



# **Rest API**

**R.I.A**

**Obligatorio 1**

**2018**

Gastón Añón - 4.919.990-9

Bruno Sasso - 4.824.259-3

Mónica Rodríguez - 4.473.862-5

Natalie Di Bono - 4.333.664-0

**Laboratorio 1- Taller de Aplicaciones de Internet Ricas**  
**Spring 5.0 Rest- Acceso a Google Drive**

<b>Proyecto SpringRestAPI</b>	<b>2</b>
Introducción	2
Repositorio	2
<b>Servicio Restful - Google Drive</b>	<b>2</b>
Pasos para Generar una URL Pública API DRIVE de Google	2
Estructura del Proyecto	6
Clase Directorio	7
Clase Archivo	8
Clase Spring Rest Controller	9
Anotaciones que se utilizan:	10
Mapeo a JAVA Object:	11
Mapeo a String:	11
Mapeo del Objeto JAVA obtenido a String con GSON:	12
Clase Main "RunAPI.java"	13
<b>Servicio Restful - ECHO</b>	<b>14</b>
Estructura del proyecto	14
Clase EchoString	14
Clase Spring Rest Controller	15
Anotaciones que se utilizan:	15
<b>Ejemplo de ejecución de ambas APIS usando POSTMAN:</b>	<b>17</b>
API Echo	17
API Google Drive	18

# Proyecto SpringRestAPI

## Introducción

En este documento se pretende explicar y ejemplificar la implementación de **2 servicios RestFul**: uno en el cual dada una **URL Pública** de un directorio de **Google Drive** se retorne un JSON con la lista de archivos que se encuentran en él, y otro **ECHO**, donde se pase como parámetro un string, y como resultado devuelva en formato Json el mismo String que se pasó como parámetro.

## Repositorio

El proyecto se encuentra con acceso público en GITHUB en la siguiente url:

<https://github.com/ndb1985/SpringRestAPI>

## Servicio Restful - Google Drive

Google facilita determinadas API's, entre ellas Google Drive API, la cual es invocada a través de un método GET realizado a una URL proporcionada por dicha API, permitiendo obtener una respuesta en formato JSON con el listado requerido.

Por último, dicho listado es mapeado a objetos JAVA siendo retornados posteriormente como un tipo de dato String en formato JSON

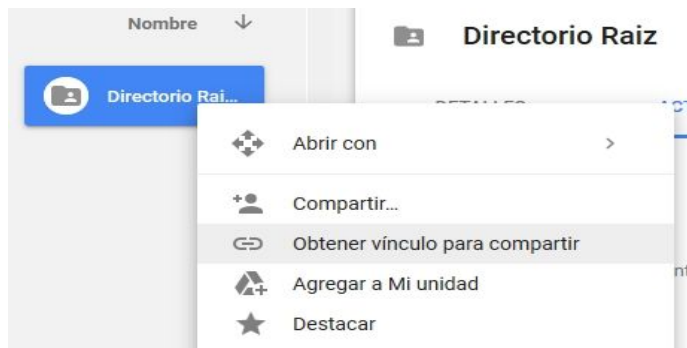
## Pasos para Generar una URL Pública API DRIVE de Google

1. Obtener enlace compartido sin autorización de la carpeta raíz:

Para ello habrá que hacer click derecho sobre la carpeta Raíz a la que queremos acceder y luego seleccionar "Obtener vínculo para Compartir".

## Laboratorio 1- Taller de Aplicaciones de Internet Ricas

### Spring 5.0 Rest- Acceso a Google Drive



Se obtendrá un link, y en él se mostrará el id de carpeta pública (en rojo), que es lo que necesitamos :

[https://drive.google.com/open?id=1xob03jh8nZbHsT4tZF63w\\_tYR9ZW4mhZ](https://drive.google.com/open?id=1xob03jh8nZbHsT4tZF63w_tYR9ZW4mhZ).

2. Ir a [https://developers.google.com/apis-explorer/?hl=en\\_GB#p/drive/v3/drive.files.list](https://developers.google.com/apis-explorer/?hl=en_GB#p/drive/v3/drive.files.list)

Esta URL nos redirige al sitio de la API de Drive y nos provee un formulario el cual luego de completar los campos necesarios nos permite obtener la url para el GET request.

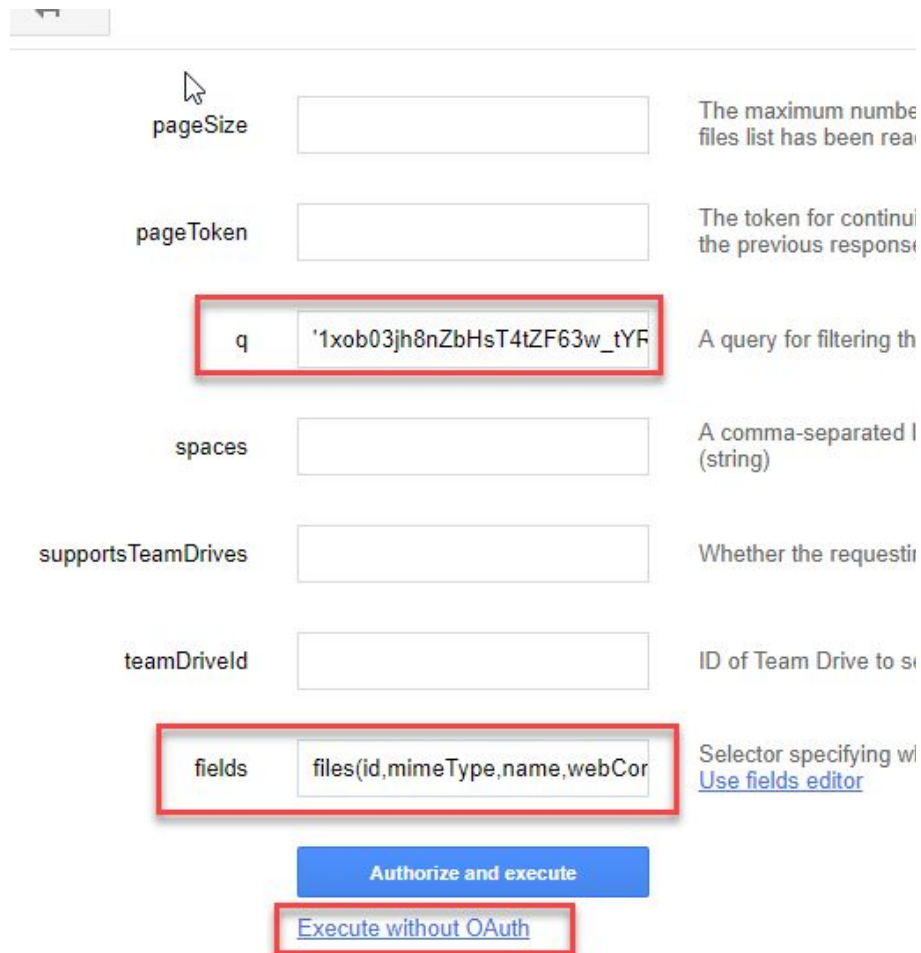
3. En el formulario se visualizarán varios campos vacíos, nosotros debemos completar el campo **q** y el campo **fields**.

Campo “q”: agregamos la siguiente información: '{su\_id\_de\_carpeta\_pública}' in parents  
Ej: **'1xob03jh8nZbHsT4tZF63w\_tYR9ZW4mhZ' in parents**

Campo “fields”: seleccionar “Use fields editor”, se abrirá un pop-up, seleccionar aquellos datos que nos interese recibir en la respuesta JSON. En nuestro caso será: “Id”, “name”, “mimeType”, “webContentLink” y “webViewLink”.

## Laboratorio 1- Taller de Aplicaciones de Internet Ricas Spring 5.0 Rest- Acceso a Google Drive

Una vez ingresado los datos seleccionamos “Execute without OAuth”.



pageSize	<input type="text"/>	The maximum number of files the list has been read
pageToken	<input type="text"/>	The token for continuing the previous response
q	<input type="text" value="'1xob03jh8nZbHsT4tZF63w_tYF'"/>	A query for filtering the results
spaces	<input type="text"/>	A comma-separated list of space names (string)
supportsTeamDrives	<input type="text"/>	Whether the request is for a team drive
teamDriveId	<input type="text"/>	ID of Team Drive to search
fields	<input type="text" value="files(id,mimeType,name,webContentLink,webViewLink)"/>	Selector specifying which fields to include. <a href="#">Use fields editor</a>

[Authorize and execute](#)

[Execute without OAuth](#)

Obtendremos una respuesta en la misma página con la URL para realizar el GET:

GET

```
https://www.googleapis.com/drive/v3/files?q='1xob03jh8nZbHsT4tZF63w_tYR9ZW4mhZ'+in+parents&fields=files(id%2CmimeType%2Cname%2CwebContentLink%2CwebViewLink)&key={YOUR_API_KEY}
```

Importante: debemos cambiar la sintaxis de la URL para que sea interpretada correctamente por el código.

Cambiamos %2C por , (coma):

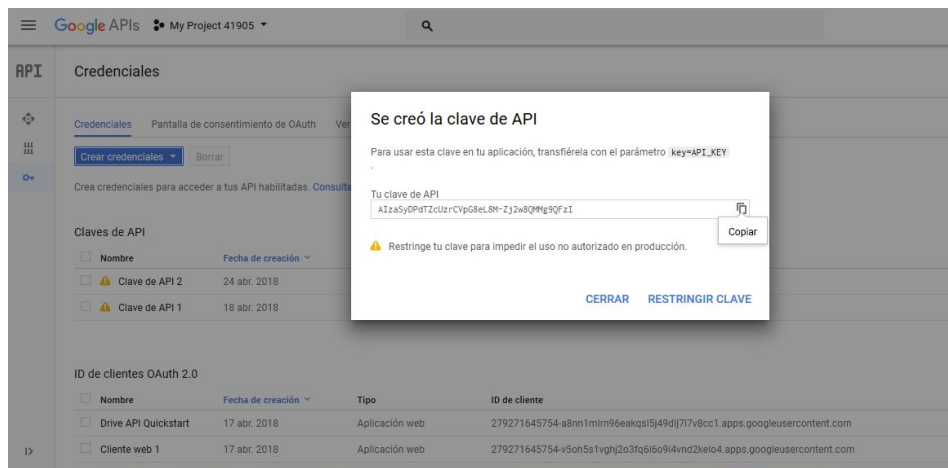
```
https://www.googleapis.com/drive/v3/files?q='1xob03jh8nZbHsT4tZF63w_tYR9ZW4mhZ'+in+parents&fields=files(id,mimeType,name,webContentLink,webViewLink)&key={YOUR_API_KEY}
```

## Laboratorio 1- Taller de Aplicaciones de Internet Ricas Spring 5.0 Rest- Acceso a Google Drive

### 4. Obtener Clave API (**API Key**):

- a).Acceder a Google Console: <https://console.developers.google.com/>
- b).Loguearse con la cuenta de google en la que se compartió públicamente un directorio
- c).Seleccionar Credenciales en el menú de la izquierda
- d).Crear credenciales > Clave de API
- e).Copiar la clave, click en “Cerrar”.

Ej: Clave de API: **AlzaSyDJymYnm4OA2Cf4YBEZhH5\_V\_\_vUu10rkY**



### 5. Generar URL:

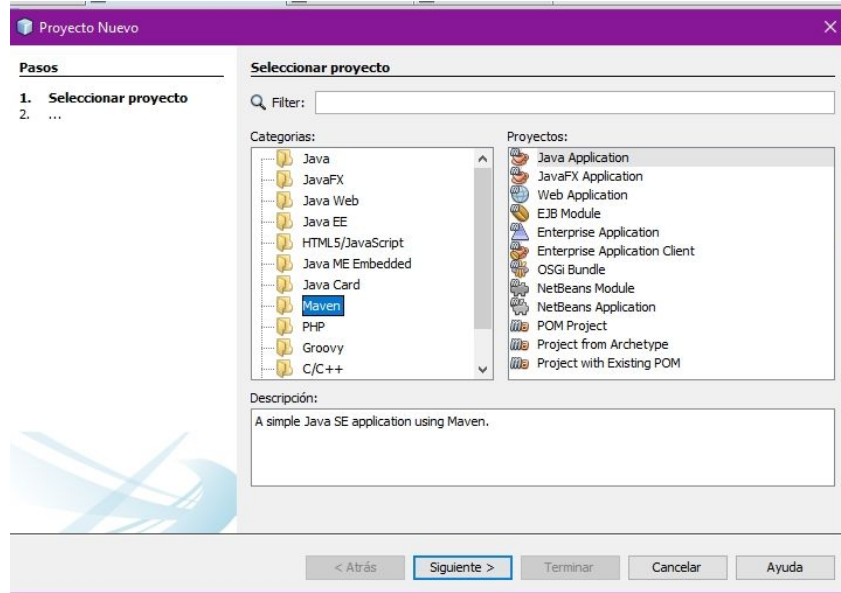
Se genera con la url dada en el Request + la API Key del paso anterior.

Ej: **URL Final**

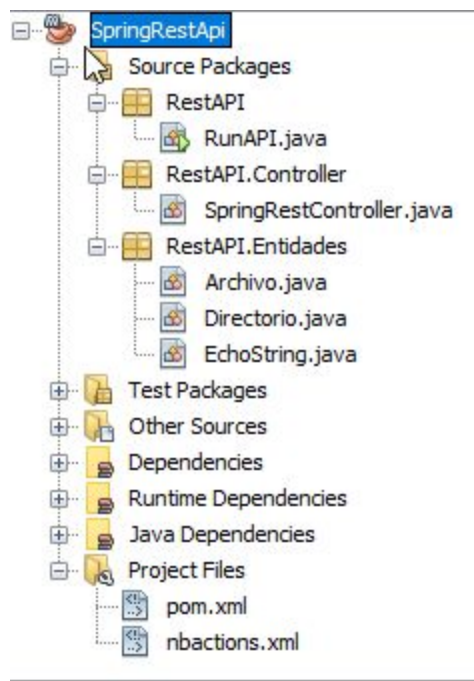
[https://www.googleapis.com/drive/v3/files?q='1xob03jh8nZbHsT4tZF63w\\_tYR9ZW4mhZ'+in+parents&fields=files\(id,mimeType,name,webContentLink,webViewLink\)&key=AlzaSyDJymYnm4OA2Cf4YBEZhH5\\_V\\_\\_vUu10rkY](https://www.googleapis.com/drive/v3/files?q='1xob03jh8nZbHsT4tZF63w_tYR9ZW4mhZ'+in+parents&fields=files(id,mimeType,name,webContentLink,webViewLink)&key=AlzaSyDJymYnm4OA2Cf4YBEZhH5_V__vUu10rkY)

## Estructura del Proyecto

Se creará un proyecto utilizando Maven al que le llamaremos SpringRestApi:



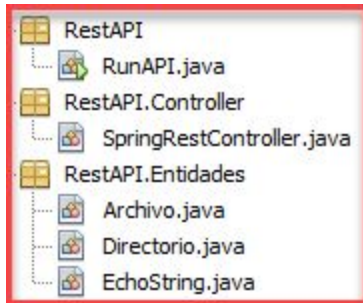
Se debe crear una estructura de paquetes a partir del paquete donde se encuentra la clase principal de Spring:



## Laboratorio 1- Taller de Aplicaciones de Internet Ricas

### Spring 5.0 Rest- Acceso a Google Drive

Se crearán 3 paquetes: - RestAPI que contiene la clase RunAPI.java,  
- RestAPI.Controller que contiene la clase SpringRestController.java  
- RestAPI.Entidades que contiene las clases Archivo.java y Directorio.java (EchoString se detalla más adelante).



## Clase Directorio

POJO (Plain Old Java Object) que define al objeto directorio:

```
@JsonIgnoreProperties(ignoreUnknown = true)
public class Directorio {
    @JsonProperty("files")
    private List<Archivo> listaArchivos;

    /.....constructor...Getters Setters...../
}
```

-Entre los atributos se define una lista de tipo Archivo, dado que el directorio a mostrar contiene una lista de los mismos.

### Anotaciones que se utilizan:

-@JsonIgnoreProperties(ignoreUnknown = true) : propiedad de la librería JSON que nos permite ignorar atributos que vengan en el response y no sean necesarios.

-@JsonProperty("files")  
private String tipo;



## Laboratorio 1- Taller de Aplicaciones de Internet Ricas

### Spring 5.0 Rest- Acceso a Google Drive

Permite definir un atributo en los POJOS (Plain Old Java Object) con un nombre diferente al que se envía en el JSON y hacer un match. Es decir todo lo que sea “files” se mostrará con el nombre de “listaArchivos”.

## Clase Archivo

```
@JsonIgnoreProperties(ignoreUnknown = true)
public class Archivo {
    @JsonProperty("id")
    private String id;
    @JsonProperty("name")
    private String nombre;
    @JsonProperty("mimeType")
    private String tipo_de_archivo;
    @JsonProperty("webContentLink")
    private String link_descarga;
    @JsonProperty("webViewLink")
    private String link_preview;

    /.....constructor...Getters Setters...../
}
```

-Es similar a directorio, contiene las mismas propiedades para ignorar y hacer match con determinados atributos del JSON (@JsonIgnoreProperties y @JsonProperty)

-Dicha clase no define ninguna lista.

-Se definen los atributos que nos interesa recibir en la respuesta JSON: id, nombre, tipo\_de\_archivo, webContentLink (Permite obtener un link para descargar el archivo siempre y cuando este no sea un documento propio de Google, es decir un Google doc, Google sheet, etc), webViewLink (Permite obtener un link para poder realizar un preview de todos los archivos incluidos los documentos de Google).

## Clase Spring Rest Controller

```
/*RequestMapping mapea las solicitudes http al metodo getListOfFilesFromDrive*/
@RequestMapping(value = "/sendGetToDrive", method = RequestMethod.GET, produces = "application/json")

public String getListOfFilesFromDrive() {

    //Template que provee Spring para interactuar de manera sencilla con servicios Rest desde el lado cliente
    RestTemplate restTemplate = new RestTemplate();
    //Metodo de RestTemplate que nos permite acceder directamente a la respuesta y extraer el body entre otros (headers,
    Directorio dir = restTemplate.getForObject("https://www.googleapis.com/drive/v3/files?q="
        /*FolderId*/ + "'1xob03jh8nZbHsT4tZf63w_tYR9Zw4mhZ'+in+parents"
        /*Partial Response*/ + "&fields=files(id,mimeType,name,webContentLink,webViewLink)"
        /*APIKey*/ + "&key=AIzaSyDJymYnm4OA2Cf4YBEZhH5_V_vUul0rkY", Directorio.class);

    /*IMPRIMIMOS LA INFO EN FORMATO JSON*/
    //Gson nos ayuda a imprimir con formato el JSON en String
    Gson gson = new GsonBuilder().setPrettyPrinting().create();
    String jsonString = gson.toJson(dir);

    //Imprimimos en consola en formato JSON
    System.out.println(jsonString);

    //retornamos un String con la data del json
    return jsonString;
}
```

-Dicha clase actúa como controlador y maneja los requests de modo que en cada respuesta sea posible retornar un objeto de dominio (ejemplo en formato JSON).

-Se define un método de tipo String dado que se va a retornar el body del response como String en formato JSON.

-Para esto se realizan los siguientes pasos:

1. Se instancia RestTemplate
2. Se obtiene la data con getForObject()
3. Se Mapea al ser obtenida a un objeto JAVA (Directorio)
4. Con Gson se pasa el objeto dir a String para imprimirlo en formato JSON con el metodo toJson de GSON.

## **Anotaciones que se utilizan:**

### **@RestController**

-Define a la clase como Controlador

### **@RequestMapping**

-Asegura que las solicitudes HTTP que se realicen a determinada url se mapean al método correcto. En éste caso nos aseguramos que las solicitudes realizadas a "localhost:8080//sendGetToDrive" se mapean al método: *getListOfFilesFromDrive()*.

### **RestTemplate**

-Template que provee Spring para interactuar de manera sencilla con servicios Rest  
-Simplifica la comunicación con los servidores HTTP aplicando principios RESTful. Maneja conexiones HTTP, permitiendo a través de la URL la extracción de los resultados.

### **restTemplate.getForObject()**

(getForObject(java.net.URI url, java.lang.Class<T> responseType))

-Recupera el response haciendo un GET a la URL dada

-Es posible recuperar una representación del response en String o bien en cualquier tipo de objeto definido como POJO en el proyecto.

## **Ejemplos de mapeos:**

## Laboratorio 1- Taller de Aplicaciones de Internet Ricas

### Spring 5.0 Rest- Acceso a Google Drive

#### Mapeo a JAVA Object:

```
Directorio dir =  
restTemplate.getForObject("https://www.googleapis.com/drive/v3/files?q='1xob03jh8nZbHsT4tZ  
F63w_tYR9ZW4mhZ'+in+parents&fields=files(id,mimeType,name,webContentLink,webViewLink  
)&key=AlzaSyDJymYnm4OA2Cf4YBEZhH5_V__vUu10rkY", Directorio.class);
```

Habiendo desarrollado el método toString en las clases Directorio y Archivo, esta es una posible impresión en consola del contenido del objeto mapeado:

```
System.out.println(dir.toString());
```

Mostrando en consola:

```
Directorio{', listaArchivos=[Archivo{id=1dYFI2EmJnbWePIZ9DfIXPiBq7QyPDuFi,  
nombre=Copy of TestFunc_Práctico2_2018.pdf, tipo_de_archivo=application/pdf,  
link_descarga=https://drive.google.com/uc?id=1dYFI2EmJnbWePIZ9DfIXPiBq7QyPDuFi&expor  
t=download,  
link_preview=https://drive.google.com/file/d/1dYFI2EmJnbWePIZ9DfIXPiBq7QyPDuFi/view?usp  
=drivesdk}, .....etc....}}}
```

#### Mapeo a String:

```
String jsonString =  
restTemplate.getForObject("https://www.googleapis.com/drive/v3/files?q='1xob03jh8nZbHsT4tZ  
F63w_tYR9ZW4mhZ'+in+parents&fields=files(id,mimeType,name,webContentLink,webViewLink  
)&key=AlzaSyDJymYnm4OA2Cf4YBEZhH5_V__vUu10rkY", String.class);
```

Impresión del string generado con RestTemplate:

```
System.out.println(jsonString);
```

Mostrando en Consola:

```
{  
  "listaArchivos": [  
    {  
      "id": "1dYFI2EmJnbWePIZ9DfIXPiBq7QyPDuFi",  
      "nombre": "Copy of TestFunc_Práctico2_2018.pdf",  
      "tipo_de_archivo": "application/pdf",
```

## Laboratorio 1- Taller de Aplicaciones de Internet Ricas

### Spring 5.0 Rest- Acceso a Google Drive

```
"link_descarga":
"https://drive.google.com/uc?id\u003d1dYFI2EmJnbWePIZ9DfIXPiBq7QyPDuFi\u0026export\u003ddownload",
"link_preview":
"https://drive.google.com/file/d/1dYFI2EmJnbWePIZ9DfIXPiBq7QyPDuFi/view?usp\u003ddrivesdk"
},
{
  "id": "1hFy4KtSSf-NrT04VUjNxx8CKvmINxt9jrJKTn2O1_bQ",
  "nombre": "HOLA AAAAD$%\"\\u0026^\"·$$ wsr/(\u0026%$!á",
  "tipo_de_archivo": "application/vnd.google-apps.drawing",
  "link_descarga":
"https://docs.google.com/drawings/d/1hFy4KtSSf-NrT04VUjNxx8CKvmINxt9jrJKTn2O1_bQ/image?w\u003d512\u0026ouid\u003d0",
  "link_preview":
"https://docs.google.com/drawings/d/1hFy4KtSSf-NrT04VUjNxx8CKvmINxt9jrJKTn2O1_bQ/edit?usp\u003ddrivesdk"
},
.....
}
]
}
```

### **Maapeo del Objeto JAVA obtenido a String con GSON:**

Gson nos ayuda a imprimir con formato JSON el String obtenido a partir del objeto JAVA.

```
Gson gson = new GsonBuilder().setPrettyPrinting().create();
String jsonString = gson.toJson(dir);
Imprimiendo en Consola:
System.out.println(jsonString);
```

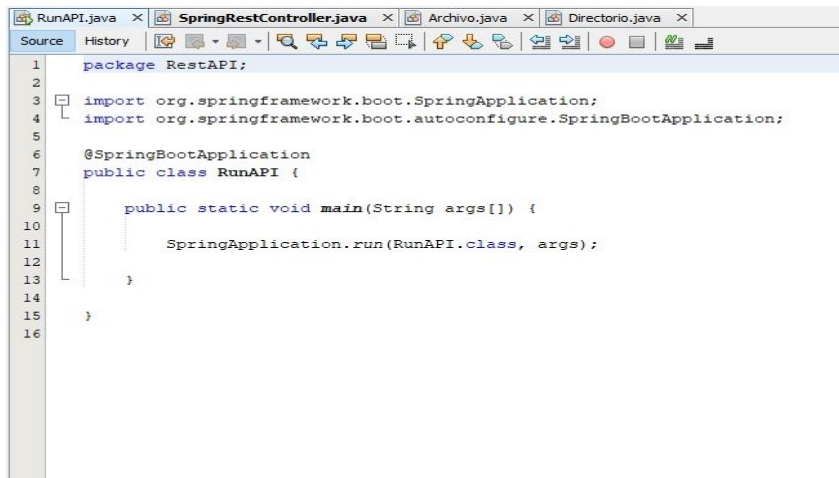
Retornamos un String con la data del json, (ésto nos permite poder generar un get request desde un navegador o cualquier aplicación como Postman y ver en pantalla el response como fue mapeado al POJO (clase Directorio)).

```
return jsonString;
```

## Clase Main “RunAPI.java”

Esta es la clase principal del proyecto, la que inicia el servidor ya que con la ayuda de **Spring Boot** es posible iniciar el server sin necesidad de utilizar Tomcat u otro servicio aparte (ya que este se encuentra embebido).

Para ello se define la anotación **@SpringBootApplication** y luego en la clase dentro del método Main se invoca el método `SpringApplication.run` pasandole como parámetro el nombre de la clase `RunAPI.class`



```
1 package RestAPI;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class RunAPI {
8
9     public static void main(String args[]) {
10
11         SpringApplication.run(RunAPI.class, args);
12
13     }
14
15 }
16
```

## Servicio Restful - ECHO

Aplicación Restful que hace eco de los requests que haga el usuario.  
Dicho de otro modo, ésta API retorna en formato JSON el string ingresado por el usuario en la URL.

**Ejemplo:**

**GET request:**

localhost:8080/enviando request de prueba

**Response:**

```
{  
  "id":1,  
  "content":"enviando request de prueba"  
}
```

## Estructura del proyecto

Ver “Estructura del Proyecto” en “Servicio Restful - Google Drive”.

## Clase EchoString

```
public class EchoString {  
  
    private final long id;  
    private final String content;  
  
    /.....constructor...Getters Setters...../  
}
```

## **Clase Spring Rest Controller**

Ver “Clase Spring Rest Controller” en “Servicio Restful - Google Drive”.

-Para el servicio echo, en dicha clase se agrega (o define) lo siguiente:

```
@RestController
public class SpringRestController {

    private final AtomicLong counter = new AtomicLong();
    .....}
```

1.El atributo *counter* (de tipo *AtomicLong* explicado más abajo) se define como atributo de la clase *SpringRestController*:

-**AtomicLong**: es un tipo de dato long que tiene la capacidad de poder actualizarse automáticamente mediante un metodo propio *incrementAndGet*. Por lo que en cada petición o request hecho al servidor, permite que el atributo ID incremente por sí solo.

2.Y el método *EchoString* encargado de retornar el string en formato JSON:

```
public EchoString mensaje(@PathVariable String sample) {

    EchoString msjEcho = new EchoString(counter.incrementAndGet(), sample);
    return msjEcho;
}
```

-En éste método simplemente se instancia la clase *EchoString* y se crea el objeto *msjEcho* al cual se le pasan los siguientes parámetros: *counter* y *sample*.



### **Anotaciones que se utilizan:**

#### **@RequestMapping("/{sample}")**

-Asegura que las solicitudes HTTP que se realicen a “localhost:8080/algun string” se mapean al método *EchoString mensaje*.

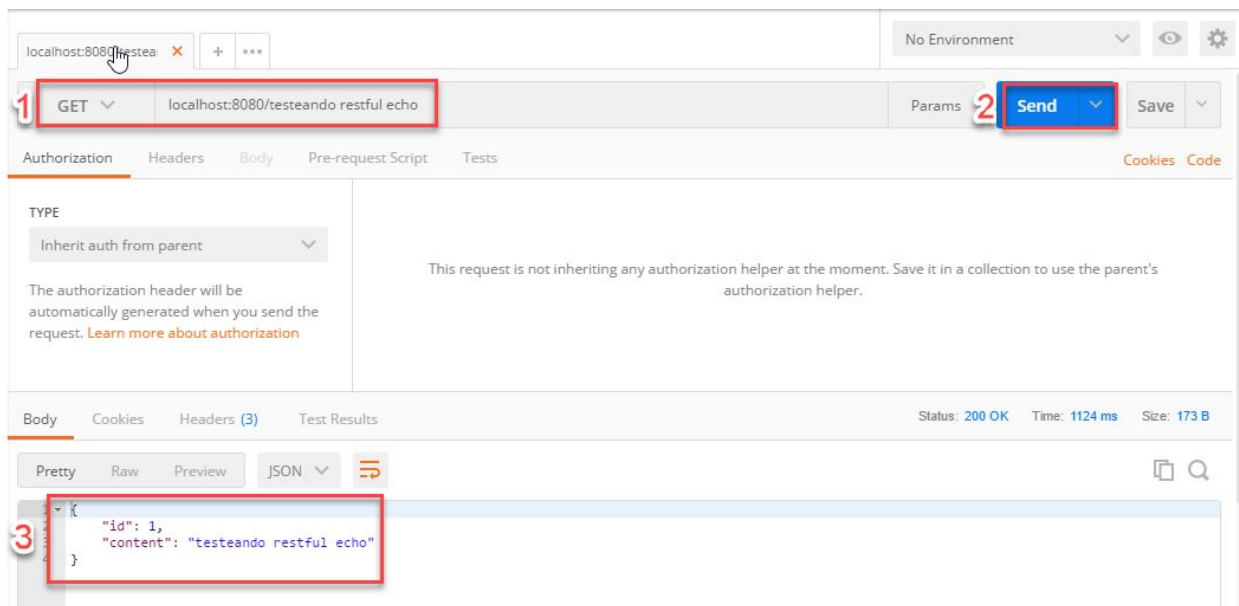
#### **@PathVariable String sample**

-@PathVariable indica que determinado parámetro (el String “sample” en éste caso) se vinculará a determinada variable en la URL.

Para ejecutar éste servicio rest, se utiliza la clase definida como clase principal del proyecto “RunAPI.java”, la cual se explica en detalle en “ Clase Main “RunAPI.java” en “Servicio Restful - Google Drive”.

## Ejemplo de ejecución de ambas APIs usando POSTMAN:

### API Echo



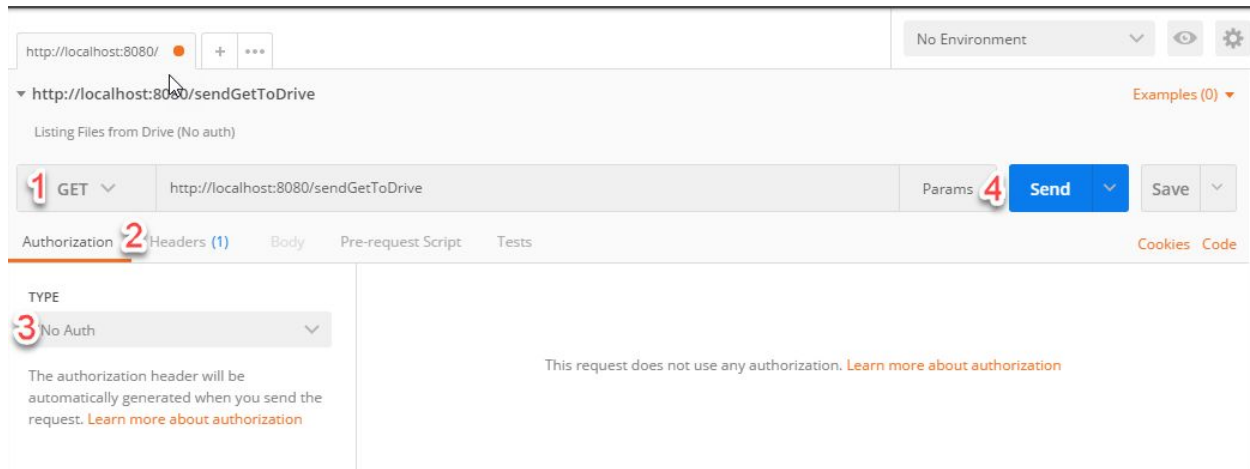
1-Abrimos POSTMAN, agregamos una nueva pestaña, Seleccionamos GET y escribimos la url a la cual vamos a realizar el request:  
`http://localhost:8080/testeando restful echo`

2-Presionamos "Send"

3-Obtenemos la respuesta en formato JSON

```
{  
  "id": 1,  
  "content": "testeando restful echo"  
}
```

## API Google Drive

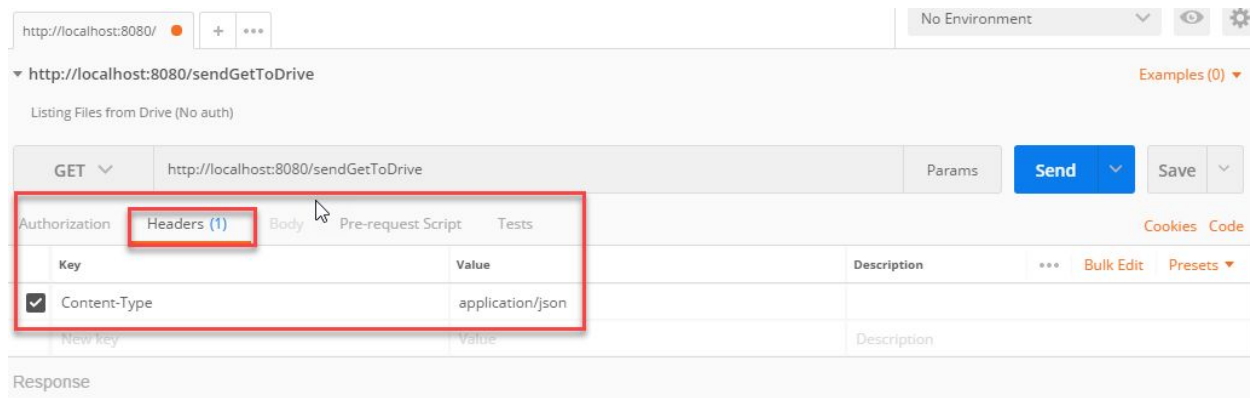


1-Abrimos POSTMAN, agregamos una nueva pestaña, Seleccionamos GET y escribimos la url a la cual vamos a realizar el request:

<http://localhost:8080/sendGetToDrive>

2-Seleccionamos Headers y agregamos el tipo de MediaType que acepta como respuesta nuestra API:

**Key:** Content-Type **Value:** application/json



3-Seleccionamos tipo de autorización: “No auth” dado que no necesitamos autorización de ningún tipo siendo que la URL es publica.

## Laboratorio 1- Taller de Aplicaciones de Internet Ricas

### Spring 5.0 Rest- Acceso a Google Drive

4- Presionamos "Send"

5- Obtenemos la respuesta en formato JSON

