

# Configuración y Ejecución (Prueba Técnica Fintonic)

Autor: Sebastián Molina

<a href="#">Configuración</a>	1
<a href="#">Ejecución</a>	4
<a href="#">Nota</a>	10

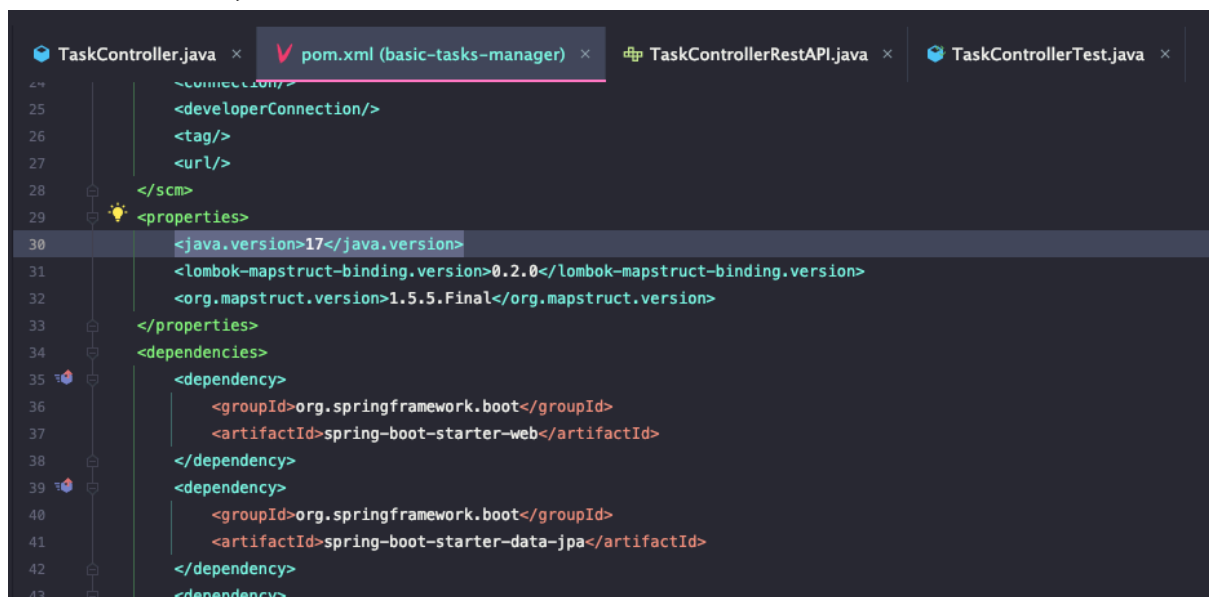
## Configuración

Lo primero que debemos tener en cuenta es que tengamos java 17 instalado, vamos a la consola y ejecutamos el siguiente comando

java --version

```
openjdk 17.0.10 2024-01-16 LTS
OpenJDK Runtime Environment Corretto-17.0.10.7.1 (build 17.0.10+7-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.10.7.1 (build 17.0.10+7-LTS, mixed mode, sharing)
```

Si no es el caso podemos tener la tentación de ir al pom y modificarlo para que acepte otra versión como java 11



Luego vamos a settings del IDE y observamos que tenga la misma versión que hemos puesto en el pom.

Pero esto no nos servirá ya que el proyecto se ha hecho con Spring Boot 3 que solo soporta Java 17, por lo que debemos tener Java 17 LTS.

También debemos ver que tengamos MySQL instalado

```
cbash.ml@SML-MP:~ $ mysql -V
mysql Ver 8.0.28 for macos12.2 on x86_64 (Homebrew)
```

En caso de que no tengamos ni java ni mysql, podemos instalarlo desde las siguientes páginas

Java 17 LTS:

<https://adoptium.net/es/marketplace/?version=17>

MySQL (version LTS):

<https://downloads.mysql.com/archives/community/>

Si se desea se puede instalar MySQL Workbench

<https://downloads.mysql.com/archives/workbench/>

Si lo hacemos desde Windows tendremos el MySQL Installer for Windows

<https://dev.mysql.com/downloads/>

En el cual al instalar solo hace falta que instalemos el mysql server y el workbench (si se desea)

una vez tengamos ambos instalados,  
entramos con el usuario root a mysql

```
cbash.ml@SML-MP:~ $ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 183
Server version: 8.0.28 Homebrew

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

y creamos la siguiente base de datos con el siguiente comando

`CREATE SCHEMA `basic_task_db` DEFAULT CHARACTER SET utf8 ;`

creamos también un nuevo usuario test, con password Test-localhost 1234  
como aparece en la configuración de las propiedades el application.properties

```
Spring Boot configuration files are supported by IntelliJ IDEA Ultimate
1  spring.application.name=basic-tasks-manager
2
3  spring.datasource.url = jdbc:mysql://localhost:3306/basic_tasks_db
4  spring.datasource.username =test
5  spring.datasource.password =Test-localhost 1234
6  spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
7  spring.jpa.database-platform = org.hibernate.dialect.MySQLDialect
8  spring.jpa.show-sql = true
9
10 spring.jpa.hibernate.ddl-auto=create
```

ejecutando el siguiente comando

```
CREATE USER 'test'@'localhost' IDENTIFIED BY 'Tes-localhost 1234';
```

luego damos permisos al nuevo usuario sobre el schema basic\_task\_db  
con el siguiente comando

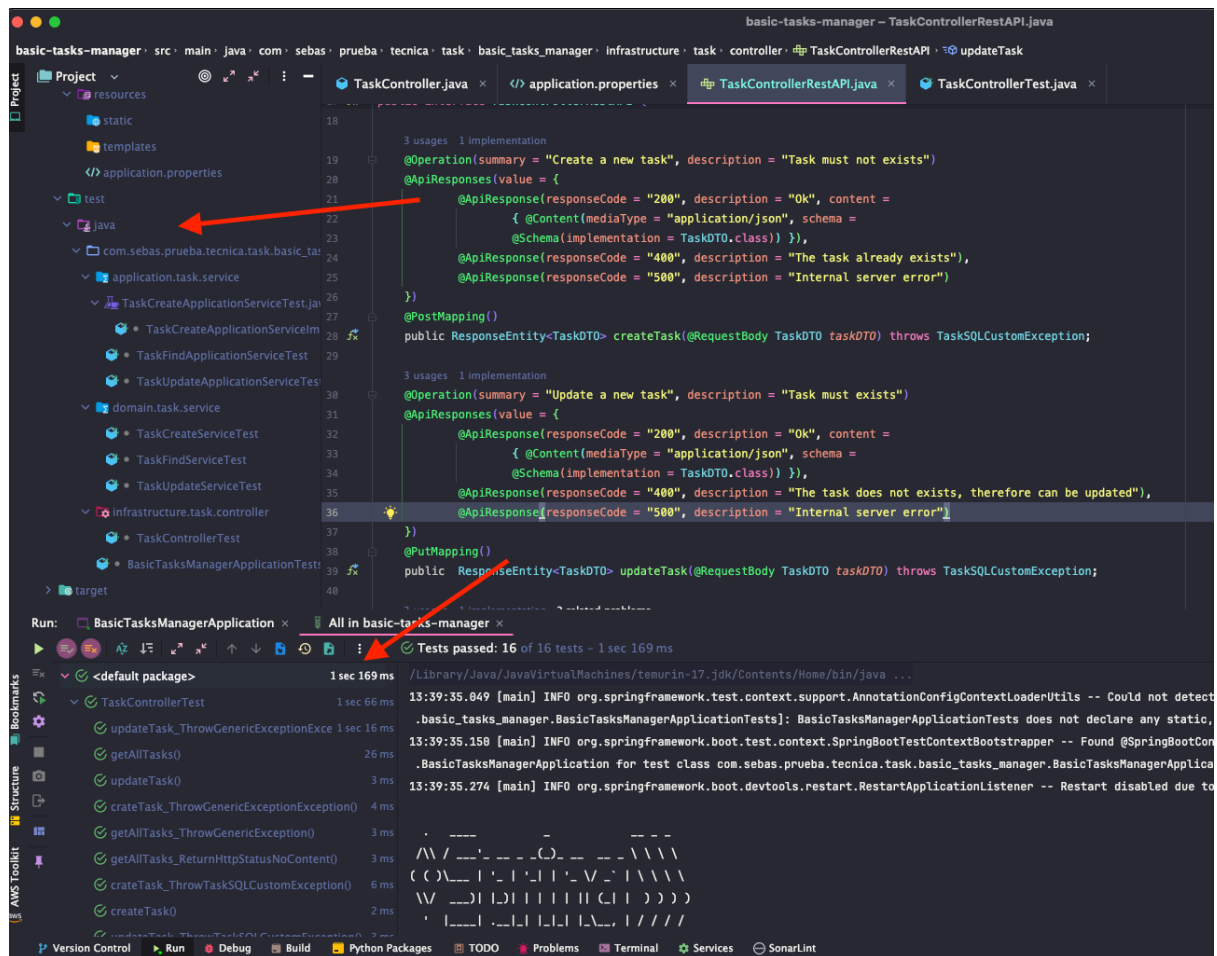
```
GRANT ALL PRIVILEGES ON basic_task_db.* TO 'test'@'localhost';
```

Con esto ya podremos ejecutar la aplicación.

## Ejecución

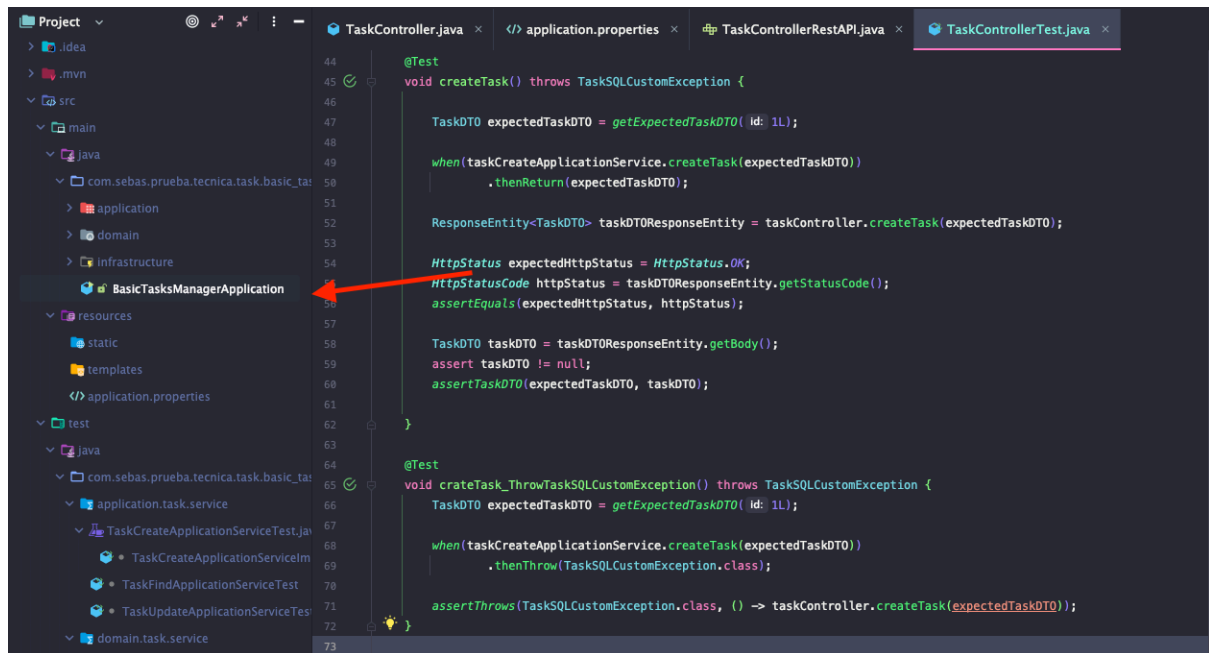
Empezamos por ir al IDE, en este caso utilizaremos IntelliJ IDEA, lo primero que haremos antes de levantar el servidor embebido tomcat, para posteriormente levantar la aplicación será ejecutar los tests para ver que ninguno falla.

vamos a la estructura del proyecto buscamos test >> java (click derecho) >> run 'All Test' y en la parte izquierda inferior podemos ver que se han ejecutado todos los test correctamente.

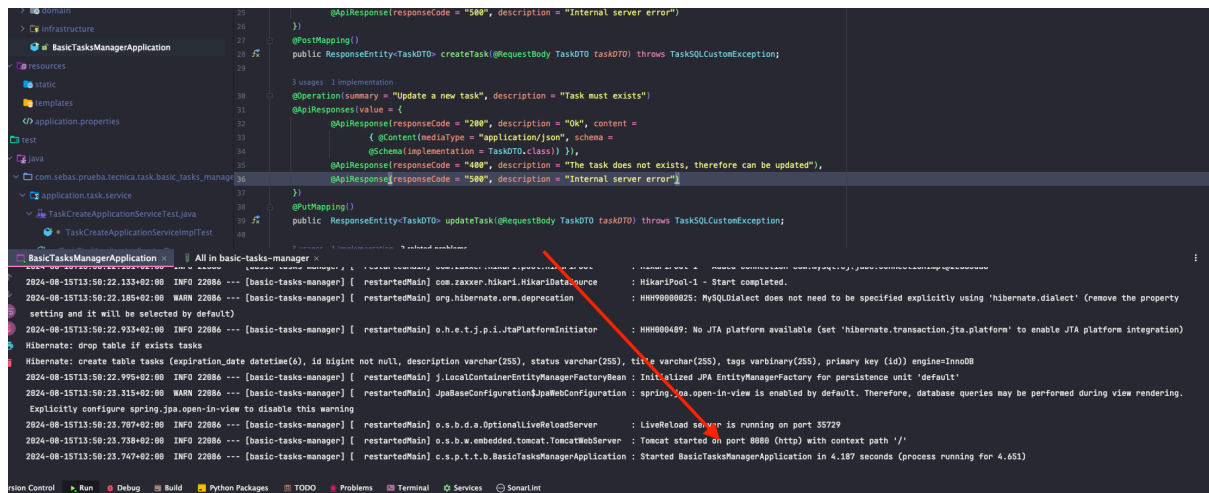


Ahora vamos a ejecutar la aplicación, buscamos en la estructura principal del proyecto la clase BasicTaskManagerApplication que se encuentra en com.sebas.prueba.tecnica.task.basic\_tasks\_manager

damos click derecho en ella y luego en Run BasicTaskManagerApplication.. Main()

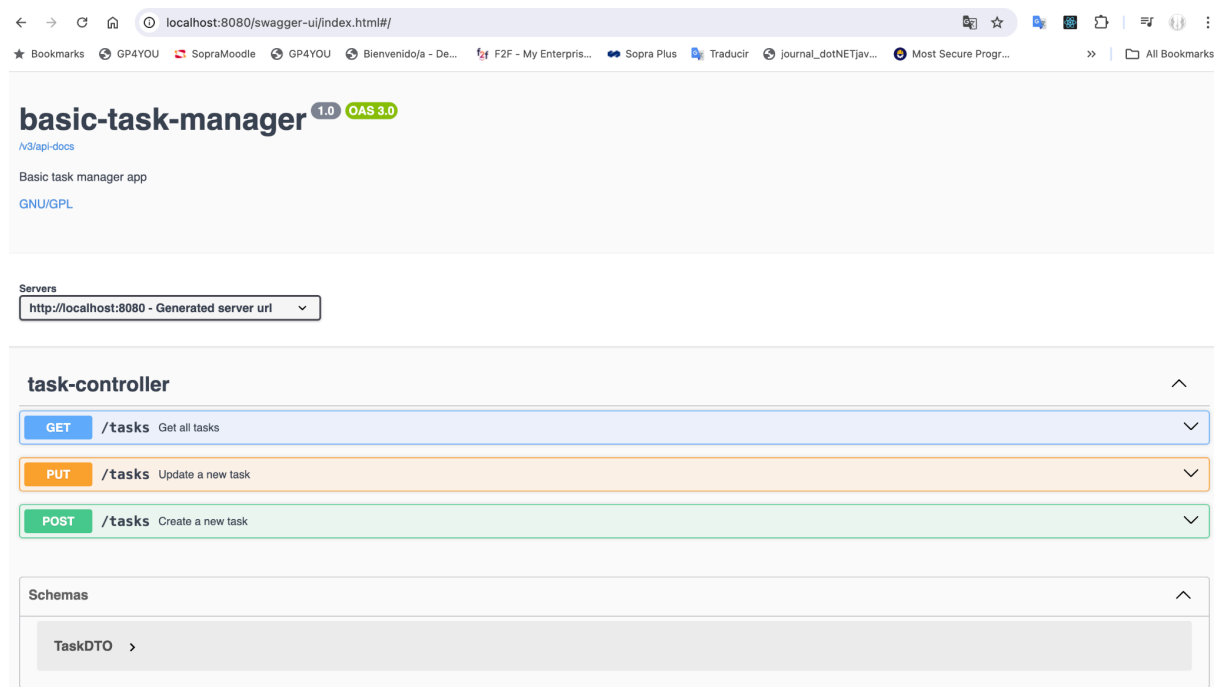


y una vez vemos que ha arrancado bien la aplicación



vamos a el siguiente enlace, que pertenece al swagger

<http://localhost:8080/swagger-ui/index.html#/>



Ahora empezamos ejecutando uno de los endpoint donde el json que vamos a usar será principalmente el siguiente

```
{
  "id": 1,
  "title": "Testing",
  "description": "Testing descirption",
  "expirationDate": "01/01/2099 00:00:00",
  "status": "Pending",
  "tags": [
    "tag1",
    "tag2"
  ]
}
```

vamos al endpoint para crear tareas y pegamos el json

**POST** /tasks Create a new task

Task must not exists

**Parameters** Cancel Reset

No parameters

**Request body** required application/json

```
{
  "id": 1,
  "title": "Testing",
  "description": "Testing description",
  "expirationDate": "01/01/2099 00:00:00",
  "status": "Pending",
  "tags": [
    "tag1",
    "tag2"
  ]
}
```

Execute Clear

damos a execute y tenemos una respuesta 200 OK con el json de respuesta

**Server response**

Code	Details
200	<div><b>Response body</b><pre>{   "id": 1,   "title": "Testing",   "description": "Testing description",   "expirationDate": "2099-01-01T00:00:00",   "status": "Pending",   "tags": [     "tag1",     "tag2"   ] }</pre><span>Download</span></div> <div><b>Response headers</b><pre>connection: keep-alive content-type: application/json date: Thu, 15 Aug 2024 12:00:01 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre></div>

Ahora probamos el de actualizar, en este caso actualizaremos el estado.

**PUT** /tasks Update a new task

Task must exists

**Parameters** Cancel Reset

No parameters

**Request body** required application/json

```
{
  "id": 1,
  "title": "Testing",
  "description": "Testing description",
  "expirationDate": "01/01/2099 00:00:00",
  "status": "Done",
  "tags": [
    "tag1",
    "tag2"
  ]
}
```

Execute Clear

damos a execute y tenemos una respuesta 200 OK con el json de respuesta

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "title": "Testing",
  "description": "Testing description",
  "expirationDate": "2099-01-01T00:00:00",
  "status": "Done",
  "tags": [
    "tag1",
    "tag2"
  ]
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Thu, 15 Aug 2024 12:02:22 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

ahora vamos al endpoint para obtener todos las tareas, creamos una tarea demás en el camino para que nos salga

GET /tasks Get all tasks

Tasks must exists

Parameters

Cancel

No parameters

Execute

Clear

damos a execute y tenemos una respuesta 200 OK con el json que contiene la lista de tareas

Server response

Code

Details

200

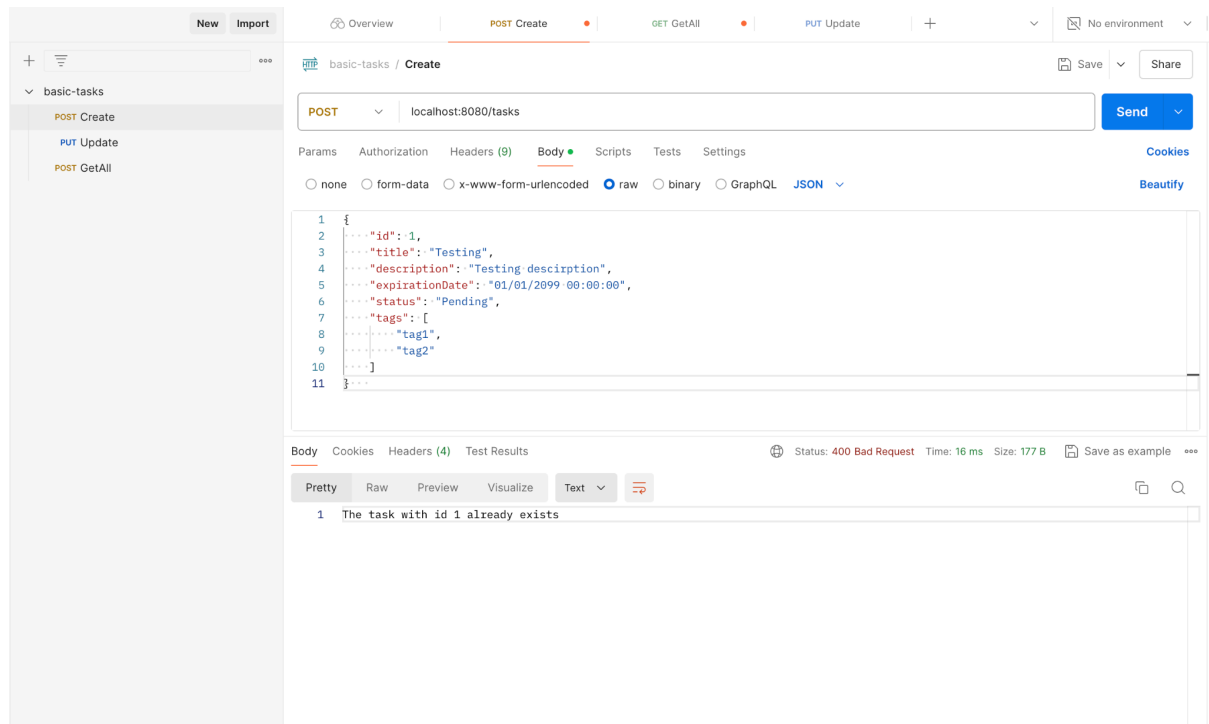
Response body

```
[
  {
    "id": 1,
    "title": "Testing",
    "description": "Testing description",
    "expirationDate": "2099-01-01T00:00:00",
    "status": "Done",
    "tags": [
      "tag1",
      "tag2"
    ]
  },
  {
    "id": 2,
    "title": "Testing",
    "description": "Testing description",
    "expirationDate": "2099-01-01T00:00:00",
    "status": "Pending",
    "tags": [
      "tag1",
      "tag2"
    ]
  }
]
```

Download

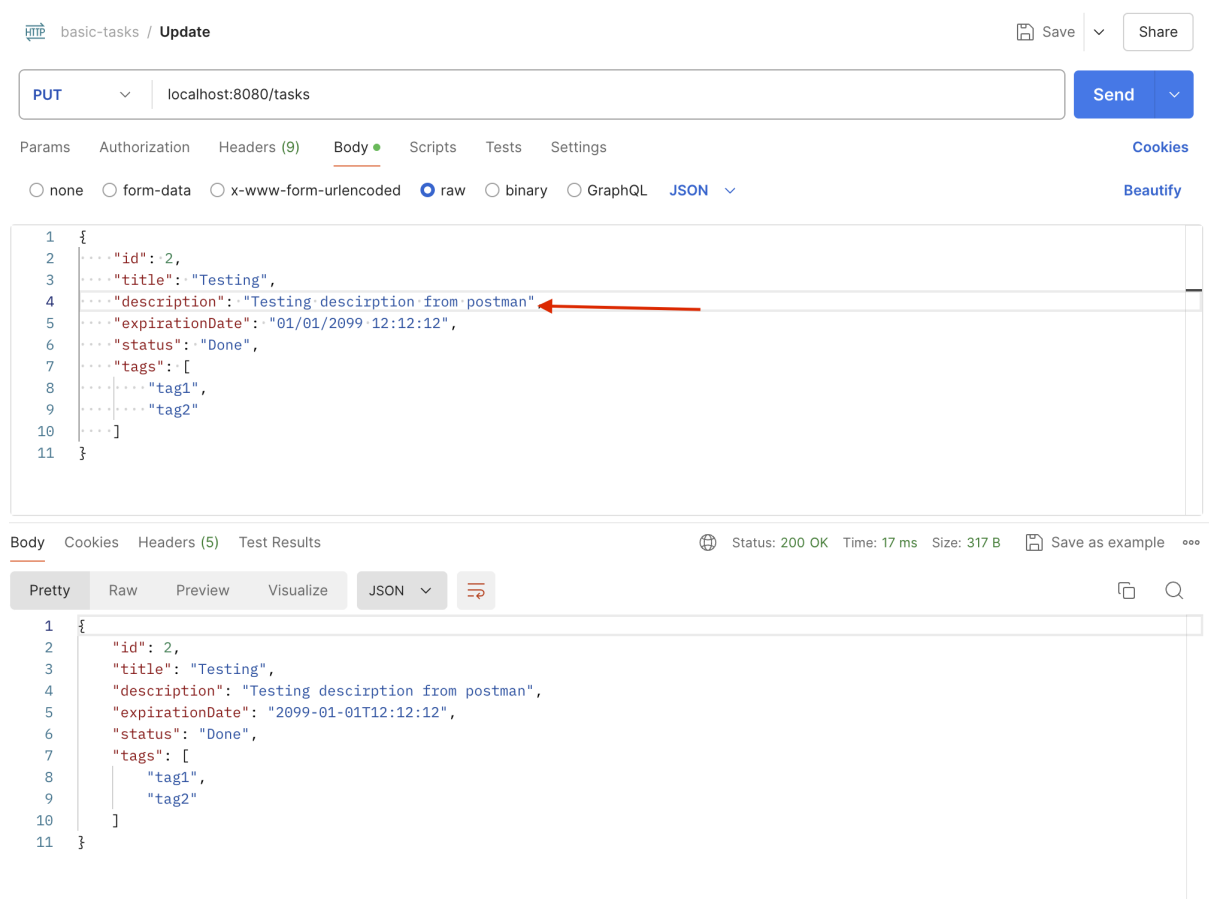


Luego si se desea también lo podemos probar desde postman



En este caso nos da un 400 Bad Request porque ya existe dicho id.

probemos ahora un update cambiando la descripción



ahora obtengamos la lista de todas las tareas para ver si se efectuó la modificación de la descripción

HTTP basic-tasks / GetAll Save Share

GET localhost:8080/tasks Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookie


Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 11 ms Size: 460 B Save as example

Pretty Raw Preview Visualize JSON

```
2  {
3    "id": 1,
4    "title": "Testing",
5    "description": "Testing descirption",
6    "expirationDate": "2099-01-01T00:00:00",
7    "status": "Done",
8    "tags": [
9      "tag1",
10     "tag2"
11   ],
12 },
13 {
14   "id": 2,
15   "title": "Testing",
16   "description": "Testing descirption from postman",
17   "expirationDate": "2099-01-01T12:12:12",
18   "status": "Done",
19   "tags": [
20     "tag1",
21     "tag2"
22   ]
23 }
24 ]
```



## Nota

No se realizó con mongodb ya que nunca he utilizado está base de datos en un proyecto y por lo mismo no la he configurado nunca para spring boot, por lo que no sabría estimar cuánto me podría llevar configurarla y aprender a usarla para realizar la prueba técnica con este motor de base de datos, que entiendo es noSQL y documental.