Tabla de contenido

[¿Qué es Spring boot?](#_o7bagkhd922e)

[Características principales de spring boot](#_v1ceudipzisj)

[Instalación del entorno de desarrollo Windows](#_mb7hs250m4j4)

[Spring Initializer](#_5da8ye7fouxx)

[¿Qué es una JPA?](#_95fjfo7r9kmj)

[Mi primer hola mundo en spring boot](#_hjy2ptpi2y9a)

[Estructura del proyecto](#_v4kajiokfu5y)

[Anotaciones en java](#_7oelqf8fwu6g)

[¿Que es la anotación @entity](#_aue5o19v0yri)

[Creación del repository](#_g1yqkufse0lo)

[Uso de dependencias en maven](#_42ss7ay0dyrn)

[Datasource y conexión a la base de datos.](#_dyhk3c9825cq)

[¿ Qué es Workbench ?](#_nwvhtqn08q4z)

[Creando las rutas web con @getMapping @postMapping @Autowired](#_veg1i76uz4m2)

[Probando nuestro programa con postman](#_f1iu9jtojlqj)

[¿Qué es JSON?](#_68h1stmy9qyd)

* En este curso aprenderás a construir una aplicación en spring boot
* Entender los conceptos de api , api rest , json
* Aprenderás el funcionamiento de spring boot a nivel general

# **¿Qué es Spring boot?**

Proyecto basado en Spring. No es lo mismo que Spring. Es un proyecto que forma parte del core de Spring, al igual que Spring Cloud, etc.

El objetivo principal es que sólo te centres en correr la aplicación, sin preocuparte por temas de configuración, etc.

Tiene la gran ventaja de poder integrar librerías de terceros de manera muy sencilla.

No tendremos que preocuparnos por configuraciones a nivel de XML, sólo configuraciones mínimas a nivel de propiedades (ponerle el puerto, etc).

No tendremos que preocuparnos de desplegar nuestra aplicación en un servidor web local cuando queramos hacer pruebas, Spring Boot también contempla esto y lleva incorporado un servidor web dónde se desplegará nuestra aplicación automáticamente.

Spring Boot tiene configurado servidores web embebidos dentro de él teniendo en cuenta que esta característica puede tener en sí un servidor como tomcat o jetty para poder desplegar las aplicaciones.

# **Características principales de spring boot**

Independiente: no tenemos que preocuparnos de las dependencias del core de Spring ni de la compilación de estas.

Incrustado Tomcat, Jetty o Undertow: Spring Boot trae consigo un servidor web como los tres mencionados donde podemos correr nuestra aplicación sin preocuparnos de generar un artefacto WAR o JAR y desplegarlo nosotros mismos.

Proporción de dependencias: no debemos preocuparnos por las configuraciones de dependencias de terceros o del core de Spring, Spring Boot se encargará de inyectar todo lo necesario.

Sin generación de XML: No debemos preocuparnos de configuración XML para que nuestra aplicación funcione.

Métricas de salud del aplicativo: podemos validar el estado de un servicio desplegado, sus dependencias, estado de memoria, magnitud de configuración, etc.

# **Instalación del entorno de desarrollo Windows**

Desarrolla tu primer API REST con Java, spring boot, jpa y mysql:

* Descargar JAVA: [https://www.java.com/es/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUZYWnlQZkFtRzhFZjB5TWJsR0xZYUdiYmRRd3xBQ3Jtc0trN0pyTlVaMzdZalNKSU5aaUZLXzZHMzZrYzlOa0dGT0tONlJfRkx1TFFXckRGOXVOd1dNY0VuNTliTElIcVNqb0ZYak0xN3I5bE9sS21aTlVReHdmOERUSUNZSC1CRGozWVRfVnZkLXl1SmNFOHFzRQ&q=https%3A%2F%2Fwww.java.com%2Fes%2F&v=OkD1wBpPsNM)
* Descargar MAVEN: [https://maven.apache.org/download.cgi](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbXd3RHVTQVI1UTh2WjNmRUlTQnN2V2NNN1BMUXxBQ3Jtc0tuMjQ1ZTI5N2stRFlJYnlfVkI0WG4yLWNTTnJmVTFKR01CMnMtSGN6NTItS3dUcDJscm9hMGFoYzNjUW5SMmRDOHQtMFZXb0pNNFd2SkRIQVFYZGtRcmxJRzMyeFZLYUFjOTdqN0hZdDY0SG80NVAzaw&q=https%3A%2F%2Fmaven.apache.org%2Fdownload.cgi&v=OkD1wBpPsNM)
* Crear cuenta en Oracle: [https://profile.oracle.com/myprofile/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUFkUG0zQlBZdE83UzdjRnVjU2dGcGZCX0JxZ3xBQ3Jtc0tuVmZKeEVPdmVnRGVyTFVJcktfOG5vWUh3LXpWOEFHbWdQQl9iYWlvT0s2Rm5zSC1aZGgxYnJJY3l6bjhzTHg4VjVfaGhFcEF3TVNTeHJoYmNPal9NR2dYWjVIakI4NTJlbXhlejhuTzNPdFk2RUJicw&q=https%3A%2F%2Fprofile.oracle.com%2Fmyprofile%2Faccount%2Fcreate-account.jspx&v=OkD1wBpPsNM)
* Descargar MySql server: [https://dev.mysql.com/downloads/mysql/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa09iLWlhN3c3cnhqSVEzV0RSU3YyR0JvSXNmUXxBQ3Jtc0trQ1JQRFZrZ3Mxai1TS2tDR3FENkRTX1lCRTlqM2FHMHo1TXktTDRuOXV2UDNQNmEwRTNOZjdjUG9JekhsYmJuZWkwNGgzM0lCQUNKV0phSDlnLWVPQWszUi1yLS1qaHhqclpqRnZUZVFYN3VvcGlxVQ&q=https%3A%2F%2Fdev.mysql.com%2Fdownloads%2Fmysql%2F&v=OkD1wBpPsNM)
* Descargar MySql workbench:[https://dev.mysql.com/downloads/workb...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbDBITk9jZ0NXM0tncDBxSVctSkN1dTkxSkUtZ3xBQ3Jtc0tudFZXelVxeThJdm00elg1M1IxTGttc21XQnlCZWFWM0dweVFubE1aY1UtVy1ZZnROd1FxbkFlQ0NGQjhyOWxPOC1BTEc2MmFSUHJ1RzVhQl9uQnVoSEFDN2VtZ08tQ0dXeVN0VVhFT2hSdV9UQzZadw&q=https%3A%2F%2Fdev.mysql.com%2Fdownloads%2Fworkbench%2F&v=OkD1wBpPsNM)
* Descargar IntelliJ: [https://www.jetbrains.com/es-es/idea/...](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa3IyZFkzSzN3T1hYR2ZmWWQ1bEcwZ1lIN3B2Z3xBQ3Jtc0tuVGdwZm5mY3c0b01wTVZJTE12eFZwdmRNY1l4eFlMVnRHUlFQVm9YaFI0bzZoNE9TTDl0TXVqMXdwWkNUaV9rR1k1d2dsck9rX3MyOE9EdUZybUduWnFqY1dMX3NueVJxcGtrWWpjdFhzTnRPREF1aw&q=https%3A%2F%2Fwww.jetbrains.com%2Fes-es%2Fidea%2Fdownload%2F%23section%3Dwindows&v=OkD1wBpPsNM)
* Descargar Postman: [https://www.postman.com/downloads/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbC1MU2lKS2hEdV9aWHBtTDF1VGl1Y2RzeGc4QXxBQ3Jtc0ttXzkyNWJuNmN6azlkUHR4aVY4UGswNDVKY2JMeHFJQVozYmdmd0QxWk1FeGlKMC1WQW50cGwyc2tzb1JmU21uMzEwdFJKTUthczY1U3hUR2htTkxyNmhvXy1idDVILUY1bkE1eWI3STJSNTNRd2JNaw&q=https%3A%2F%2Fwww.postman.com%2Fdownloads%2F&v=OkD1wBpPsNM)

# **Spring Initializer**

# **¿Qué es una JPA?**

La API de persistencia Java es una especificación que indica cómo se debe hacer la persistencia o almacenamiento de los objetos en una aplicación construida en java.

**¿Qué es REST?**

RestFul API

Interfaz de programación de aplicaciones que se ajusta a la arquitectura REST.

Rest:

Es el conjunto de requerimientos o solicitudes HTTPS responsables de las operaciones básicas necesarias para la manipulación de datos.

**POST:Crea datos en el servidor**

**GET:Obtiene los datos**

**PUT:Modifica / actualiza los datos**

**DELETE:Borra los datos**

**POST = Create**

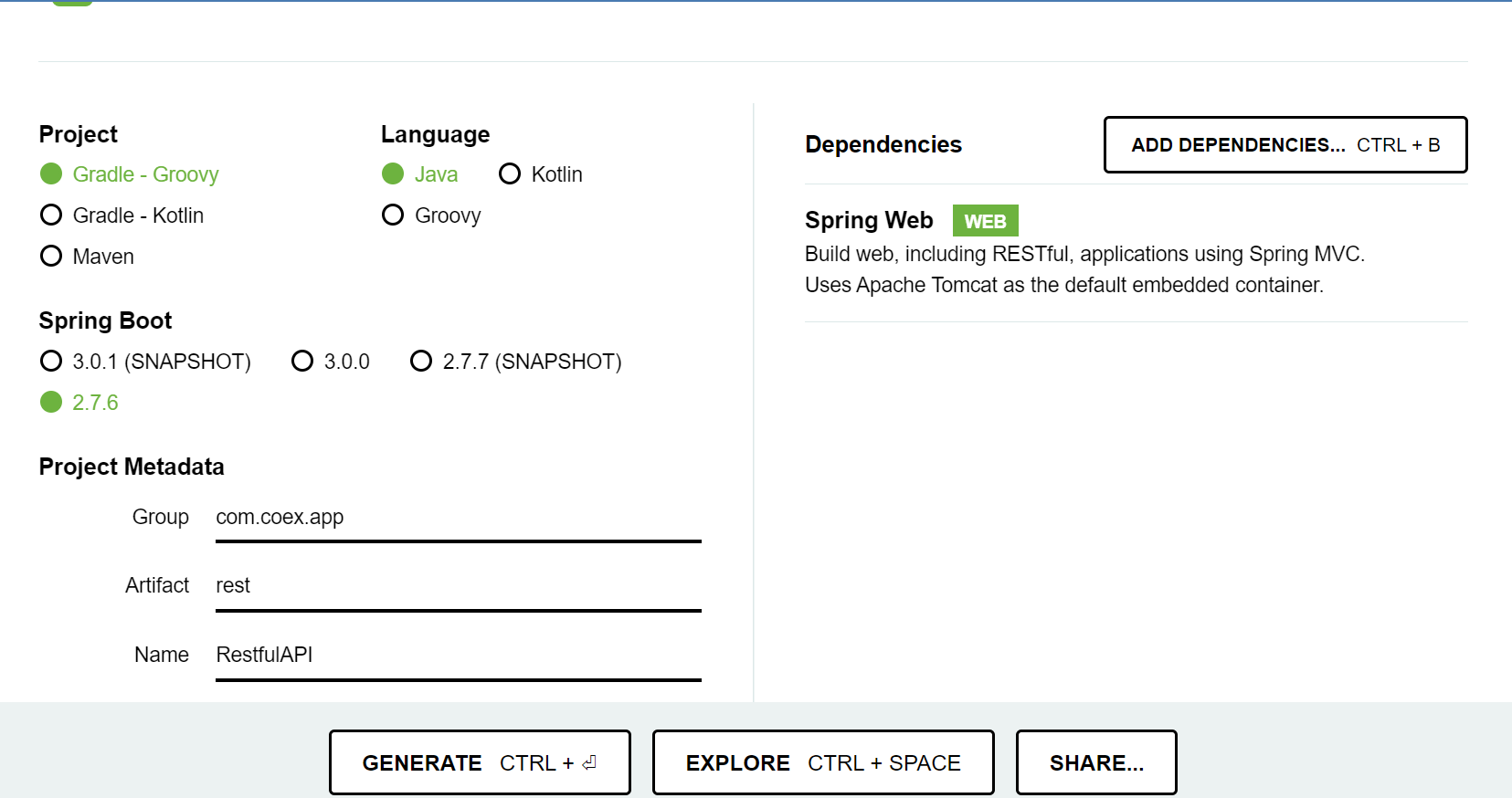
**GET = Read**

**PUT = Update**

**DELETE = Delete**

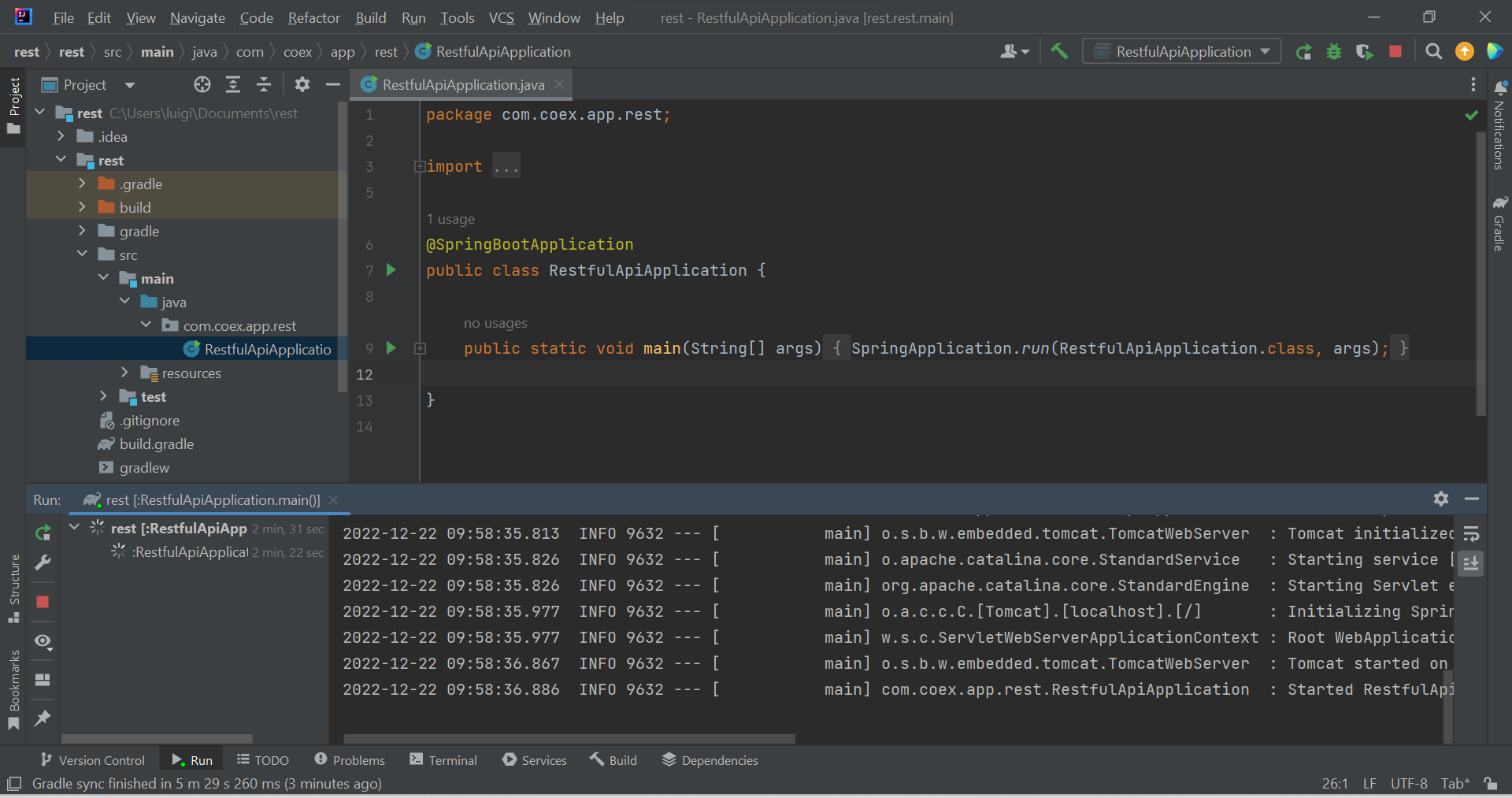
# **Mi primer hola mundo en spring boot**

Antes de continuar en este apartado es necesario que tengas instalado Java, Maven , Mysql server y Mysql workbench o en su defecto tener PHP Phpmyadmin para configurar Mysql.



De esta forma tenemos la configuración de nuestro proyecto.

Abrimos el proyecto en nuestro IDE.



Vamos a crear una capa llamada controlador y dentro de esa capa de controlador vamos a crear una nueva clase llamada todoController.

Este todoControles se encargará de manejar la puerta de entrada de nuestra aplicación.

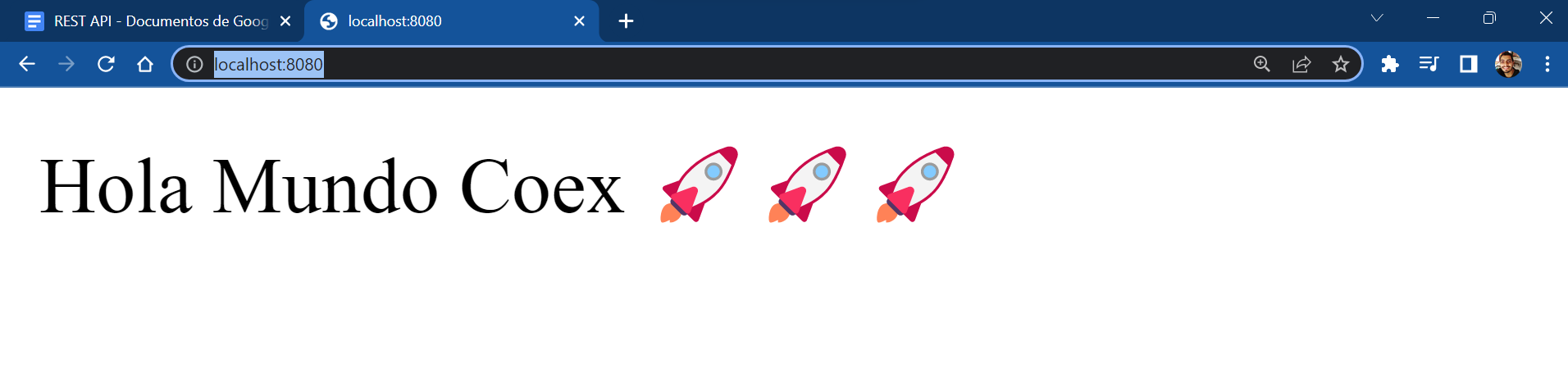
para que el programa comprenda que una clase se convertirá en un controlador usaremos la notación @ResController

@getMapping es otra notación que usaremos y esta es para que nos muestre los datos de salida de nuestro string y de esta forma cuando el programa levante el servidor nos muestre en el navegador el string.

si usamos

@GetMapping(value = "/")

Nos servirá para indicar que en la carpeta raíz se ejecutará el programa y mostrará lo que se encuentra en esta ruta .



# **Estructura del proyecto**

La estructura del proyecto es fundamental ya que nos va a permitir que nosotros identifiquemos cuáles van a ser pasos o pautas para la construcción de nuestro aplicativo.

para esto ya tenemos una carpeta |encargada del controlador y ahora crearemos una carpeta encargada del modelo el cual nos permite tener una plantilla de la estructura de los datos a trabajar.

Nuestro proyecto está compuesto de una lista de tareas que está a su vez tendrá un idTarea

un tituloTarea una descripcionTarea.

En nuestra clase Task crearemos una anotacion de nombre @Entity.

¿Qué es un Entity en Spring boot?

Resultado de imagen para que es un entity en java

Una entidad es la representación de información que nosotros necesitamos en nuestra aplicación. Esta entidad podría ser un usuario, un producto o cualquier dato que nuestra aplicación necesita mantener persistente para luego recuperarla cuando la necesite.

<https://gustavopeiretti.com/spring-boot-entities-conceptos-iniciales/>

# **Anotaciones en java**

Uso de Anotaciones

Las anotaciones que se añaden a las clases, métodos, campos, etc., son etiquetas que generan metadatos a nuestras clases (conjunto de datos que describen el contenido o propósito de un objeto). Esto nos permite prescindir de los ficheros XML

# **¿Que es la anotación @entity**

¿Qué es un Entity en Spring boot?

Resultado de imagen para que es un entity en java

Una entidad es la representación de información que nosotros necesitamos en nuestra aplicación. Esta entidad podría ser un usuario, un producto o cualquier dato que nuestra aplicación necesita mantener persistente para luego recuperarla cuando la necesite.

<https://gustavopeiretti.com/spring-boot-entities-conceptos-iniciales/>

Cuando se tiene una tabla en la que su llave primaria es compuesta, se debe hacer una clase aparte que contenga los atributos que conforman la llave.

Creando la Clase que tendrá nuestra llave compuesta

Recuerda que debe de contener lo siguiente:

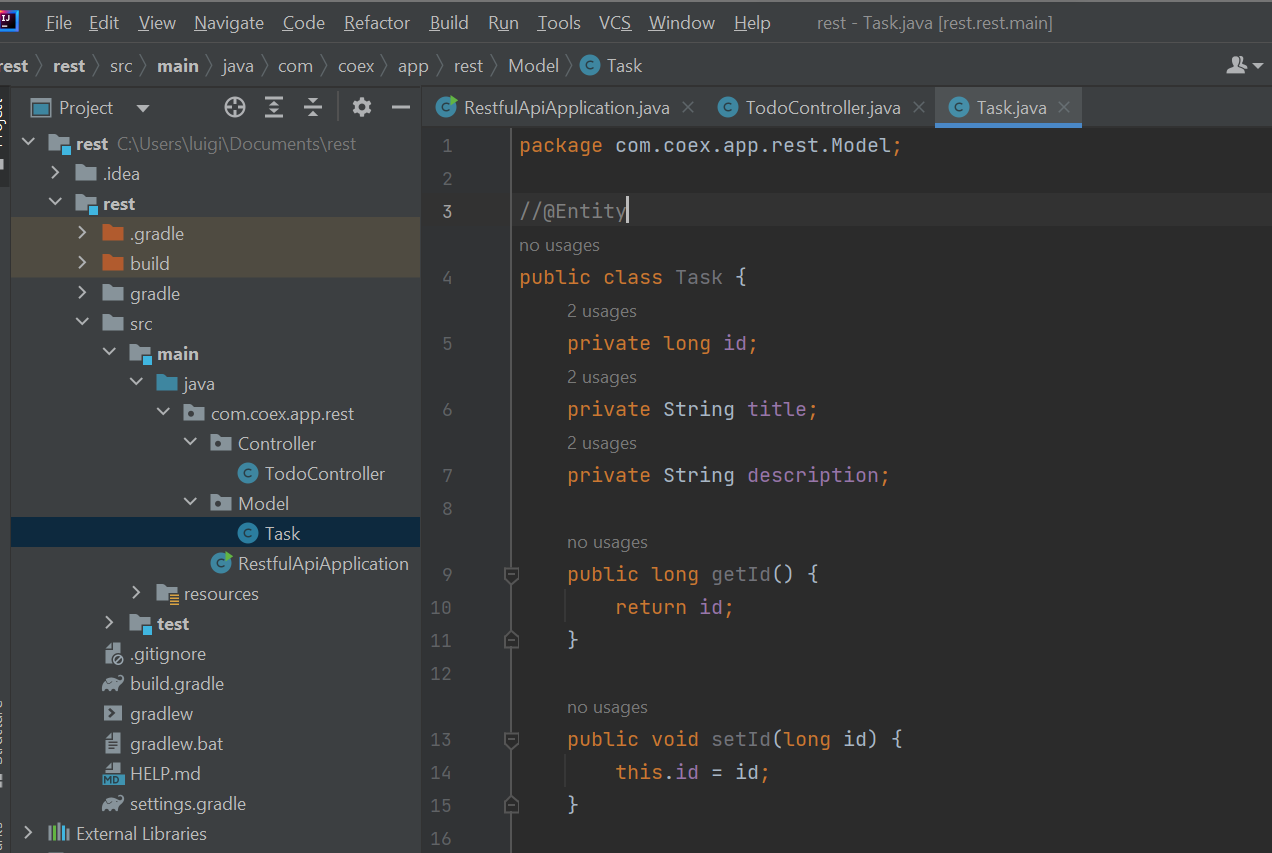
se le debe agregar la anotacion @Embeddable

se debe de hacer un “implements Serializable”

debe contener los atributos que conformen la llave compuesta

como en la mayoría de entidades debemos crear los getter y setters

Para nuestro proyecto lo que vamos a realizar ahora es la creación de los atributos que vamos a utilizar en nuestro proyecto .



Creamos los atributos y luego creamos los getter y setter del proyecto .

# **Creación del repository**

Vamos a crear nuestro repository el cual es el encargado de tener la comunicación con la base de datos y hacer un mapeo de las tablas que se usarán en el proyecto.

Spring Data Repositories

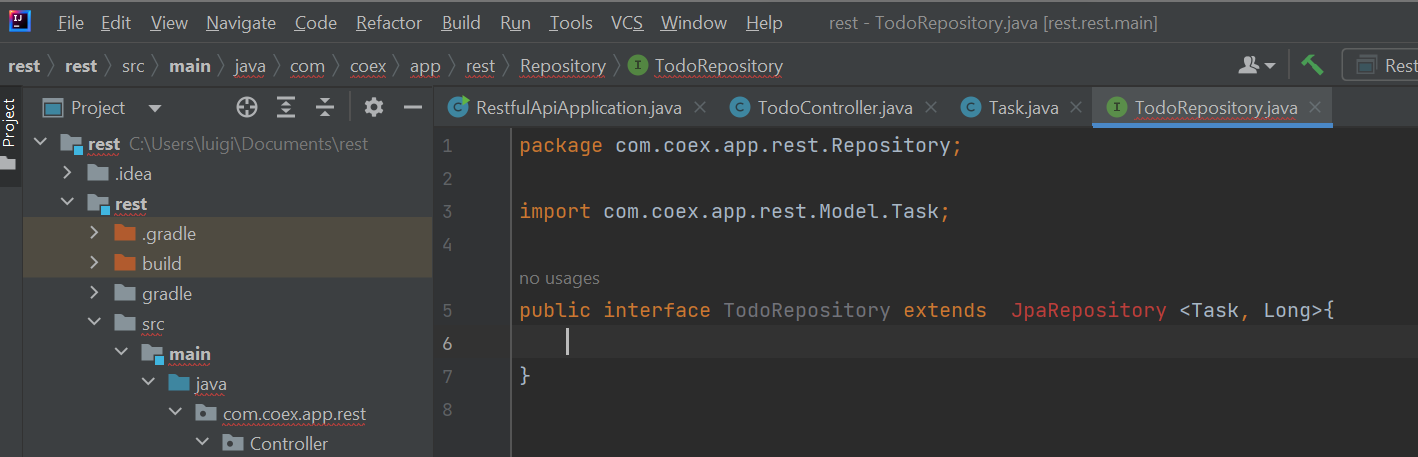
* Ahorra un montón de código y tiempo de implementación.
* Operaciones sin código en la base de datos
* Repositorios de Spring Data
* CrudRepository
* PagingAndSortingRepository
* CrudRepository

Existen Tres tipos de repositorios que de cierta manera nos permiten utilizar estos paquetes sin escribir tantas líneas de código con el fin de poder realizar tareas mucho más efectivas :

CrudRepository: Permite hacer las operaciones de CRUD.

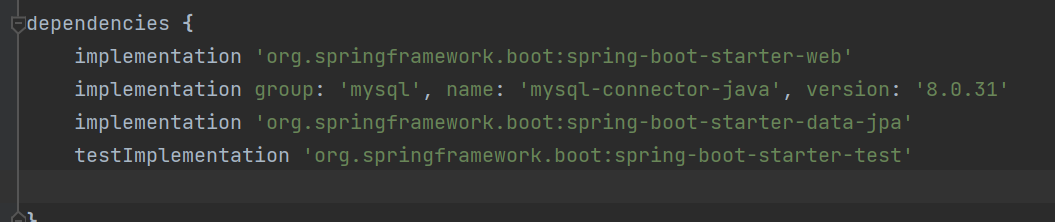
PagingAndSortingRepository: Nos permite hacer todo lo que hace el CrudRepository pero adicionalmente nos permite hacer tareas de ordenación y paginamiento de nuestro repositorio.

**JPARepository:** Nos permite hacer lo mismo que los dos anteriores, pero además nos permite hacer tareas de JPA específicas como Flush que combina o guarda todo en memoria sin que otras entidades o entornos vean esos cambios en la base de datos.



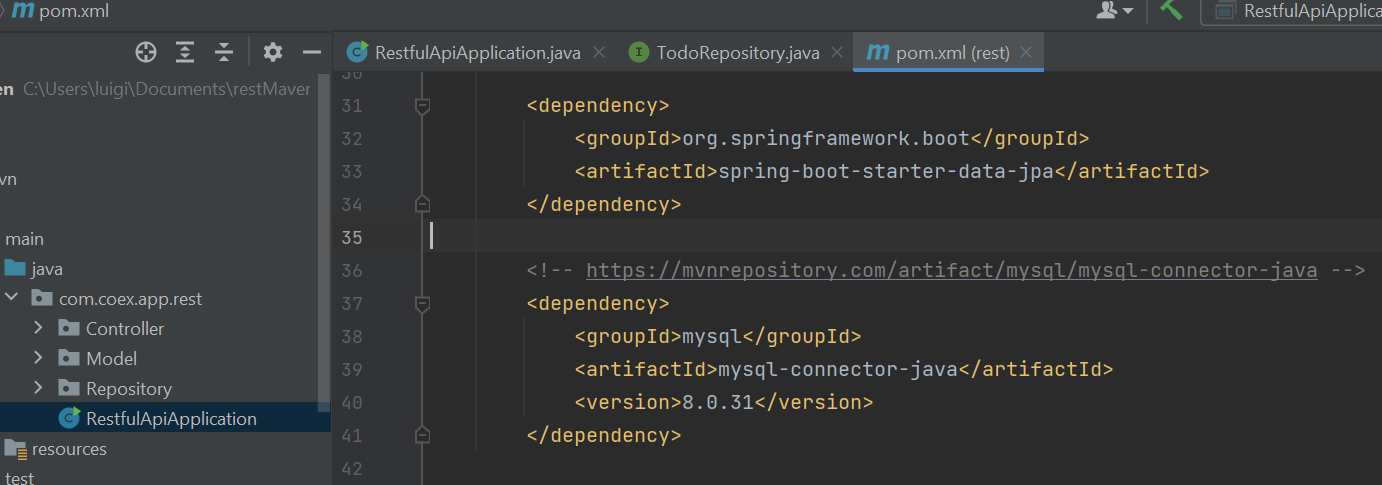
# **Uso de dependencias en maven**

Las dependencias las podemos encontrar por la página de repository maven.com

Para nuestro proyecto usaremos JpaRespository el cual nos servirá para realizar las diferentes consultas y movimientos de nuestros datos,

para importar las dependencias ya sea con maven o gradle

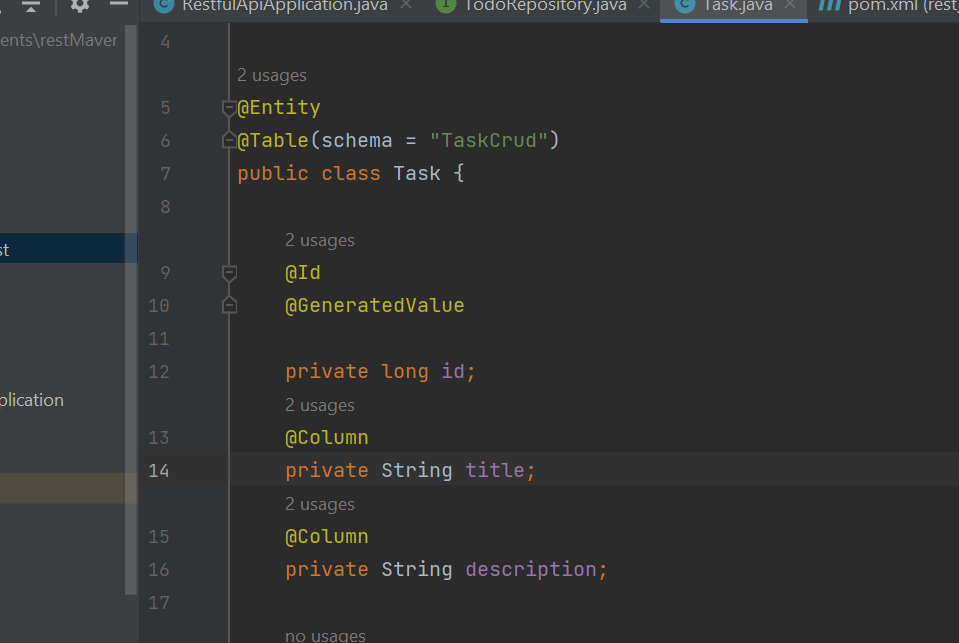
para maven modificar el archivo pom y para gradle el config gradle



Después de crear las dependencias damos clic en el botón de recargar dependencias y luego importamos la dependencia jpaRepository

nos vamos a dirigir al package modelo y allí vamos a realizar la modificación de algunas anotaciones para que nuestro sistema identifique cuales son los datos de la base de datos que contiene un primary key y un auto incrementables, así mismo darle los valores correspondientes a los valores de los atributos del sistema.

a los atributos de title y de descripción le agregamos el nombre de la anotacion@column para indicar que es una columna de la base de datos y en la parte superior agregamos el valor de @table para saber que estamos trabajando en esa base de datos.

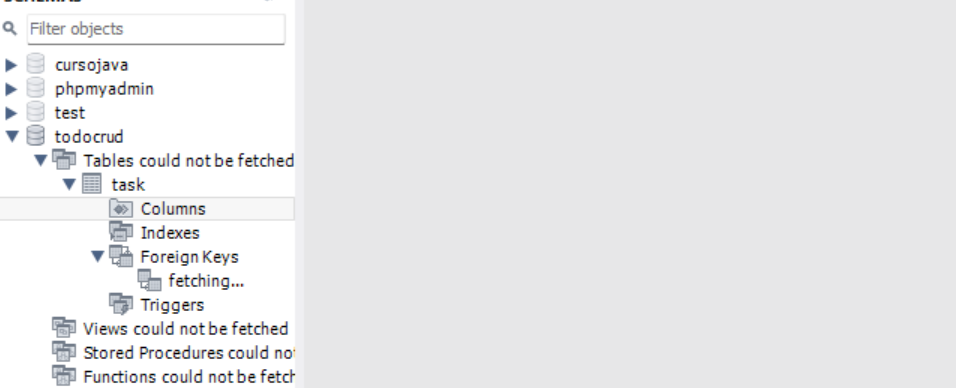


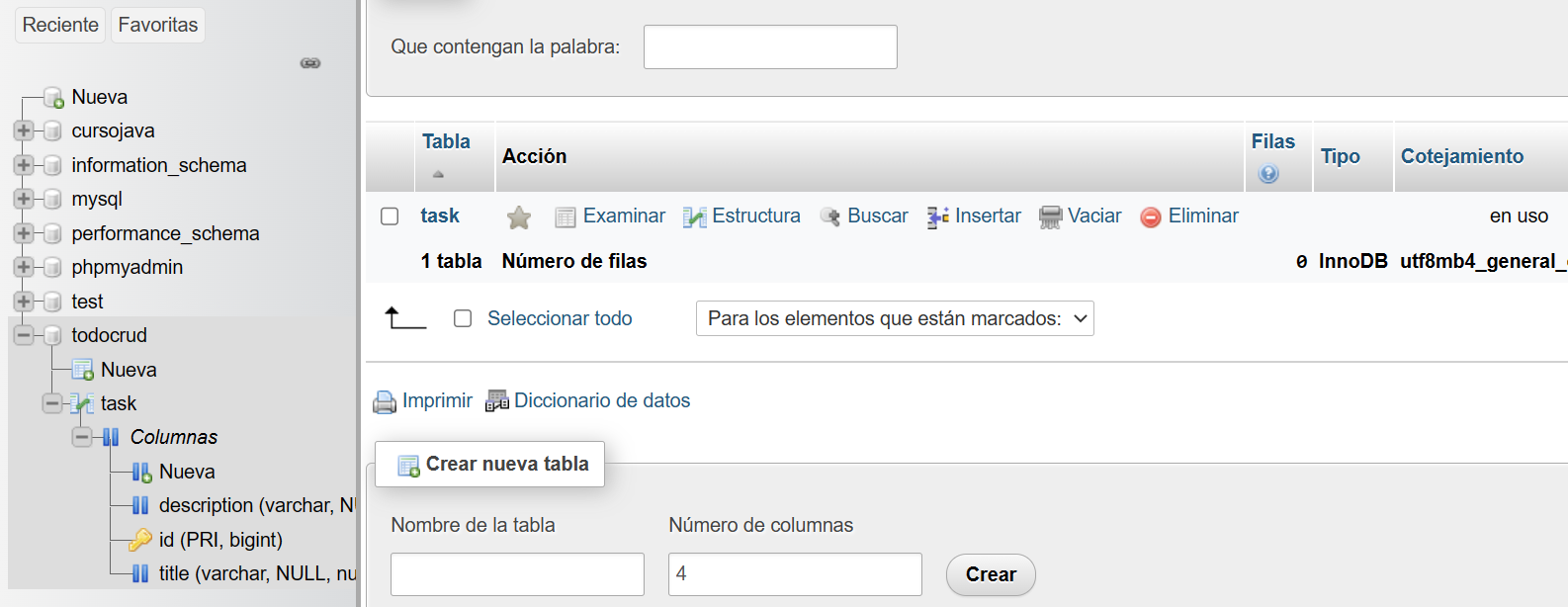
# **Datasource y conexión a la base de datos.**

Para la conexión a la base de datos vamos a usar workbench y para realizar el método que se encargará para la conexión vamos a realizar la modificación del archivo application.properties tanto en maven como en gradle.

# **¿ Qué es Workbench ?**

Creamos la base de datos shema en nuestro programa y hacemos la conexión correspondiente a la misma .

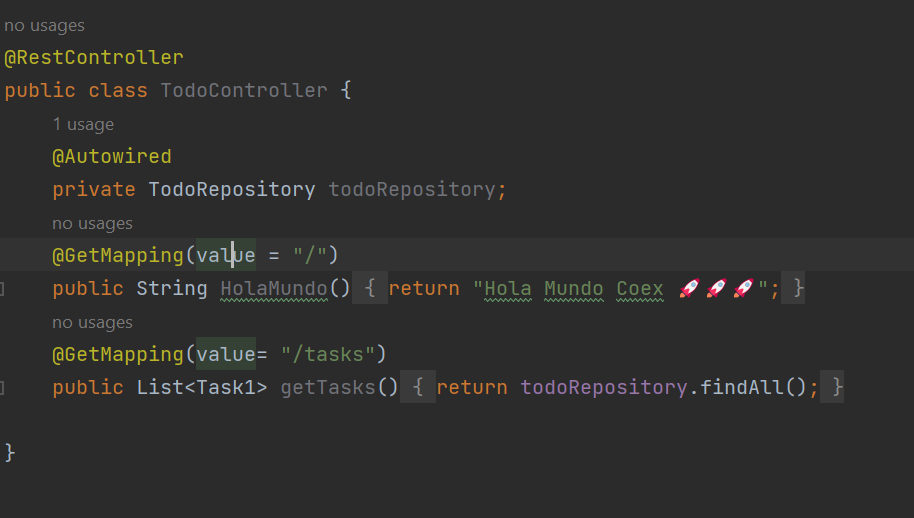




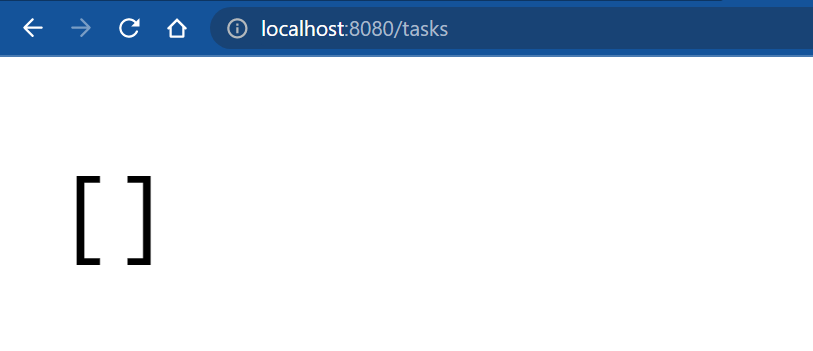
# **Creando las rutas web con @getMapping @postMapping @Autowired**

Para este caso vamos a utilizar las anotaciones @autowired esto con el fin de poder realizar la importación del repositorio a nuestro proyecto es decir traer la conexión a la base de datos.

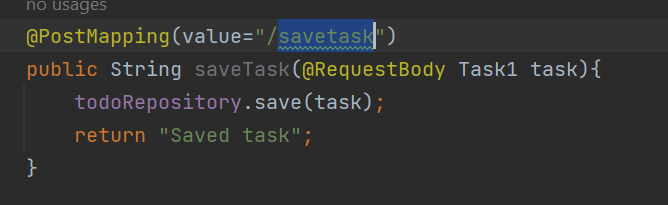
Realizamos la creacion de un metodo nuevo para traer todas las tareas eso lo hacemos con un List de java al todo repository



Ejecutamos el proyecto pero nos debe resultar un array vacío ya que no tenemos información dentro de nuestra base de datos.



Ahora vamos a crear un método para crear nuestras tareas.



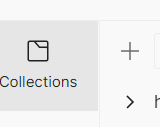
utilizamos el Post Mapping para el crear las tareas , le pasamos un request body para identificar que es para mostrarle dónde se va a almacenar la información que se está creando y esto nos devolverá un mensaje de Saved task.

# **Probando nuestro programa con postman**

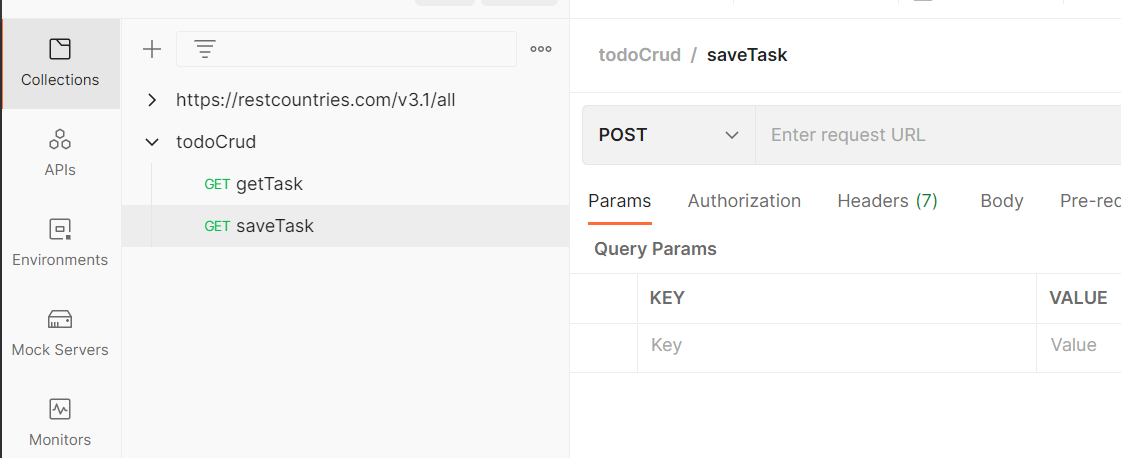
Para crear las tareas a nuestro sistema vamos a realizarlo en nuestro programa de Postman.

Primero abrimos Postman

clic en crear nueva conexión



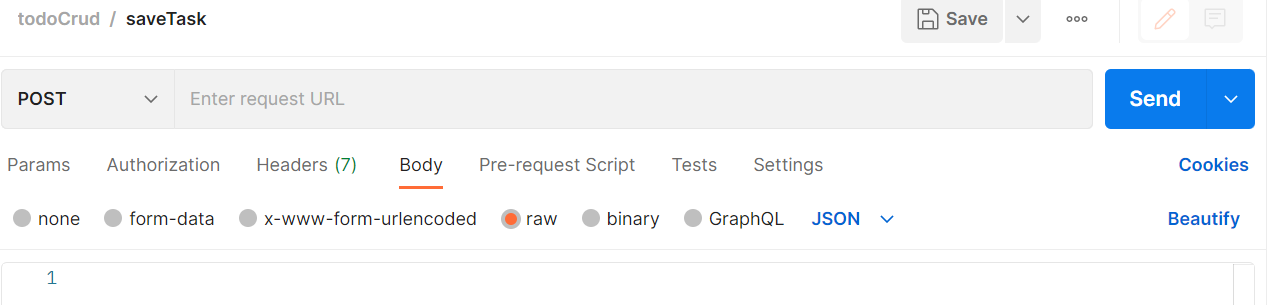
Crearemos una collection de nombre todoTask y crearemos dos request de nombre



getTas y saveTask

Ahora nos vamos a saveTask y en la pestaña body vamos a agregarle las anotaciones del método.

damos clic en body raw y agregamos JSON



# 

# **¿Qué es JSON?**

Un JSON en Javascript Object Notation formato que se usa para intercambiar o comunicar la información al servidor , es la sintaxis actual que leen casi todos los servidores modernos.

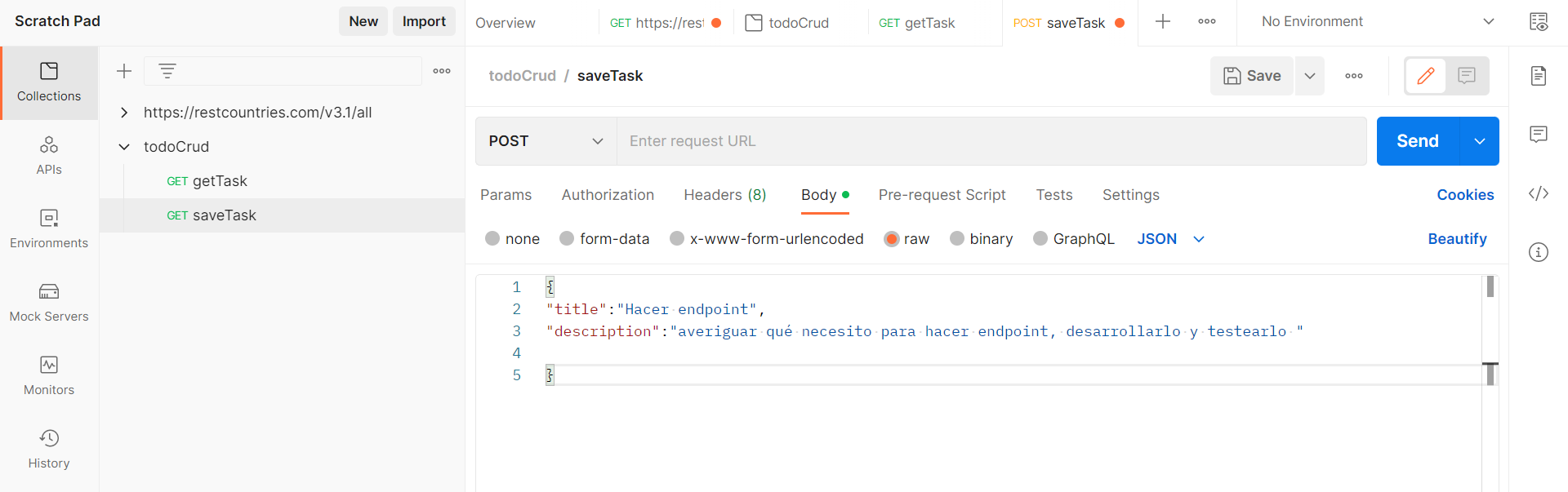
vamos a construir el JSON de la entrada de nuestra task.

{

"title":"Hacer endpoint",

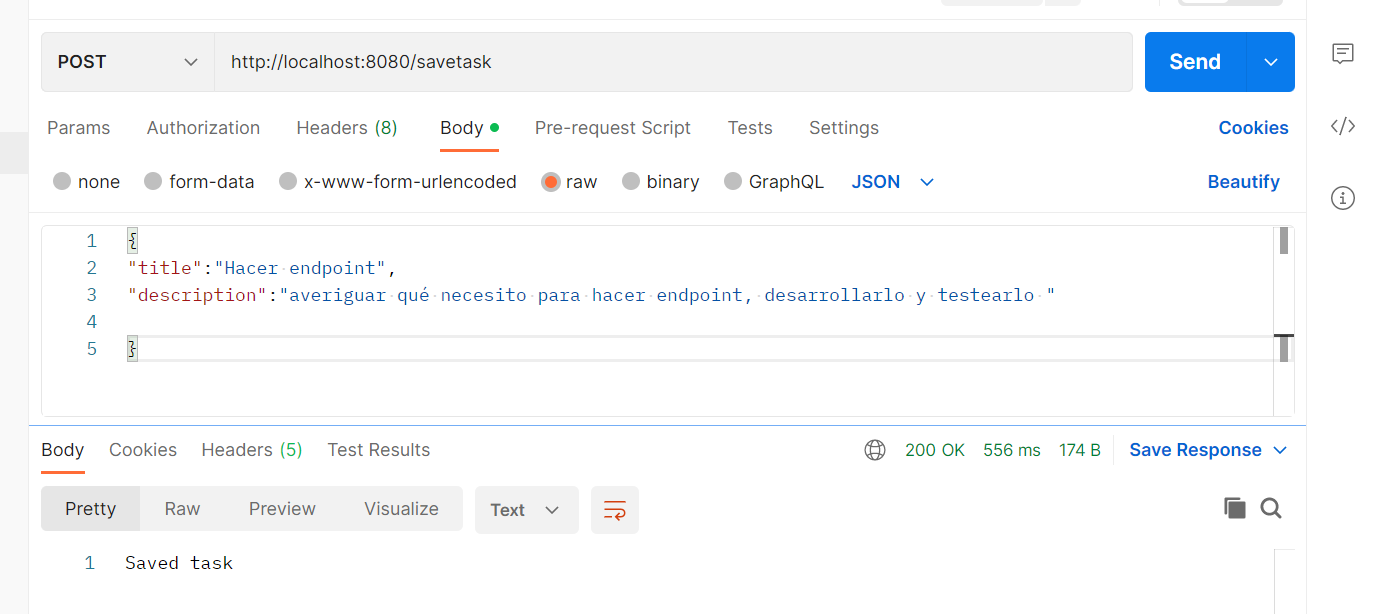
"description":"averiguar qué necesito para hacer endpoint, desarrollarlo y testearlo "

}



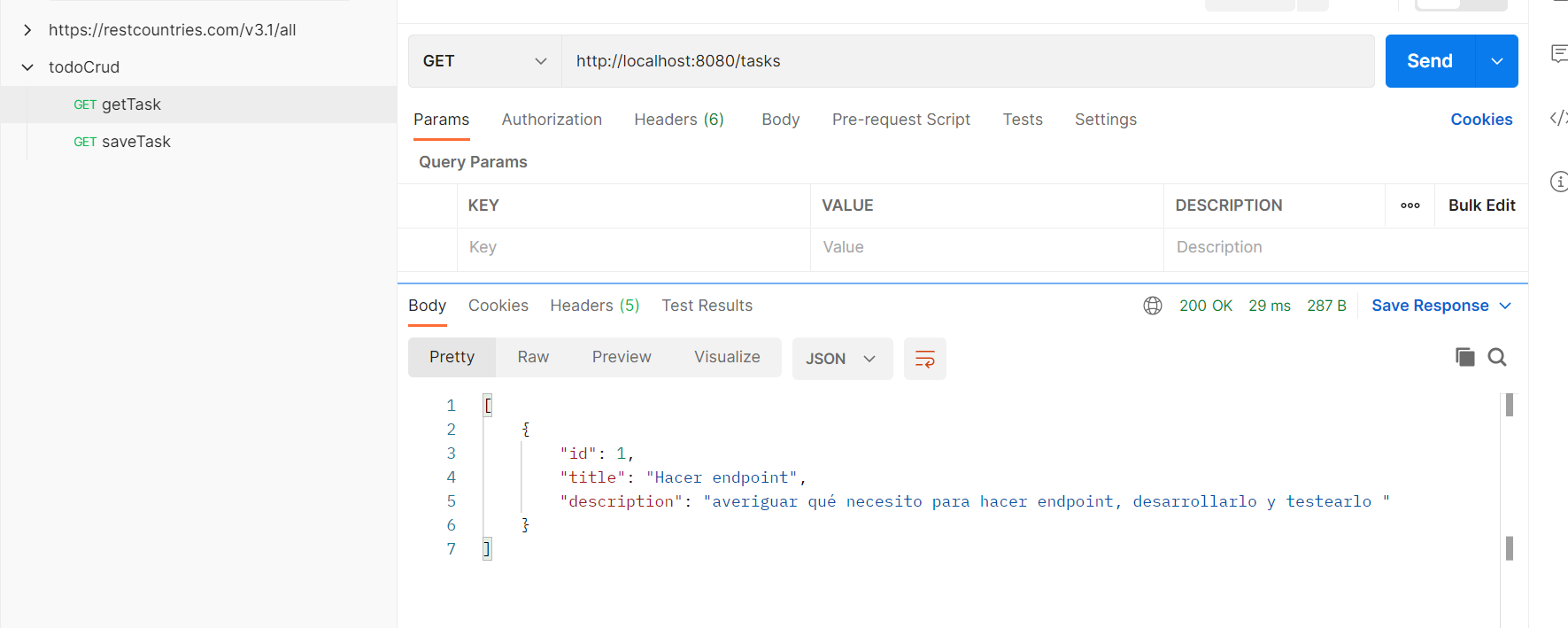
para probarlo agregamos el puerto <http://localhost:8080/savetask>

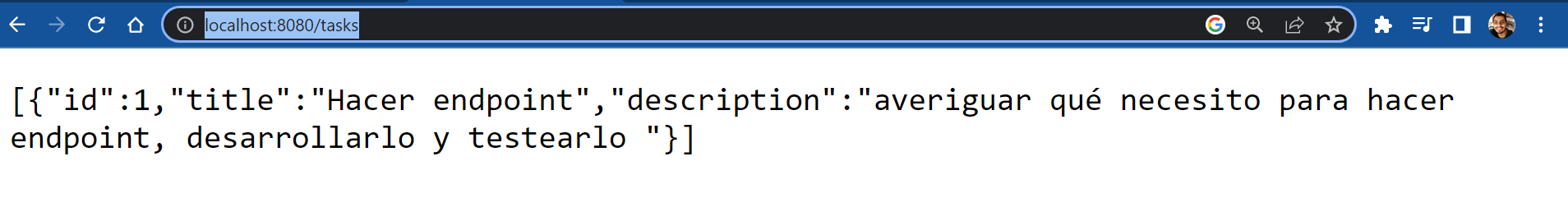
y damos clic en send si todo sale bien en la parte inferior llegara el mensaje Save task.



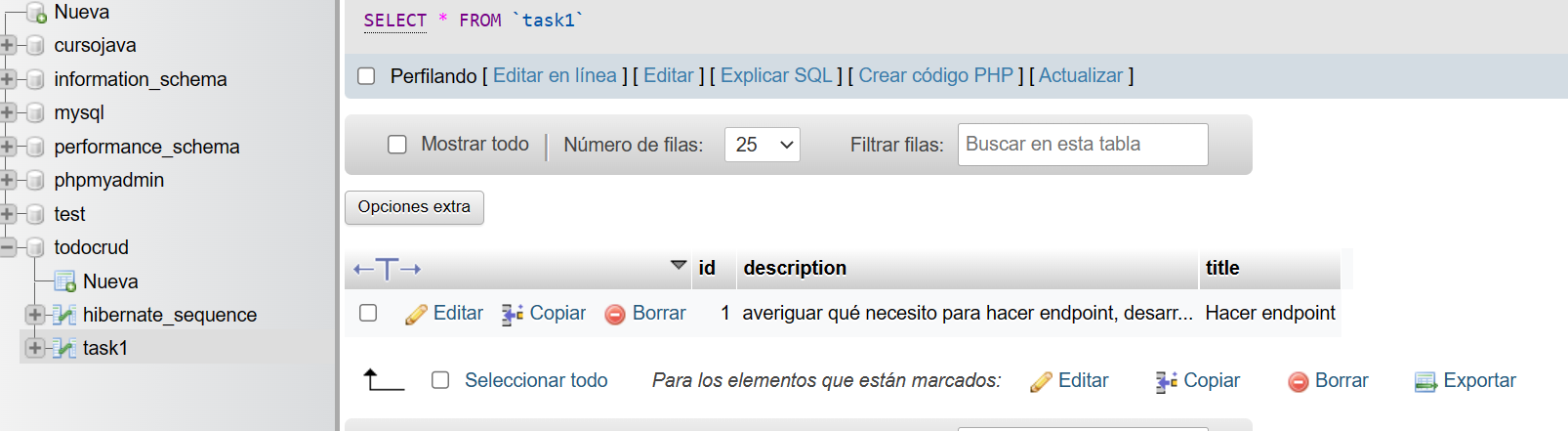
Si hacemos ahora la búsqueda de todos los elementos usando el método get nos tiene que devolver los datos almacenados en nuestra base de datos.

Hacemos la prueba usando postman y luego en el navegador.





Miramos en la base de datos y efectivamente , tenemos todo almacenado.



**Construir el crud del proyecto**

Dentro de la construcción de un crud del proyecto debemos tener en cuenta los diferentes métodos tales como :

**POST = Create**

**GET = Read**

**PUT = Update**

**DELETE = Delete**

**ya tenemos el POST Y EL GET vamos a completar el update y el Delete.**

**@PutMapping(value="/update/{id}")**

**public String updateTask(@PathVariable long id, @RequestBody Task task){**

**Task updatedTask = todoRepository.findById(id).get();**

**updatedTask.setTitle(task.getTitle());**

**updatedTask.setDescription(task.getDescription());**

**todoRepository.save(updatedTask);**

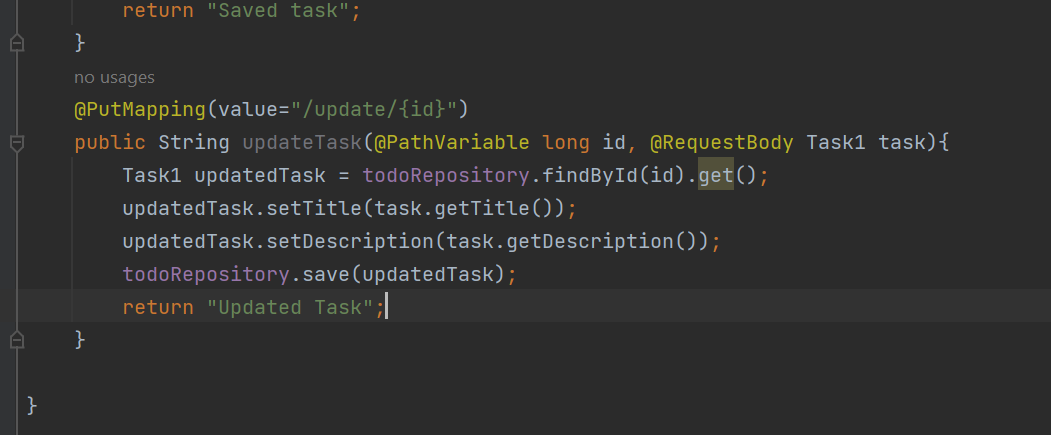
**return "Updated Task";**

**}**

Como vemos este método recibe un PutMapping que en su ruta recibe el nombre y el id del elemento que vamos a eliminar , este recibe una @path que se encargará de tomar la variable del id y luego tomar el request del cuerpo .

Creamos una variable de nombre updateTask y le pasamos el set para traer los datos almacenados dentro de nuestra base de datos según el id que tiene cada tarea.

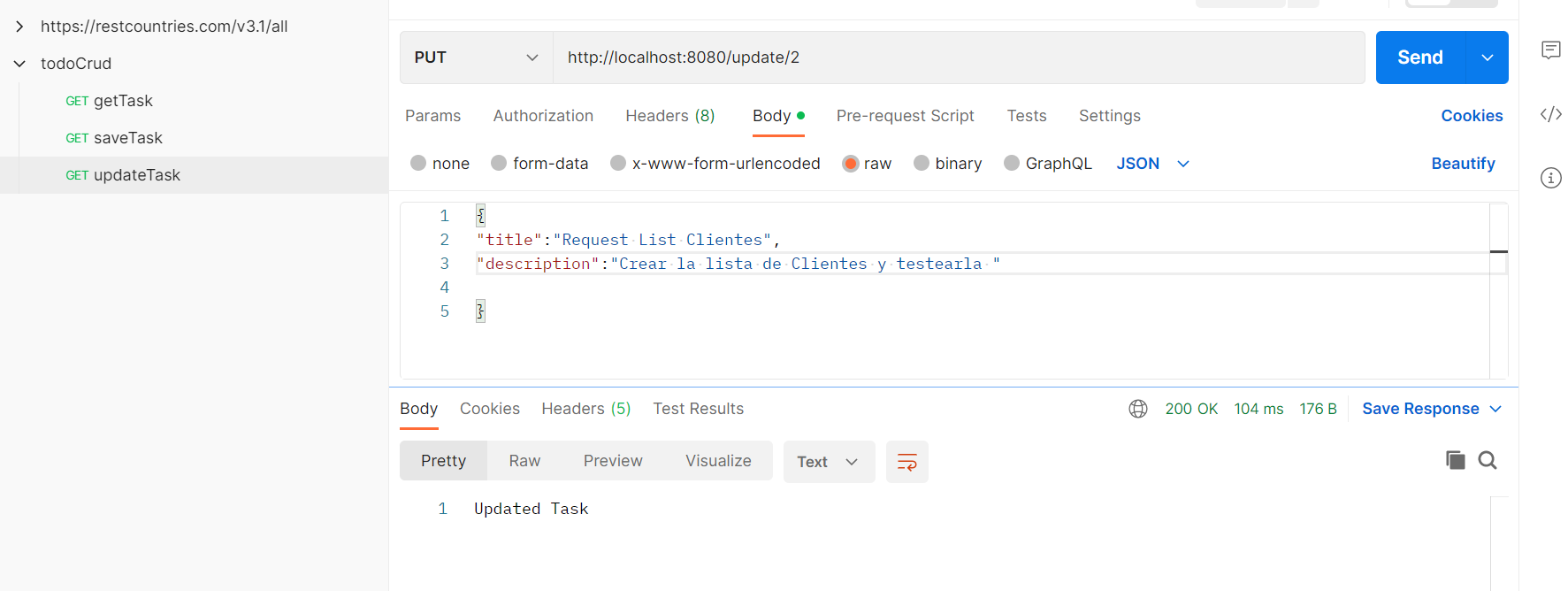
Hacemos el llamado del todoRepository y actualizamos la variable updateTask , retornamos un string que diga update Task .



Hacemos la prueba pasándola por postman y revisamos que los datos que se encuentran almacenados en las tareas se encuentren editadas correctamente.

Creamos una nueva tarea para poder hacer las pruebas.

# 



Creamos el método de Delete.

@DeleteMapping(value="delete/{id}")

public String deleteTask(@PathVariable long id){

Task deletedTask = todoRepository.findById(id).get();

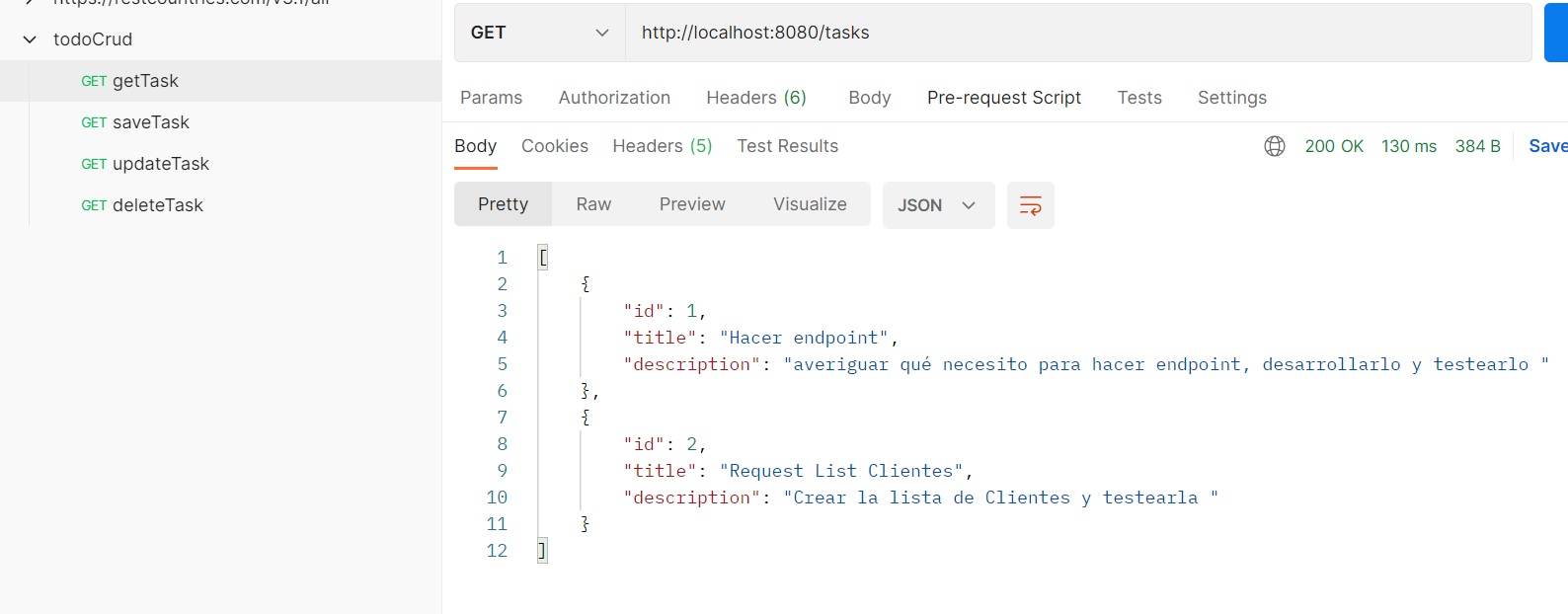
todoRepository.delete(deletedTask);

return "Deleted Task";

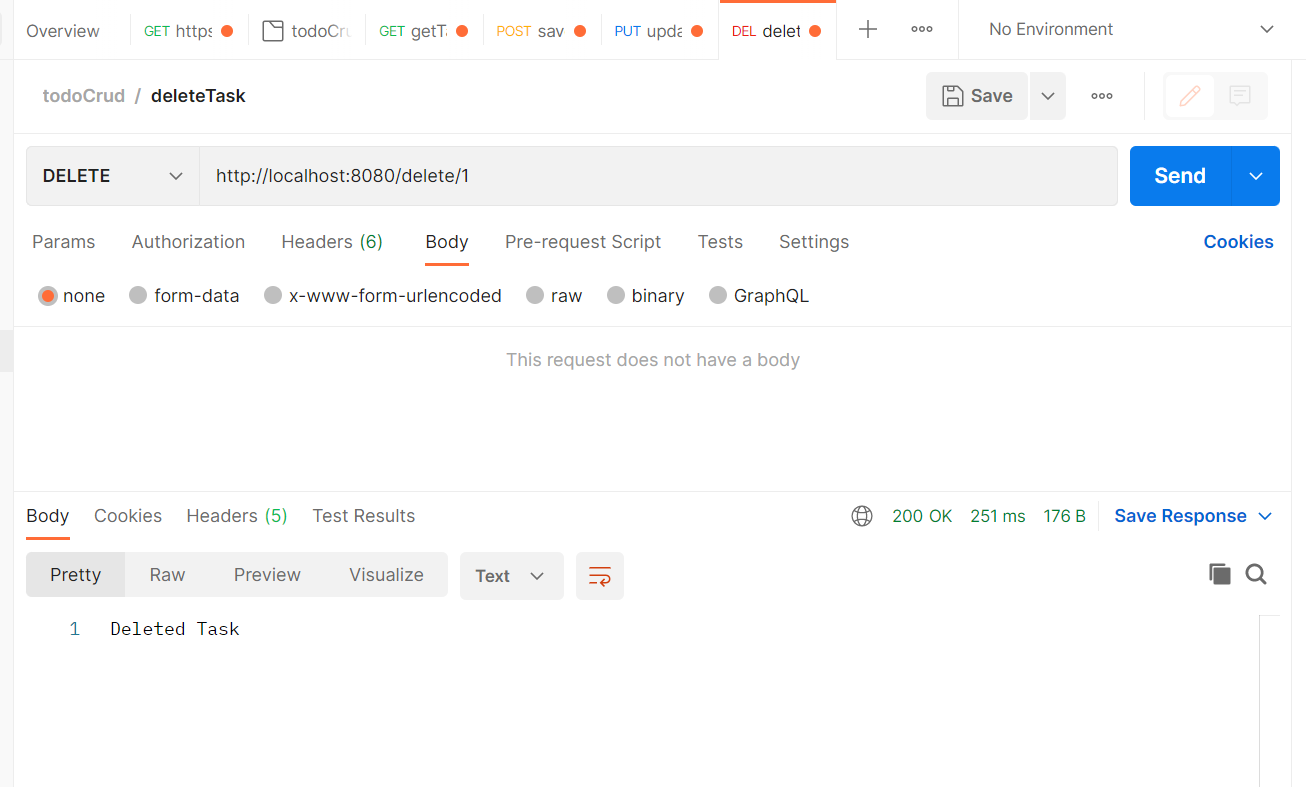
}

como vemos en este método tenemos los siguientes datos, llamamos la anotación de delete Mapping que es la encargada de darle ruta al método , luego este hace el llamado del método que solicita un pathVariable que seria el id esto con el fin de poder traer ese elemento y eliminarlo , al final retornamos un string que diga Deleted task.

Como vemos tenemos una lista de dos tareas.



y vamos a eliminar la tarea número 1.



Bibliografía

Alien Explorer (23-07-2022). *Crear una api rest en java* . Alien Explorer. <https://www.youtube.com/watch?v=OkD1wBpPsNM&ab_channel=AlienExplorer>

Platzi(22-12-2022). *Dependencias en java*. Platzi.

<https://platzi.com/clases/2317-spring-boot/38163-que-es-una-dependencia/>

Arquitectura java (3-12-2022). *Invocando un servicio rest*  , arquitectura en java

<https://www.arquitecturajava.com/java-httpclient-invocando-un-servicio-rest/>

autoservicios java (3-12-2022). *Qué es spring boot Alto Servicios*.

<https://altoservicios.com/crear-paginas-web/que-es-spring-boot/#:~:text=Spring%20Boot%20es%20una%20tecnolog%C3%ADa,despliegue%20del%20servicio%20de%20aplicaciones>.

curso básico de java (3-12-2022). *curso básico gratuito de java*.

<https://codigofacilito.com/cursos/JAVA>