "src/main/java", "src/main/resources", "src/test/java", and "pom.xml". Other folders like "apirest" and "datos1" are also listed.
- Code Editor:** Displays the content of the "pom.xml" file. The code is color-coded, with blue highlighting for XML tags like <groupId>, <artifactId>, <version>, <dependency>, etc., and black for text content.
- File Content:** The pom.xml file defines a parent dependency on "spring-boot-starter-parent" version 2.6.2, a group ID of "mx.unam.dgtic", an artifact ID of "datos1", and a version of "0.0.1-SNAPSHOT". It includes a description of the project as "Proyecto de Practica M7". The dependencies section lists "org.springframework.boot" as the group ID and "spring-boot-starter-data-jpa" as the artifact ID.
- Bottom Navigation:** Includes tabs for Overview, Dependencies, Dependency Hierarchy, Effective POM, and a selected tab labeled "pom.xml".
- Bottom Status:** Shows "0 errors, 45 warnings, 0 others".

pom.xml



pom.xml

<https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-rest>

 **Spring Boot Starter Data REST » 2.6.2**

Starter for exposing Spring Data repositories over REST using Spring Data REST

License	Apache 2.0
Organization	Pivotal Software, Inc.
HomePage	https://spring.io/projects/spring-boot
Date	(Dec 22, 2021)
Files	pom (2 KB) jar (4 KB) View All
Repositories	Central
Used By	228 artifacts

[Maven](#) [Gradle](#) [Gradle \(Short\)](#) [Gradle \(Kotlin\)](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-rest -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
    <version>2.6.2</version>
</dependency>
```

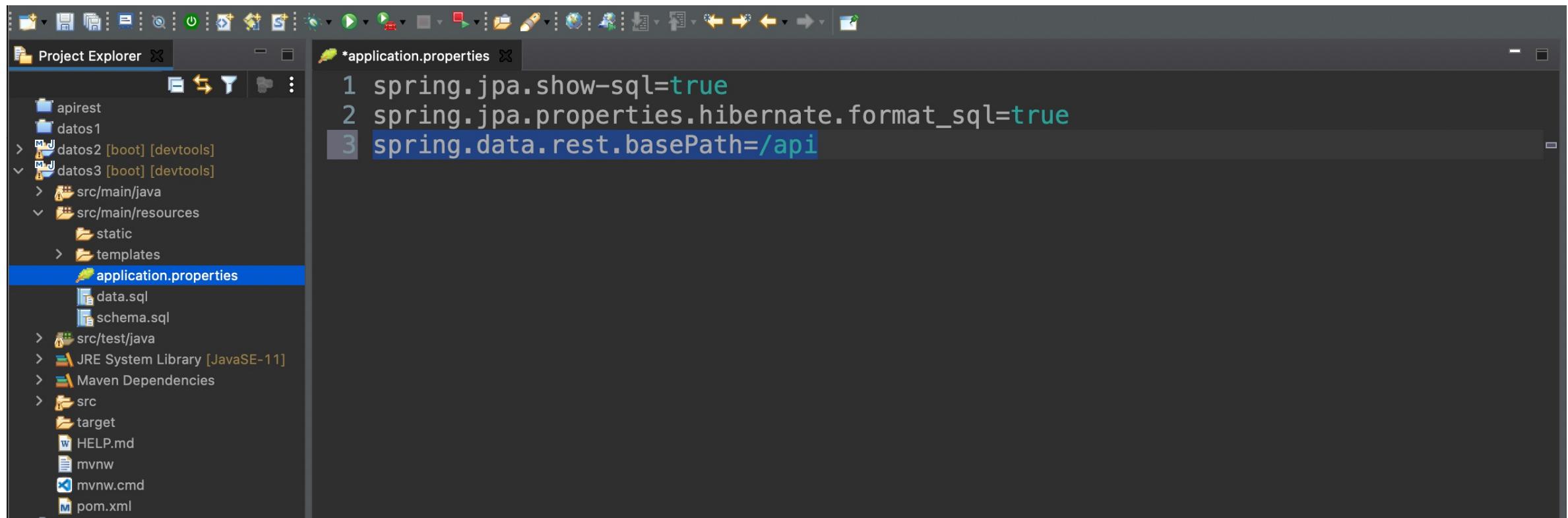
Include comment with link to declaration

pom.xml

```
<!--  
https://mvnrepository.com/artifact/org.springframework.boot/spring-  
boot-starter-data-rest -->  
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-data-rest</artifactId>  
</dependency>
```



application.properties



The screenshot shows a Java development environment with the following details:

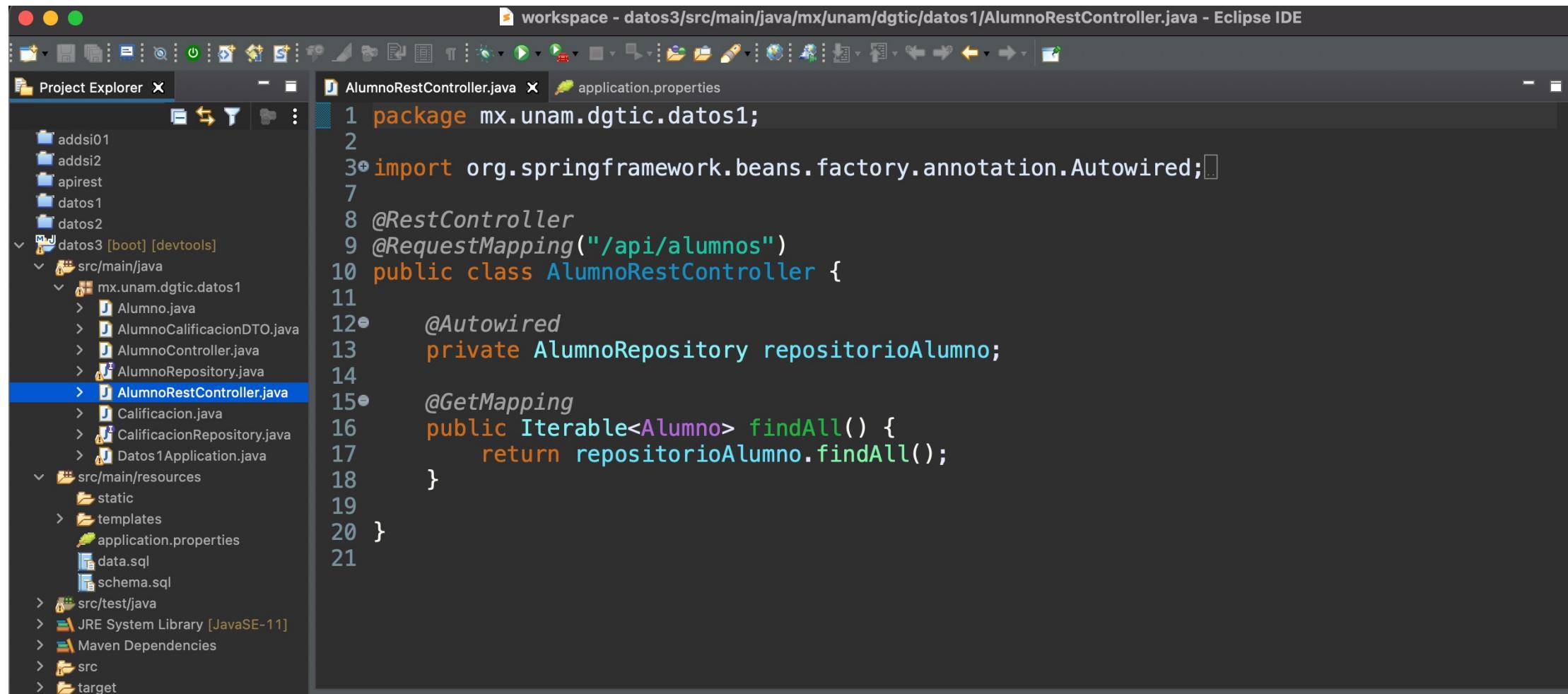
- Project Explorer:** Shows the project structure with several modules like apirest, datos1, datos2 [boot] [devtools], and datos3 [boot] [devtools]. The application.properties file is selected in the Project Explorer.
- Code Editor:** Displays the contents of the application.properties file:

```
1 spring.jpa.show-sql=true
2 spring.jpa.properties.hibernate.format_sql=true
3 spring.data.restbasePath=/api
```

@RestController

- La anotación `@RestController` permite que los métodos del controlador se expongan como API RESTful
- Es una versión especializada del controlador.
- Incluye las anotaciones `@Controller` y `@ResponseBody` y simplifica la implementación del controlador

AlumnoRestController



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - datos3/src/main/java/mx/unam/dgtic/datos1/AlumnoRestController.java - Eclipse IDE
- Project Explorer:** Shows the project structure under "datos3 [boot] [devtools]".
 - src/main/java:
 - mx.unam.dgtic.datos1
 - Alumno.java
 - AlumnoCalificacionDTO.java
 - AlumnoController.java
 - AlumnoRepository.java
 - AlumnoRestController.java (selected)
 - Calificacion.java
 - CalificacionRepository.java
 - Datos1Application.java
 - src/main/resources
 - static
 - templates
 - application.properties
 - data.sql
 - schema.sql
 - src/test/java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
- Editor:** Displays the AlumnoRestController.java code in the mx.unam.dgtic.datos1 package.

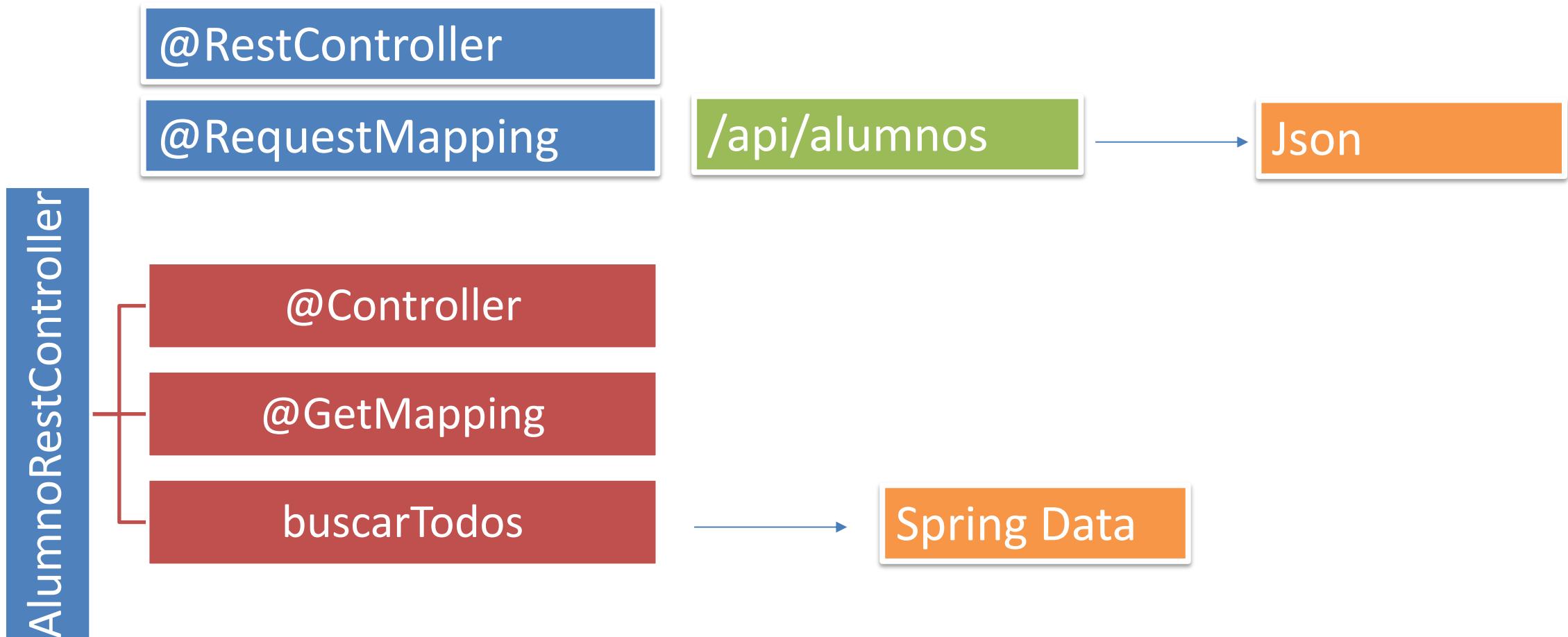
```
1 package mx.unam.dgtic.datos1;
2
3+ import org.springframework.beans.factory.annotation.Autowired;■
4
5 @RestController
6 @RequestMapping("/api/alumnos")
7 public class AlumnoRestController {
8
9     @Autowired
10    private AlumnoRepository repositorioAlumno;
11
12    @GetMapping
13    public Iterable<Alumno> findAll() {
14        return repositorioAlumno.findAll();
15    }
16
17
18}
19
20}
21
```

http://localhost:8080/api/alumnos

The screenshot shows the Postman application interface. At the top, the URL `http://localhost:8080/api/alumnos` is entered into the address bar. Below it, a dropdown menu shows the method as `GET`. To the right of the URL, there are buttons for `Save`, `Edit`, and `Send`. A blue `Send` button is highlighted. On the left, tabs for `Params`, `Auth`, `Headers (6)`, `Body`, `Pre-req.`, `Tests`, and `Settings` are visible, with `Params` being the active tab. On the right, a `Cookies` tab is also visible. Under the `Params` tab, there is a section for `Query Params` with a table header: `KEY`, `VALUE`, `DESCRIPTION`, and `Bulk Edit`. Below this is a table row with columns for `Key`, `Value`, and `Description`. Under the `Body` tab, the response is displayed in a code editor-like view. The response status is `200 OK` with `1414 ms` and `2.37 KB`. The response body is a JSON object:

```
17 },  
18 {  
19   "matricula": "2A",  
20   "nombre": "Nadia",  
21   "paterno": "Perez",  
22   "fnac": "2001-01-10T06:00:00.000+00:00",  
23   "estatura": 1.56,  
24   "calificaciones": [  
25     {  
26       "id": 1,  
27       "materia": "BD",  
28       "calificacion": 10  
29     },  
30     {  
31       "id": 2,
```

Spring REST GET



pom.xml

<https://mvnrepository.com/artifact/io.rest-assured/rest-assured>

 REST Assured » 4.4.0

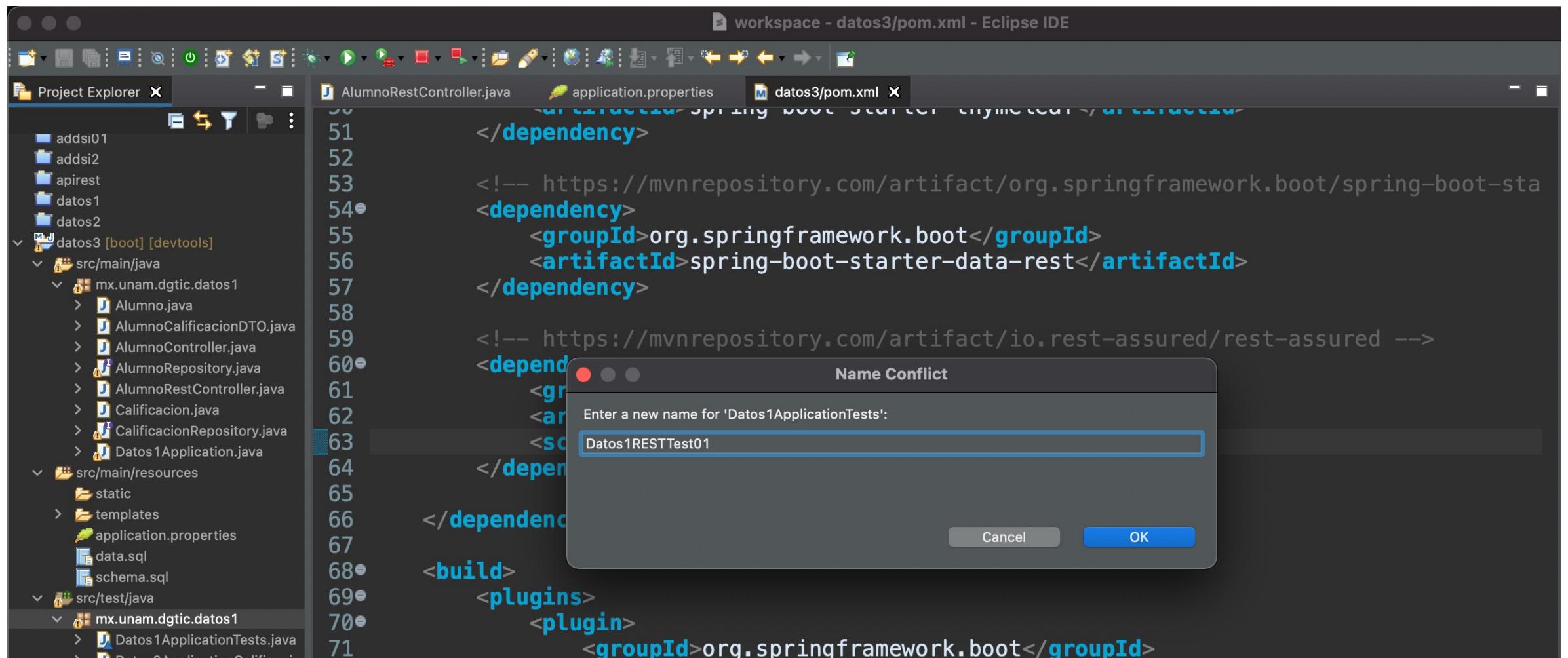
Java DSL for easy testing of REST services

License	Apache 2.0
Categories	Testing Frameworks
HomePage	http://code.google.com/p/rest-assured
Date	(May 23, 2021)
Files	bundle (680 KB) View All
Repositories	Central
Used By	1,803 artifacts
Vulnerabilities	Vulnerabilities from dependencies: CVE-2019-10172

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>4.4.0</version>
    <scope>test</scope>
</dependency>
```

Agregar nueva prueba

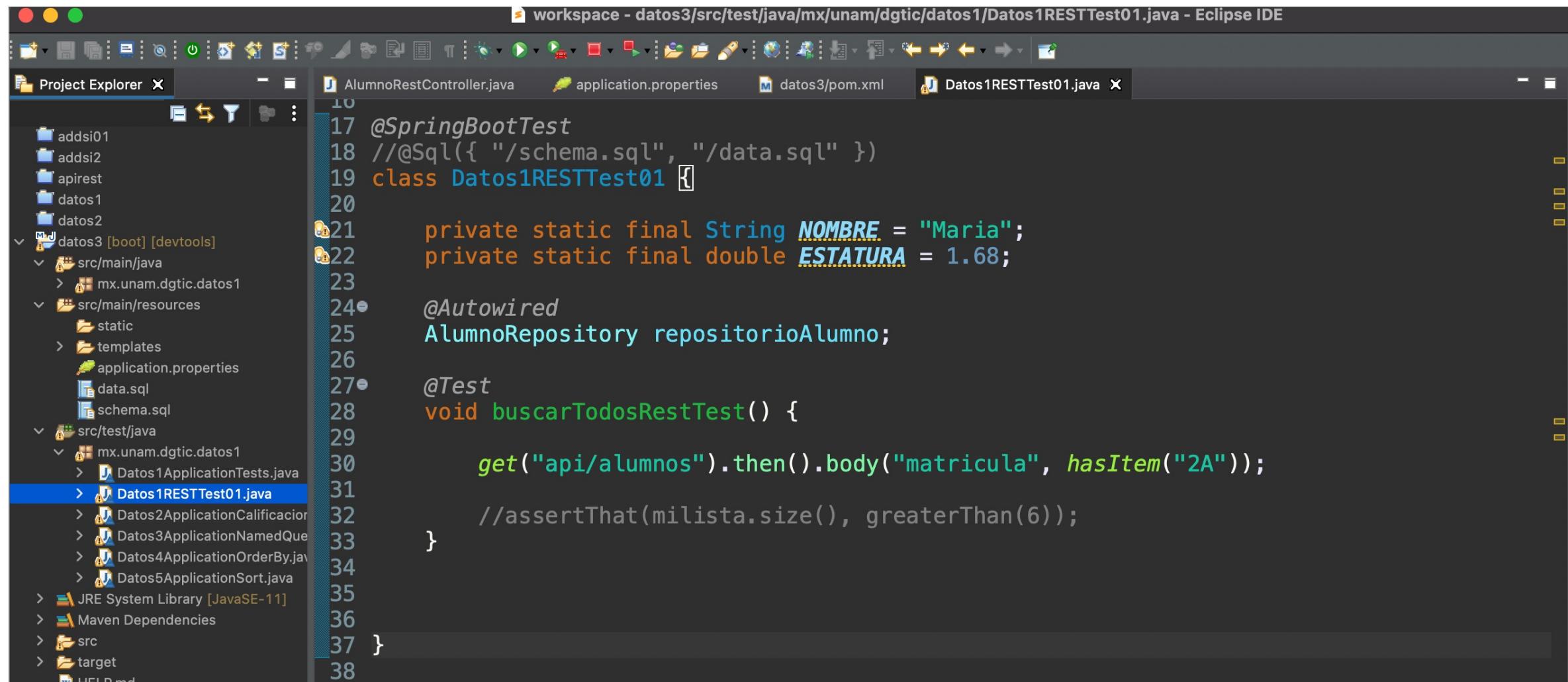


The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows several Java projects: addssi01, addssi2, apirest, datos1, datos2, and datos3 [boot] [devtools]. The datos3 project is expanded, showing its structure under src/main/java and src/test/java.
- Open Editors:** The pom.xml file is open in the main editor area, displaying Spring Boot configuration code.
- Dialog Box:** A modal dialog titled "Name Conflict" is displayed, prompting the user to "Enter a new name for 'Datos1ApplicationTests':". The input field contains the value "Datos1RESTTest01".
- Code Preview:** Below the dialog, a preview of the pom.xml code shows the addition of a new dependency for the REST test class.

```
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-rest -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <scope>test</scope>
</dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
```

Lanzar la aplicación y Lanzar la Prueba Unitaria

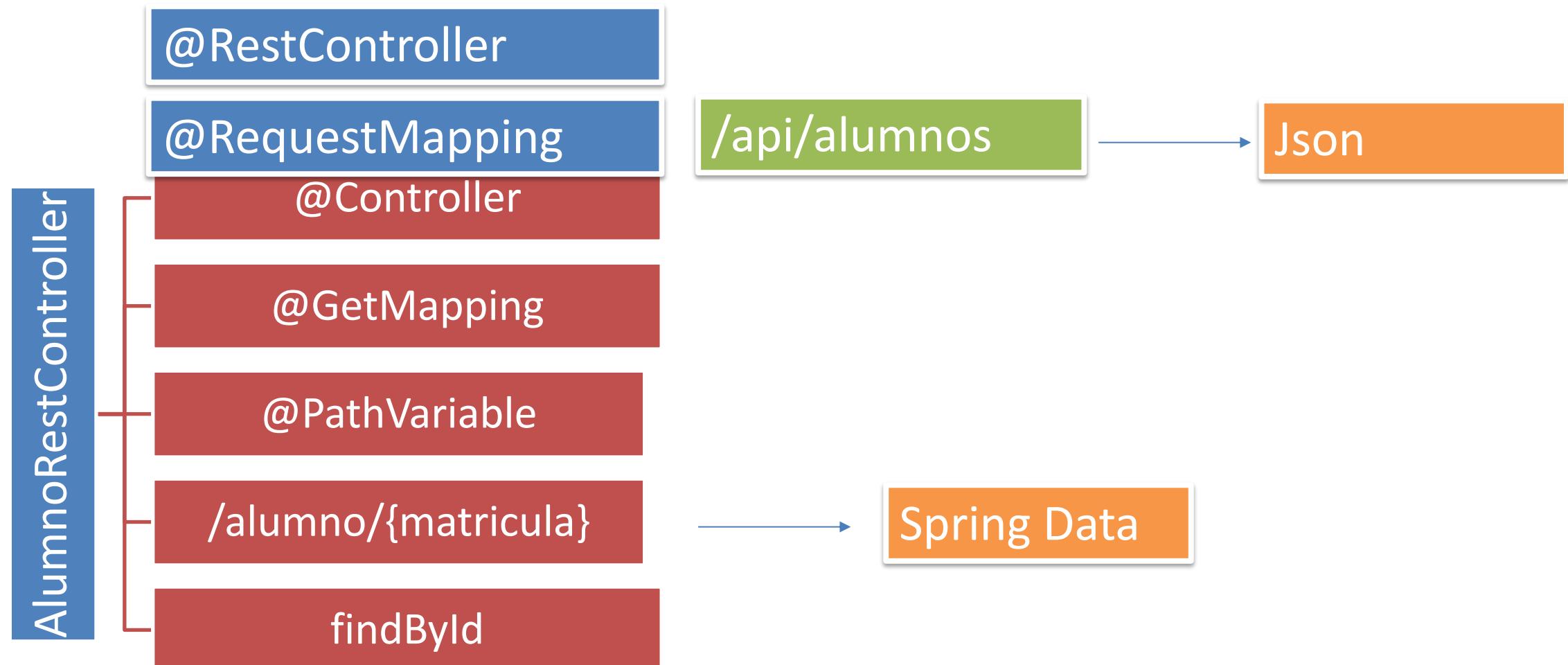


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - datos3/src/test/java/mx/unam/dgtic/datos1/Datos1RESTTest01.java - Eclipse IDE
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and navigation.
- Project Explorer:** Shows the project structure under "datos3 [boot] [devtools]".
 - src/main/java: mx.unam.dgtic.datos1 (containing AlumnoRestController.java), static, templates, application.properties, data.sql, schema.sql.
 - src/main/resources: static, templates, application.properties, data.sql, schema.sql.
 - src/test/java: mx.unam.dgtic.datos1 (containing Datos1ApplicationTests.java, Datos1RESTTest01.java, Datos2ApplicationCalificacion.java, Datos3ApplicationNamedQuery.java, Datos4ApplicationOrderBy.java, Datos5ApplicationSort.java), JRE System Library [JavaSE-11], Maven Dependencies, src, target, HELP.mdd.
- Editor:** Displays the code for the selected file, Datos1RESTTest01.java:

```
17 @SpringBootTest
18 //@Sql({ "/schema.sql", "/data.sql" })
19 class Datos1RESTTest01 {
20
21     private static final String NOMBRE = "Maria";
22     private static final double ESTATURA = 1.68;
23
24     @Autowired
25     AlumnoRepository repositorioAlumno;
26
27     @Test
28     void buscarTodosRestTest() {
29
30         get("api/alumnos").then().body("matricula", hasItem("2A"));
31
32         //assertThat(milista.size(), greaterThan(6));
33     }
34
35
36
37 }
38 }
```

Spring REST GET/{ID}



http://localhost:8080/api/alumnos/2A

The screenshot shows the Postman application interface. At the top, there are three tabs: POST https://servic..., POST https://servic..., and GET http://localhost... (which is the current tab). Below the tabs, the URL http://localhost:8080/api/alumnos/2A is entered. The method is set to GET. To the right of the URL, there are Save, Edit, and Delete buttons. The 'Params' tab is selected under the Headers section, which contains six entries. The Body tab is selected, showing a JSON response:

```
1 {
2   "matricula": "2A",
3   "nombre": "Nadia",
4   "paterno": "Perez",
5   "fnac": "2001-01-10T06:00:00.000+00:00",
6   "estatura": 1.56,
7   "calificaciones": [
8     {
9       "id": 1,
10      "materia": "BD",
11      "calificacion": 10
12    },
13    {
14      "id": 2,
15      "materia": "POO",
16      "calificacion": 9
17    }
18  ]
19}
```

The response status is 200 OK with a time of 1394 ms and a size of 460 B. There is also a 'Save Response' button.

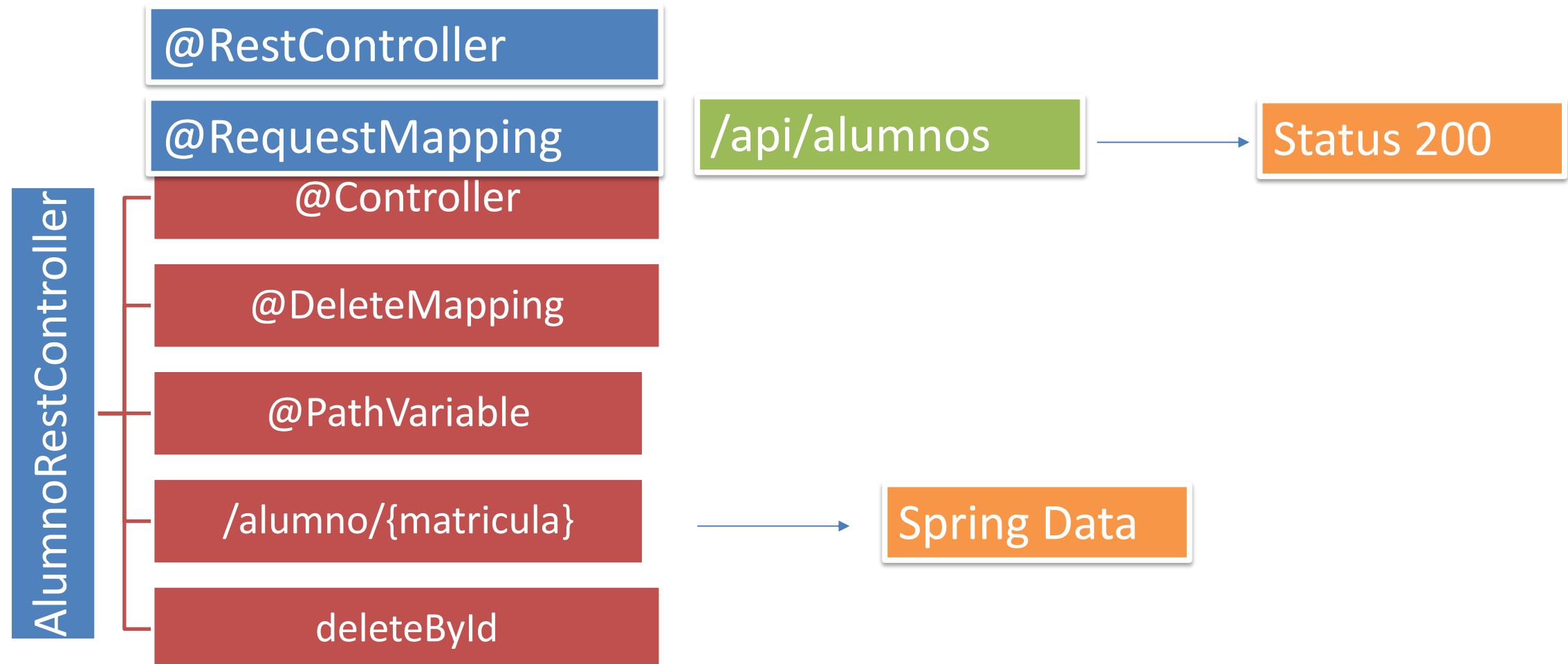


Lanzar la aplicación y Lanzar la segunda UT

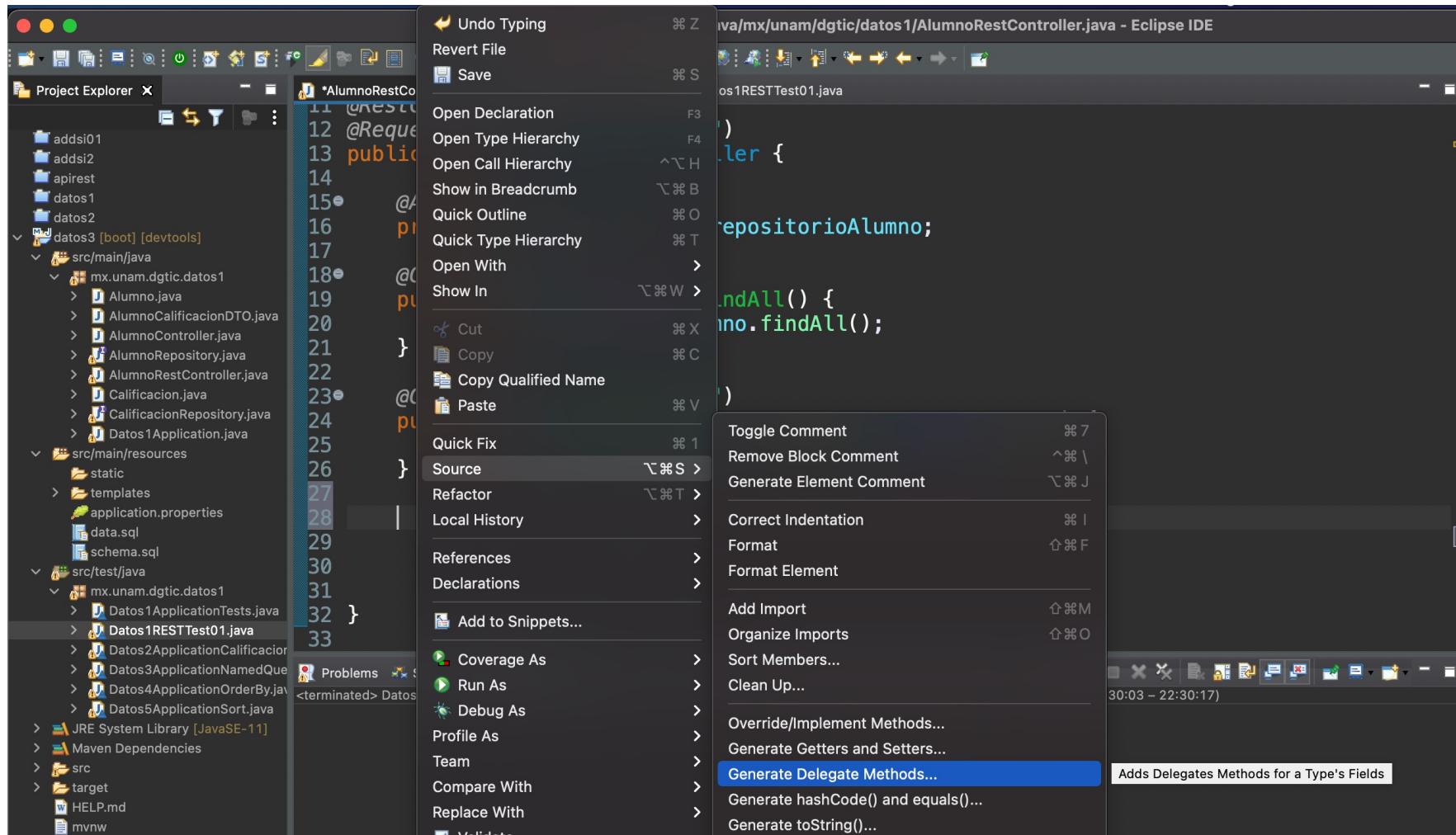
```
@Test  
void buscarUnoRestTest() {  
  
    get("api/alumnos/2A").then().body("nombre", equalTo("Nadia"));  
  
}
```



Spring REST DELETE/{ID}



Delete



DELETE

http://localhost:8080/api/alumnos/6GG

Save |  

DELETE http://localhost:8080/api/alumnos/6GG Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

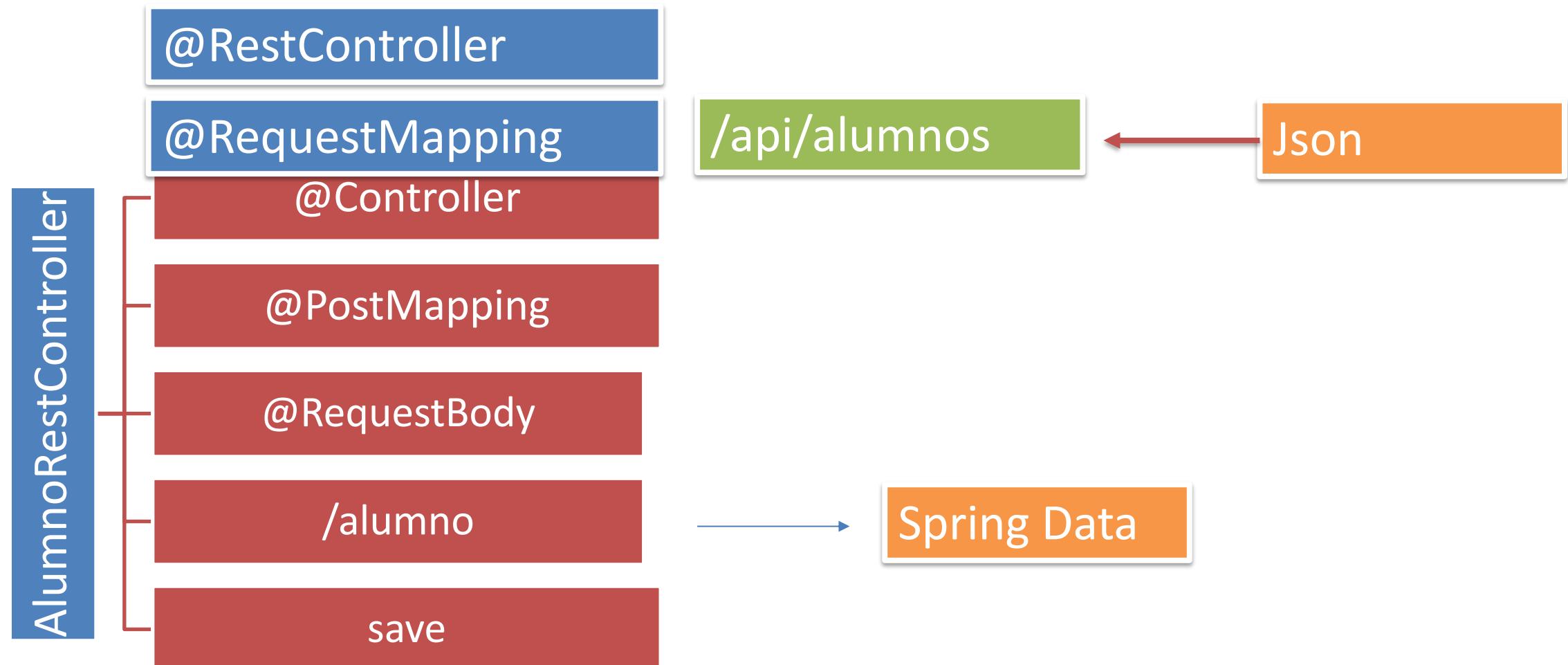
Body Cookies Headers (4) Test Results

200 OK 1397 ms 123 B Save Response

Pretty Raw Preview Visualize Text  

1

Spring REST POST



POST REST Test

```
@Test
void insertarUnoRestTest() throws JsonProcessingException {

    Alumno alumno = new Alumno("7GG", "Daniela", "Rios", new Date(), 1.62);
    ObjectMapper mapeador = new ObjectMapper();
    String objetoJsonString = mapeador.writeValueAsString(alumno);

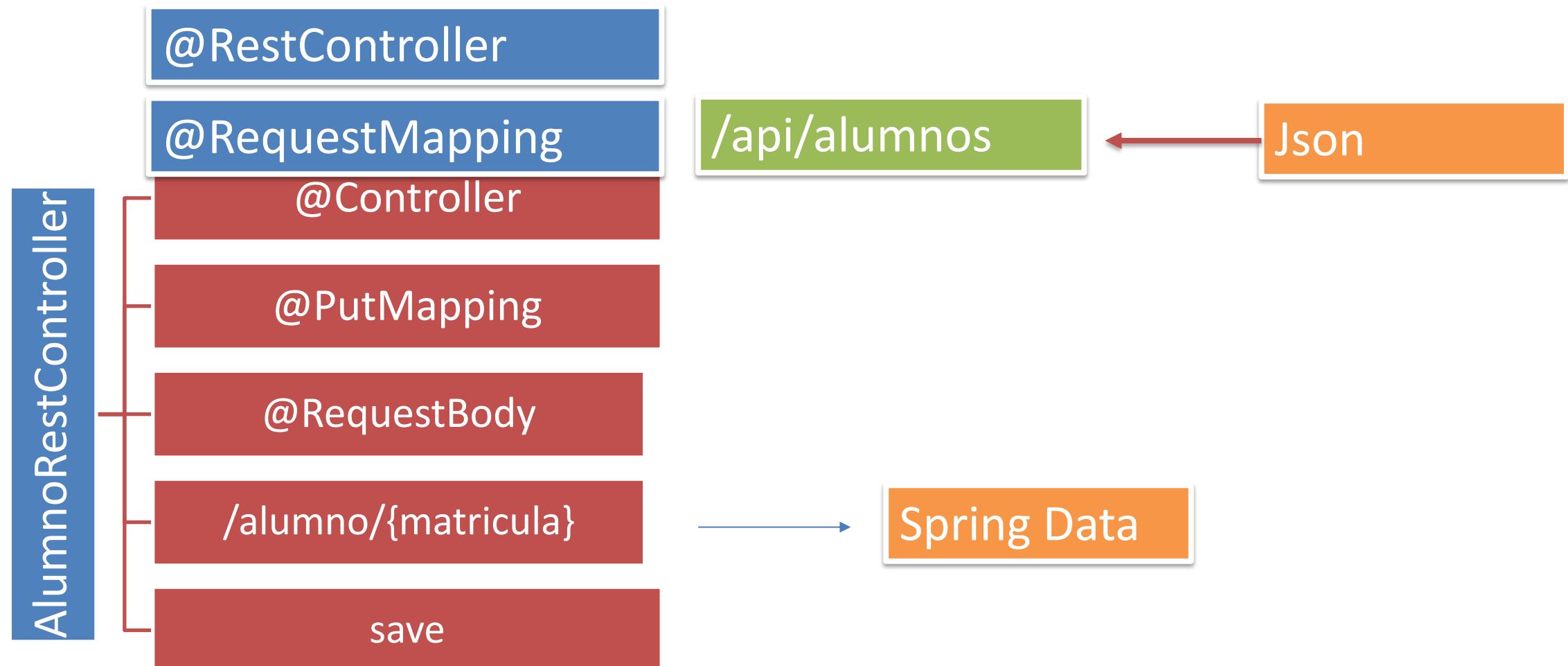
    System.out.println(objetoJsonString);

    given().header("Content-type", "application/json")
        .and().body(objetoJsonString)
        .post("api/alumnos").then().statusCode(200);

    get("api/alumnos/7GG").then().body("nombre", equalTo("Daniela"));

}
```

Spring REST PUT



Contacto

Dr. Omar Mendoza González

omarmendoza564@aragon.unam.mx

