# Restful API using Spring Rest - Spring Data JPA - H2 with Spring Boot Example

Contenido

[Restful API using Spring Rest - Spring Data JPA - H2 with Spring Boot Example 1](#_Toc524438404)

[Clase SpringRestAndDataJpaWithSpringBootApplication 8](#_Toc524438405)

[ENDPOINTS 9](#_Toc524438406)

[Clase Employee 10](#_Toc524438407)

[Clase EmployeeRestController 11](#_Toc524438408)

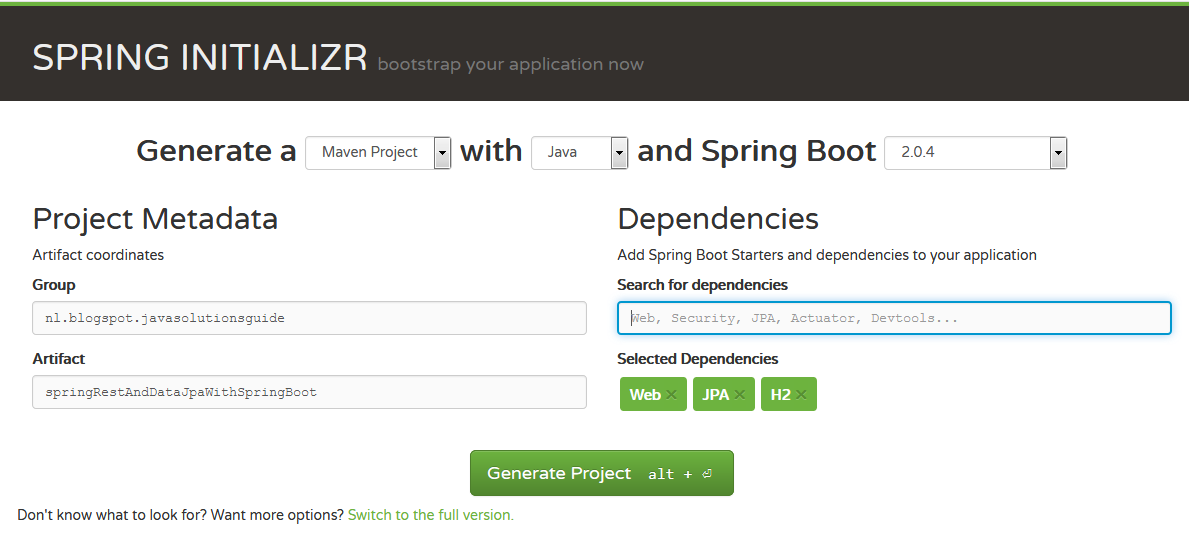
[Clase EmployeeService 12](#_Toc524438409)

[Clase EmployeeServiceImpl 13](#_Toc524438410)

[Clase EmployeeRepository 14](#_Toc524438411)

[application.properties 15](#_Toc524438412)

[Using Swagger to test the Restful endpoints 22](#_Toc524438413)



**Group**: nl.blogspot.javasolutionsguide

**Artifact**: springRestAndDataJpaWithSpringBoot

Y las **dependencias**: Web, JPA y H2

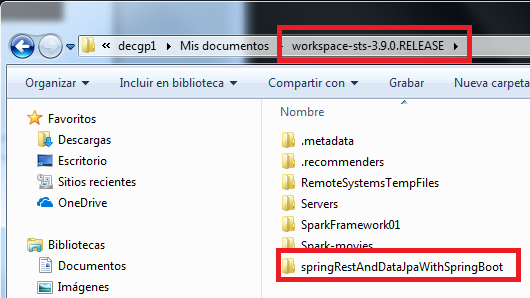
Pulsamos sobre **Generate Project**

Y se nos genera el proyecto:

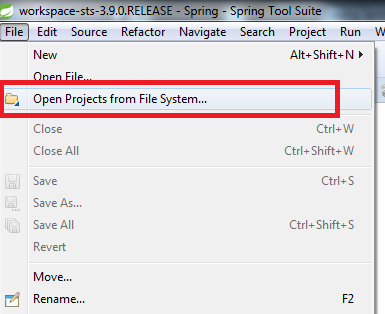


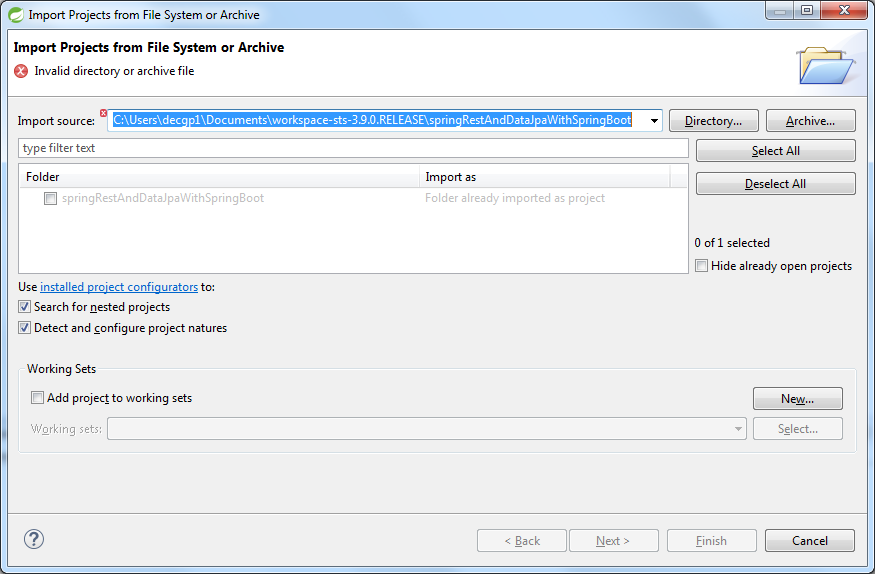
Para imporarlo en el STS (eclipse de spring)

Extremos el fichero y lo copiamos en el Workspace:



Desde el STS, lo abrimos:





Usamos el siguiente pom.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>nl.blogspot.javasolutionsguide</groupId>

<artifactId>springRestAndDataJpaWithSpringBoot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springRestAndDataJpaWithSpringBoot</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.4.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<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>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Que incluye la dependencia:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

Y las de swagger:

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

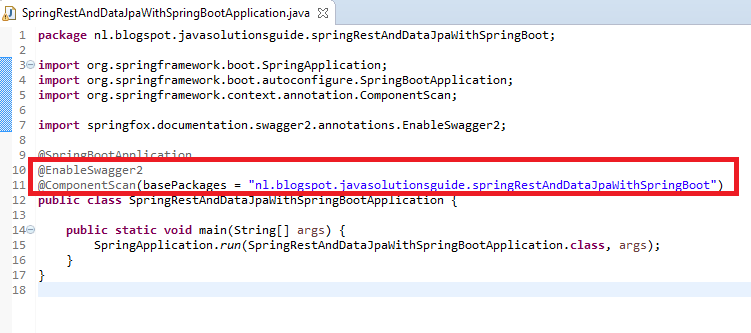
<artifactId>springfox-swagger-ui</artifactId>

<version>2.7.0</version>

</dependency>

# Clase SpringRestAndDataJpaWithSpringBootApplication

En la clase principal de proyecto añadimos 2 dependencias:



@EnableSwagger2

@ComponentScan(basePackages = "nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot")

# ENDPOINTS

Vamos a tener los siguientes rest endpoints:

**– Create Employee resource**

**– Retrieve List of Employees**

**– Retrieve Employee**

**– Update Employee resource**

**– Delete Employee resource**

Creamos todas las clases del proyecto:

# Clase Employee

**package** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="EMPLOYEE")

**public** **class** Employee {

@Id

@GeneratedValue(strategy= GenerationType.***IDENTITY***)

**private** Long id;

@Column(name="EMPLOYEE\_NAME")

**private** String name;

@Column(name="EMPLOYEE\_SALARY")

**private** Integer salary;

@Column(name="DEPARTMENT")

**private** String department;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getSalary() {

**return** salary;

}

**public** **void** setSalary(Integer salary) {

**this**.salary = salary;

}

**public** String getDepartment() {

**return** department;

}

**public** **void** setDepartment(String department) {

**this**.department = department;

}

}

# Clase EmployeeRestController

**package** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RestController;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.entity.Employee;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.service.EmployeeService;

/\*\*

\* **@author** JavaSolutionsGuide

\*

\*/

@RestController

**public** **class** EmployeeRestController {

@Autowired

**private** EmployeeService employeeService;

**public** **void** setEmployeeService(EmployeeService employeeService) {

**this**.employeeService = employeeService;

}

@GetMapping("/api/employees")

**public** List<Employee> getEmployees() {

List<Employee> employees = employeeService.retrieveEmployees();

**return** employees;

}

@GetMapping("/api/employees/{employeeId}")

**public** Employee getEmployee(@PathVariable(name = "employeeId") Long employeeId) {

**return** employeeService.getEmployee(employeeId);

}

@PostMapping("/api/employees")

**public** **void** saveEmployee(Employee employee) {

employeeService.saveEmployee(employee);

System.***out***.println("Employee Saved Successfully");

}

@DeleteMapping("/api/employees/{employeeId}")

**public** **void** deleteEmployee(@PathVariable(name = "employeeId") Long employeeId) {

employeeService.deleteEmployee(employeeId);

System.***out***.println("Employee Deleted Successfully");

}

@PutMapping("/api/employees/{employeeId}")

**public** **void** updateEmployee(@RequestBody Employee employee, @PathVariable(name = "employeeId") Long employeeId) {

Employee emp = employeeService.getEmployee(employeeId);

**if** (emp != **null**) {

employeeService.updateEmployee(employee);

}

}

}

# Clase EmployeeService

**package** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.service;

**import** java.util.List;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.entity.Employee;

/\*\*

\* **@author** JavaSolutionsGuide

\*

\*/

**public** **interface** EmployeeService {

**public** List<Employee> retrieveEmployees();

**public** Employee getEmployee(Long employeeId);

**public** **void** saveEmployee(Employee employee);

**public** **void** deleteEmployee(Long employeeId);

**public** **void** updateEmployee(Employee employee);

}

# Clase EmployeeServiceImpl

**package** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.service.impl;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.entity.Employee;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.repository.EmployeeRepository;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.service.EmployeeService;

/\*\*

\* **@author** JavaSolutionsGuide

\*

\*/

@Service

**public** **class** EmployeeServiceImpl **implements** EmployeeService{

@Autowired

**private** EmployeeRepository employeeRepository;

**public** **void** setEmployeeRepository(EmployeeRepository employeeRepository) {

**this**.employeeRepository = employeeRepository;

}

**public** List<Employee> retrieveEmployees() {

List<Employee> employees = employeeRepository.findAll();

**return** employees;

}

**public** Employee getEmployee(Long employeeId) {

Optional<Employee> optEmp = employeeRepository.findById(employeeId);

**return** optEmp.get();

}

**public** **void** saveEmployee(Employee employee){

employeeRepository.save(employee);

}

**public** **void** deleteEmployee(Long employeeId){

employeeRepository.deleteById(employeeId);

}

**public** **void** updateEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

# Clase EmployeeRepository

**package** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.repository;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** nl.blogspot.javasolutionsguide.springRestAndDataJpaWithSpringBoot.entity.Employee;

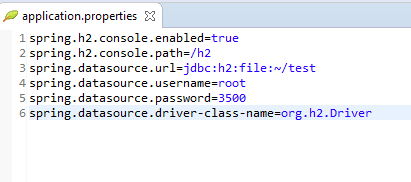
@Repository

**public** **interface** EmployeeRepository **extends** JpaRepository<Employee,Long>{

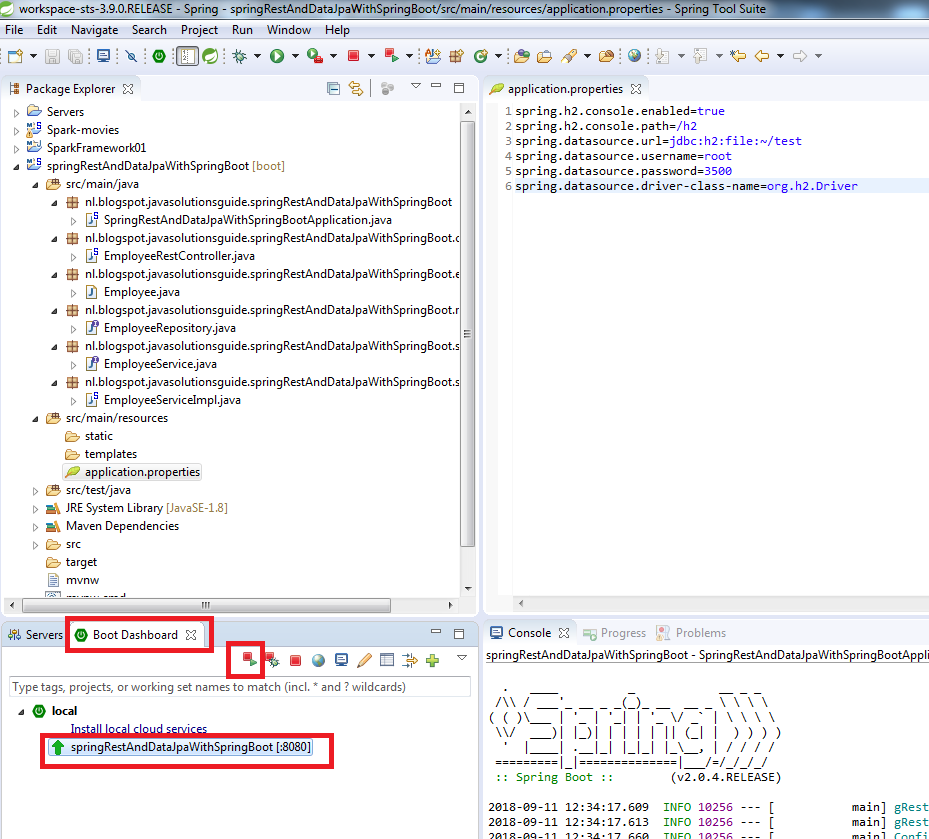
}

# application.properties

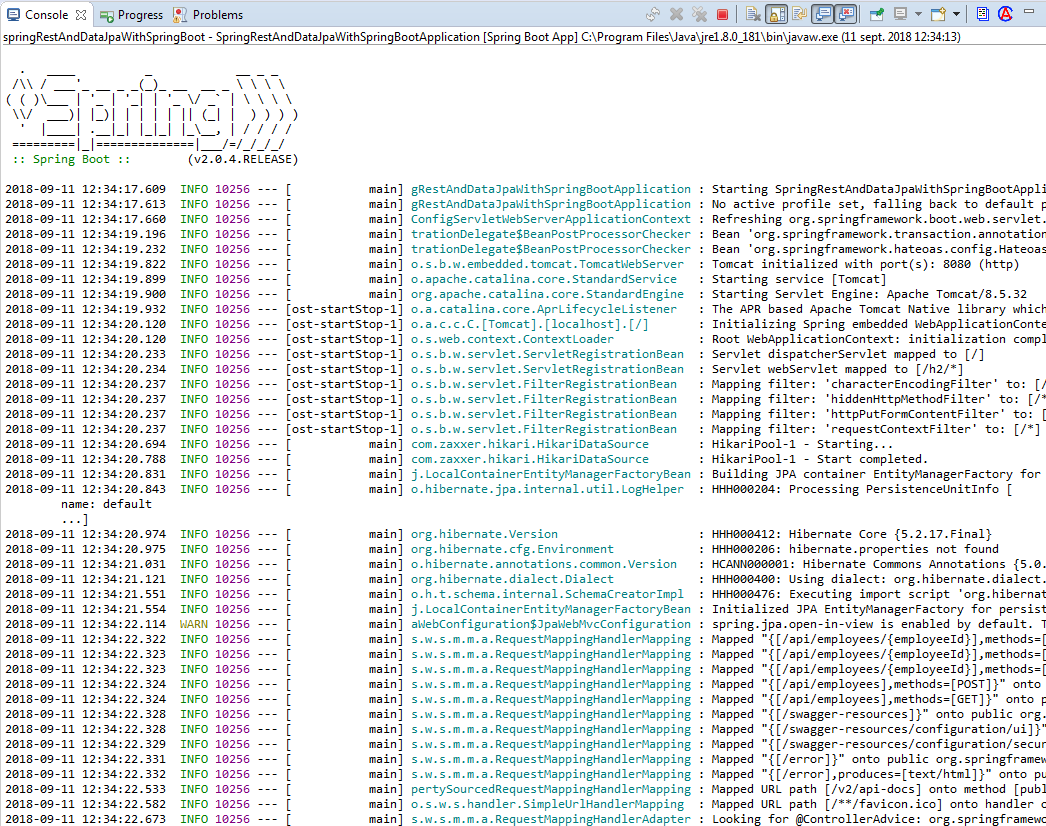
En el fichero **application.properties** damos de alta los datos de la BD H2.



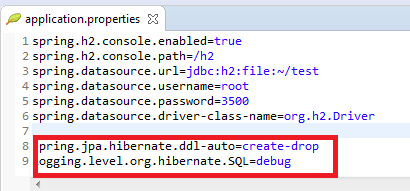
Arrancamos en proyecto Spring-boot.

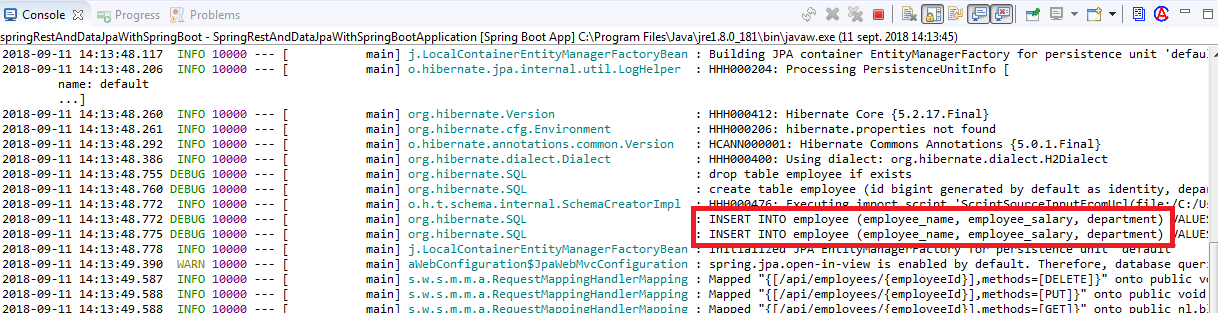


Vemos cómo levanta:



Si añadimos las líneas siguientes al **application.properties**, cuando arrancamos nos muestra como hace los insert’s del **import.sql**

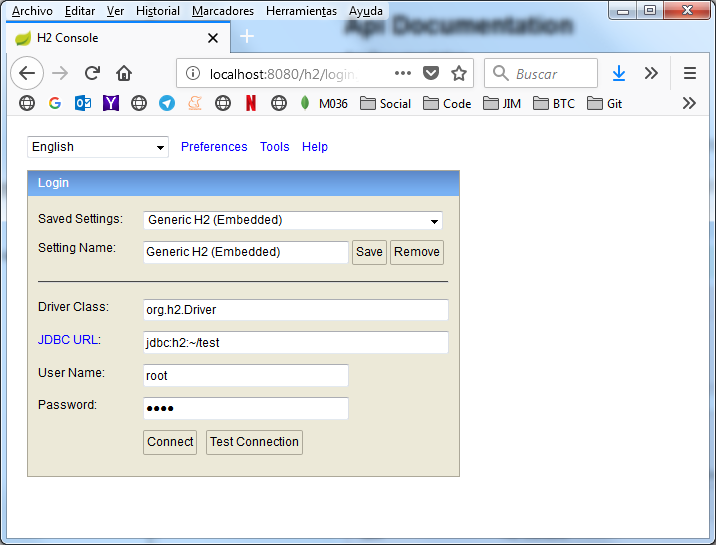




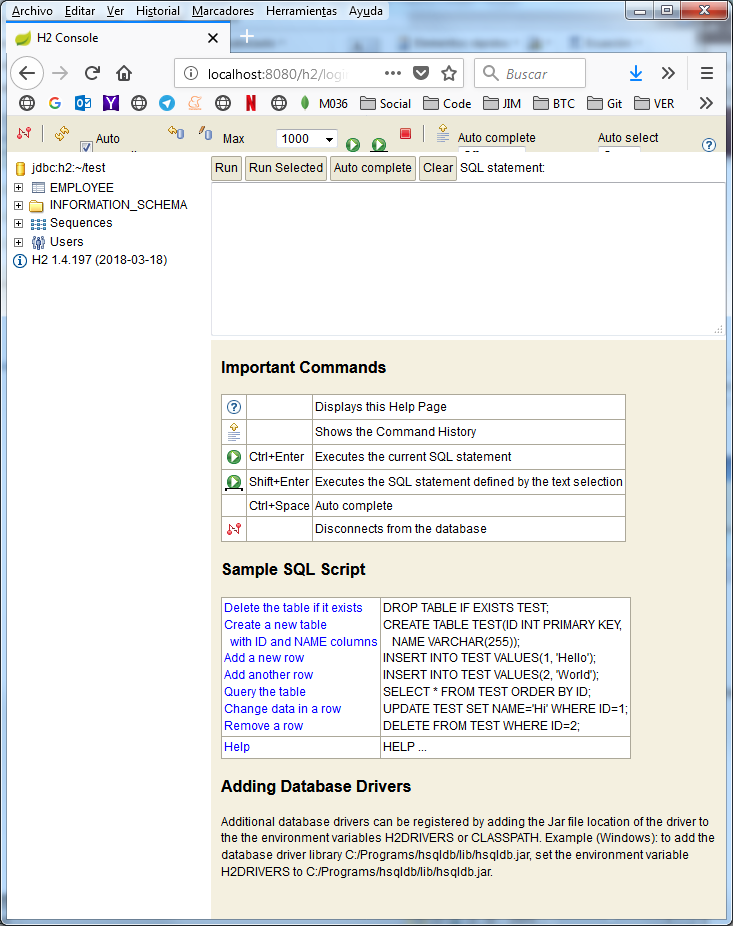
Y accedemos a la consola de la Base de Datos H2 mediante la url:

<http://localhost:8080/h2/>

metiendo el usuario y password, definido en **application.properties**



Y damos a Conectar (Connect):

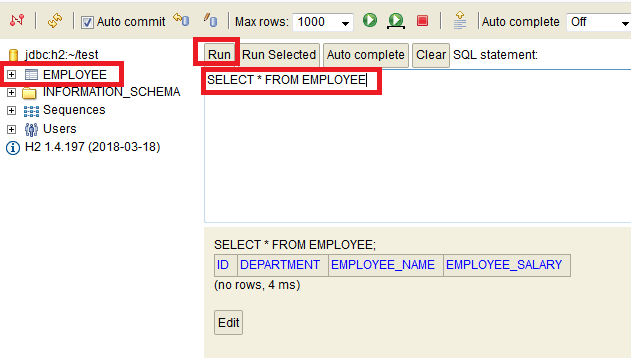


Pulsamos sobre la tabla EMPLOYEE y

Nos aparece la sentencia:

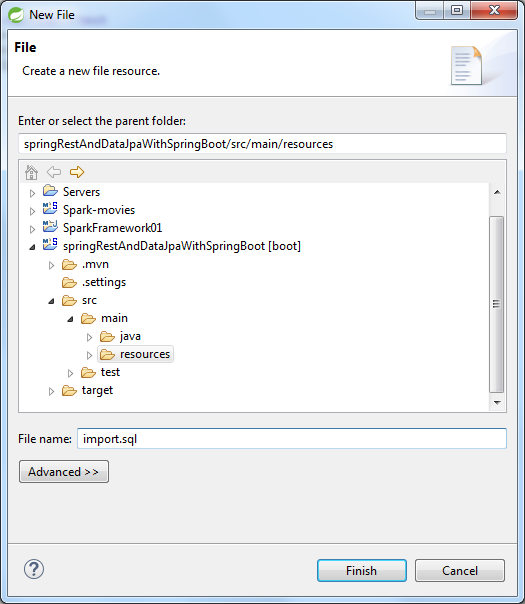
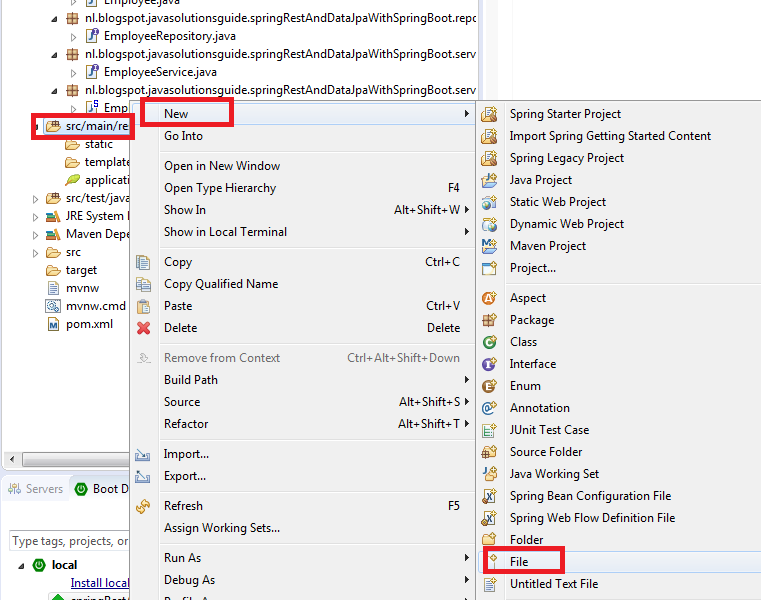
SELECT \* FROM EMPLOYEE

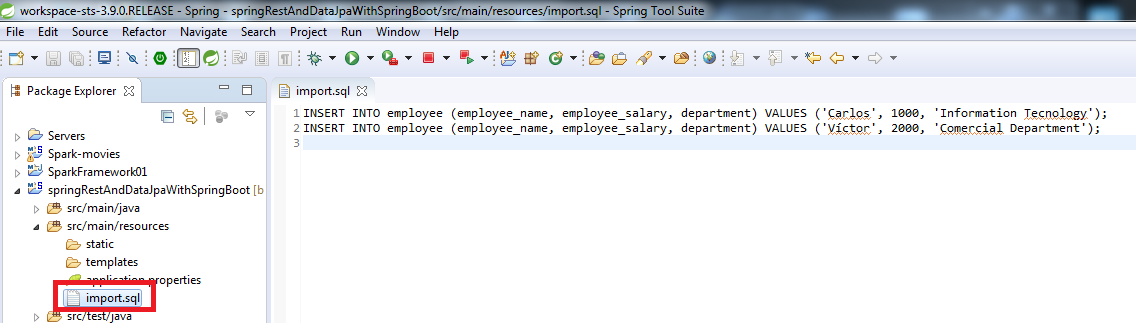
Y si pulsamos en Run, vemos que de momento no existen registros.



Cada vez que paremos y arranquemos la aplicación se pierden los registros.

Se podría crear un fichero import.sql con los insert’s iniciales deseados de la tabla y se cargarían al arrancar la aplicación spring-boot.





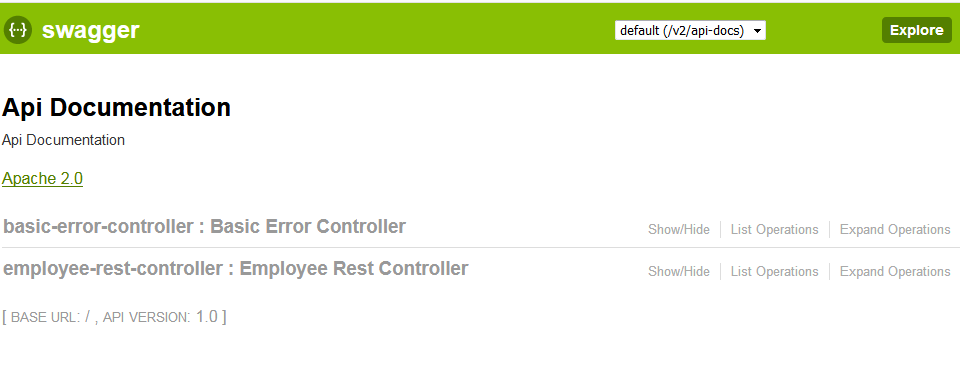
INSERT INTO employee (employee\_name, employee\_salary, department) VALUES ('Carlos', 1000, 'Information Tecnology');

INSERT INTO employee (employee\_name, employee\_salary, department) VALUES ('Víctor', 2000, 'Comercial Department');

# Using Swagger to test the Restful endpoints

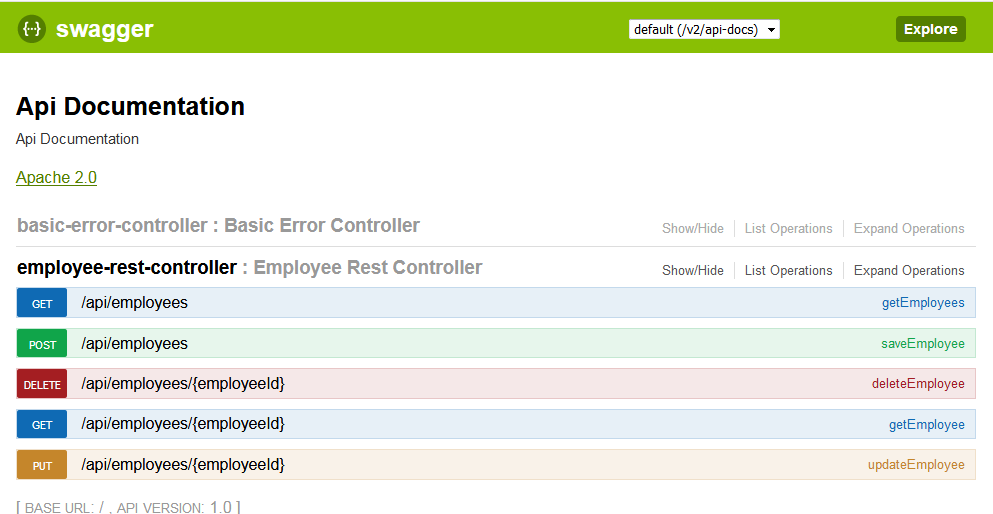
Abrimos la URL:

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



Haga clic en el enlace **employee-rest-controller**.

Se mostrarán las operaciones compatibles con este controlador de la siguiente manera:



Vamos a probar los 5 endpoints uno a uno:

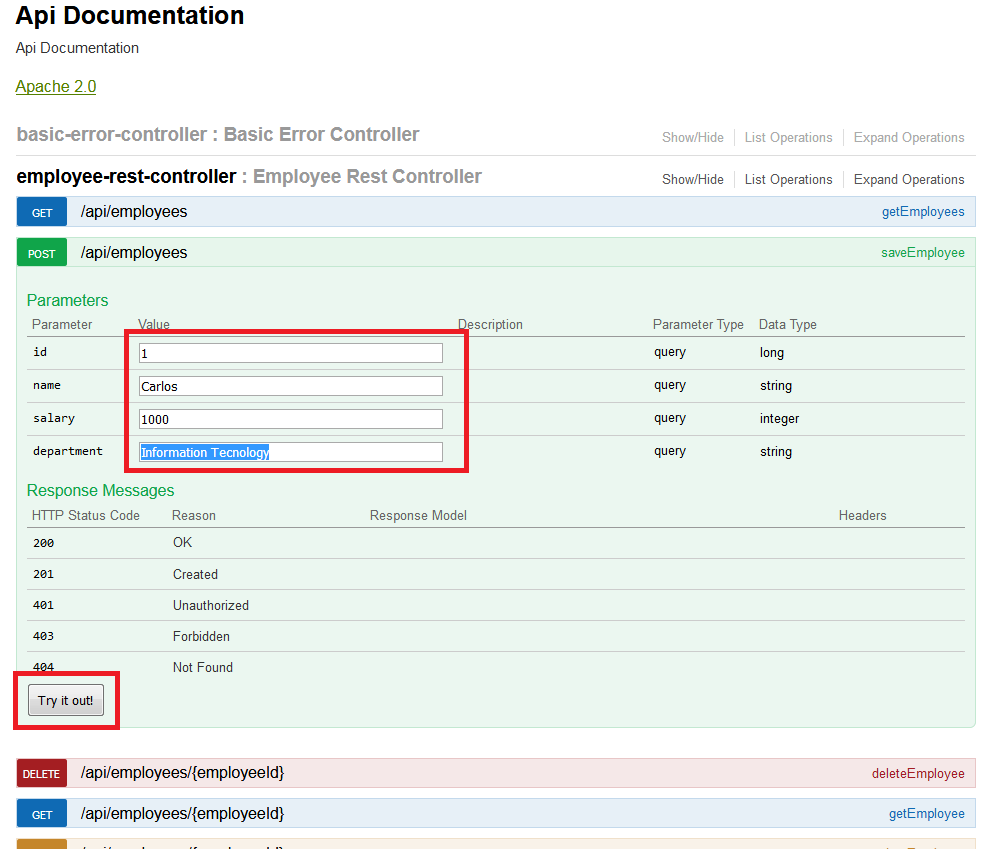
**Save Employee - / api / employees**

Lo primero que debemos hacer es crear un recurso en la base de datos.

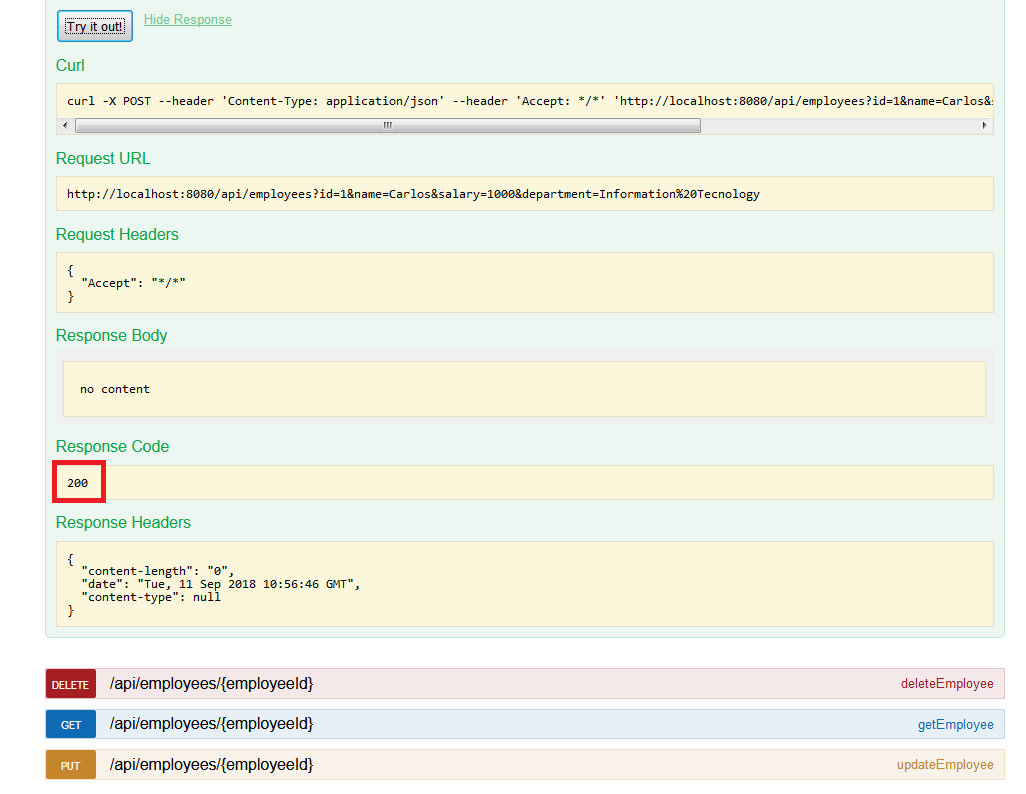
Para eso utilizaremos la operación POST y usaremos / api / employees endpoint.

Restful API

Haga clic en saveEmployee y complete todos los datos requeridos que necesitamos para crear un recurso y luego haga clic en el botón "probarlo".

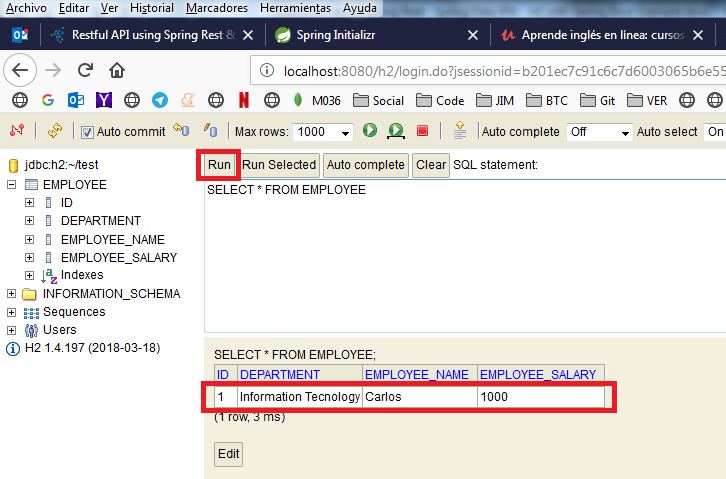


Así es como se verá tu solicitud y respuesta



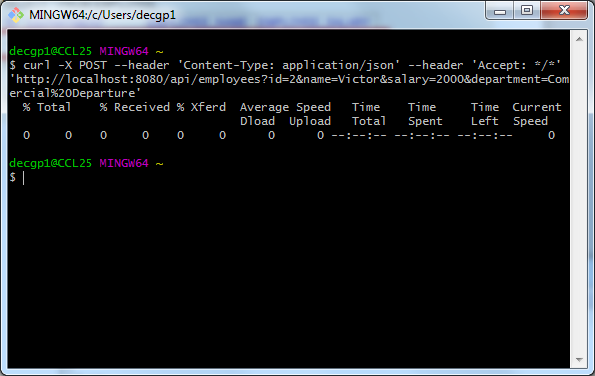
Como puede ver, el código de respuesta es 200, lo que significa ÉXITO y, por lo tanto, nuestro registro debería haberse creado en la base de datos H2.

Vamos a verificar eso.

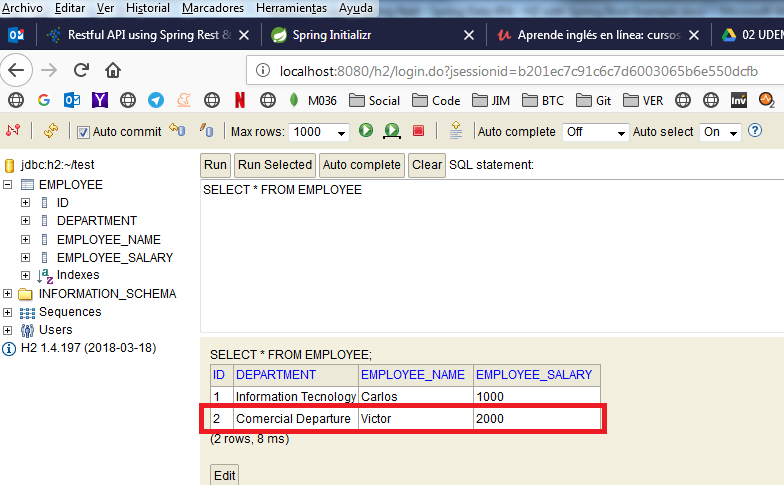


También puedo lanzar desde Git Bash un curl:

*curl -X POST --header 'Content-Type: application/json' --header 'Accept: \*/\*' 'http://localhost:8080/api/employees?id=2&name=Victor&salary=2000&department=Comercial%20Departure'*

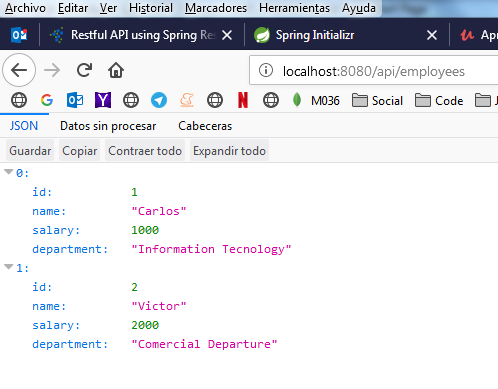


Y me insertará otro registro:

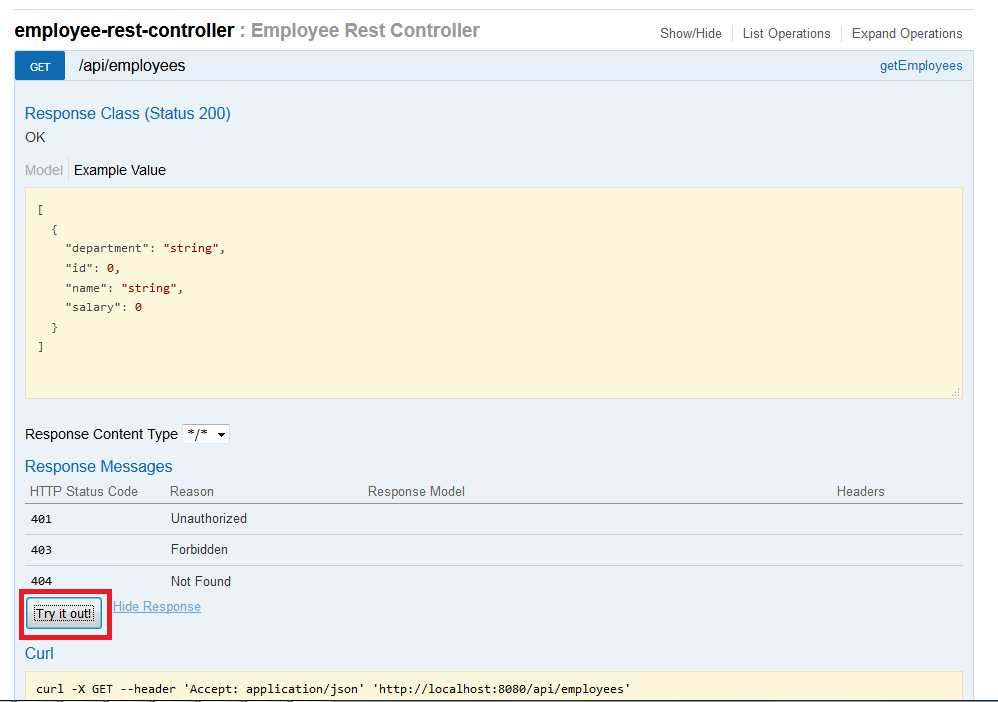


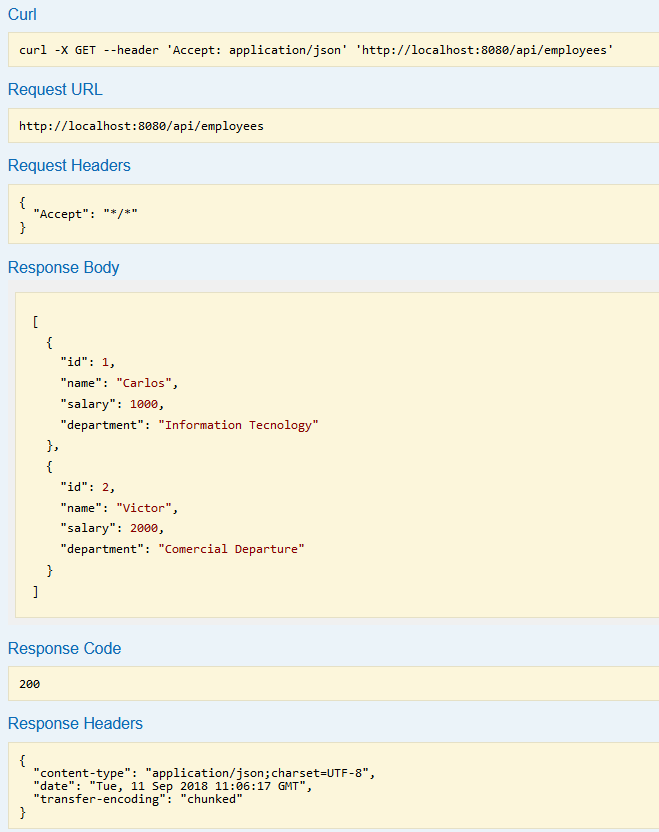
También puedo obtener los employees, por ejemplo con el navegador introduciendo la url:

<http://localhost:8080/api/employees>



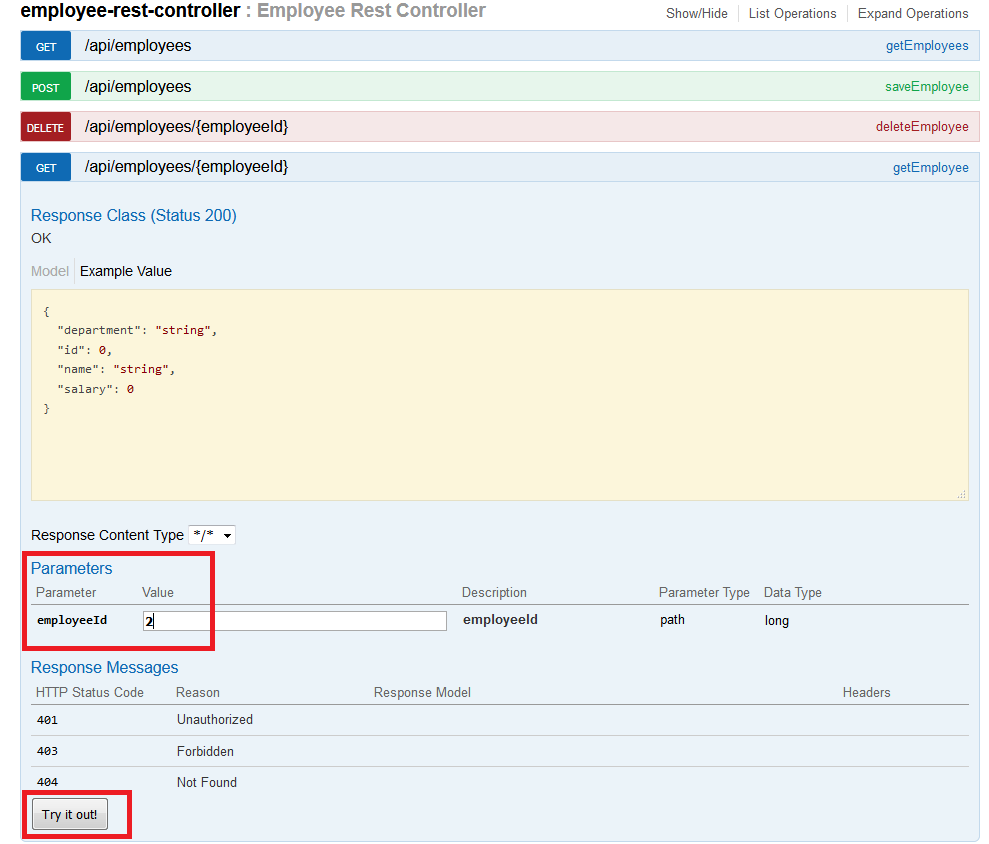
O desde swagger:



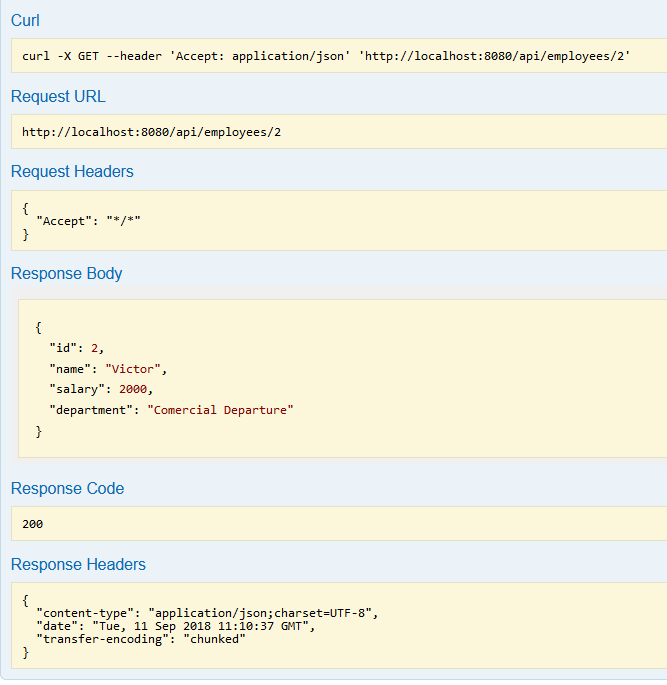


Y obtenemos los datos introducidos.

Para consultar en Empleado 2:



Y nos devuelve:



Actualizamos en empleado 2 pasando su sueldo a 3000:

Rellenamos con el json:

*{*

*"id":2,*

*"name":"Victor",*

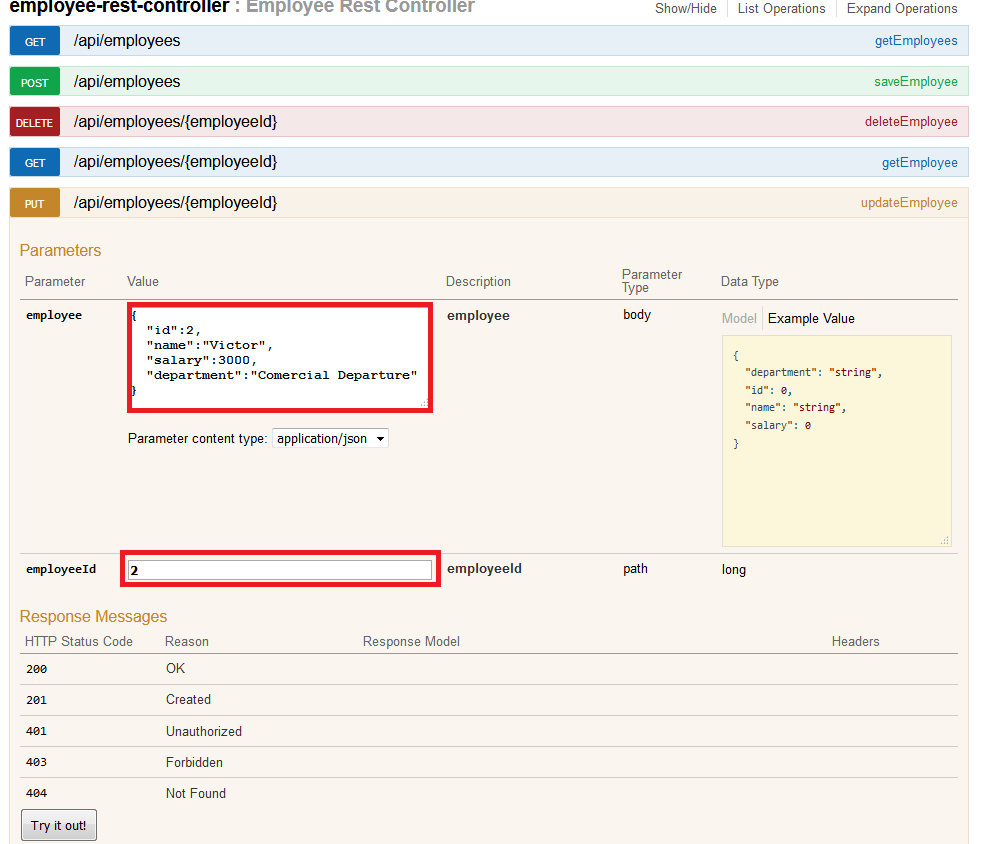
*"salary":3000,*

*"department":"Comercial Departure"*

*}*

E indicamos en emploeeId:

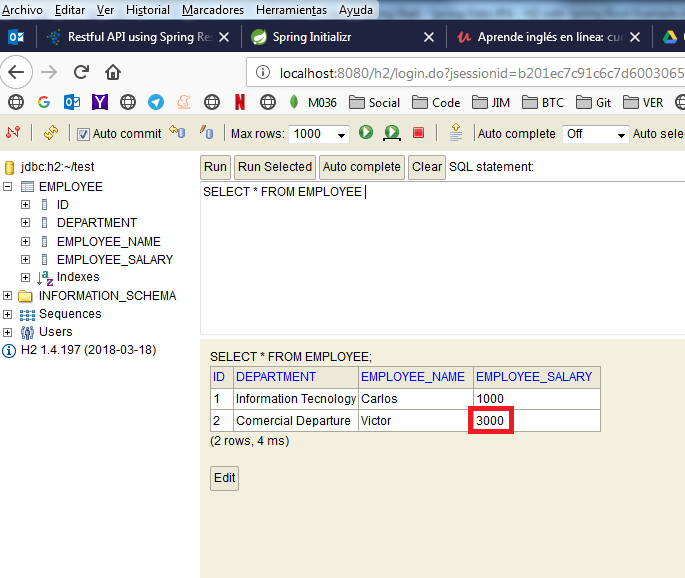
Que es 2.



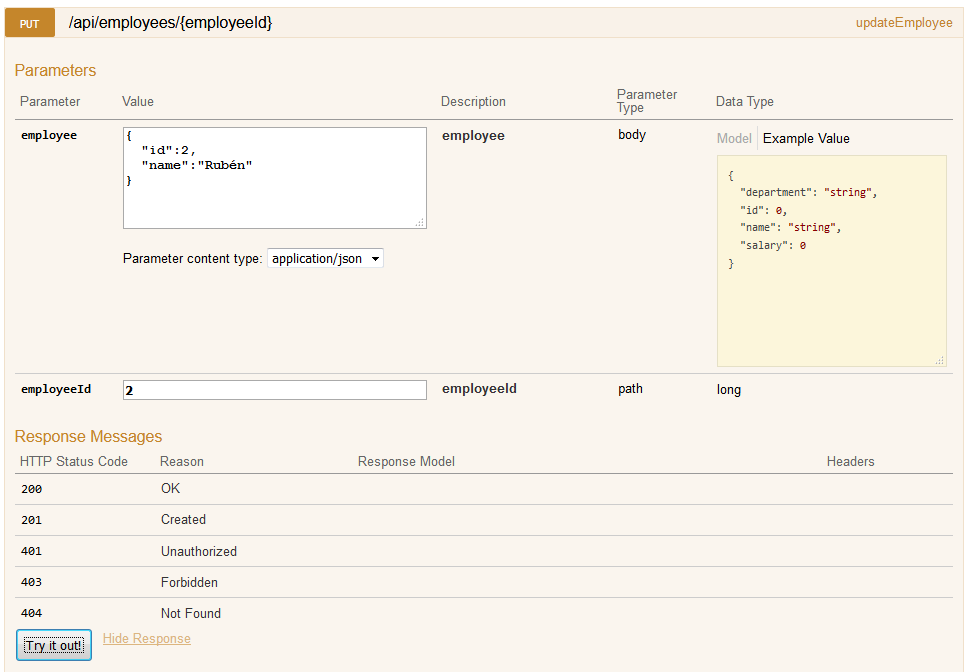
Y obtenemos:



Consultamos la BD H2, y vemos que se ha actualizado a 3000



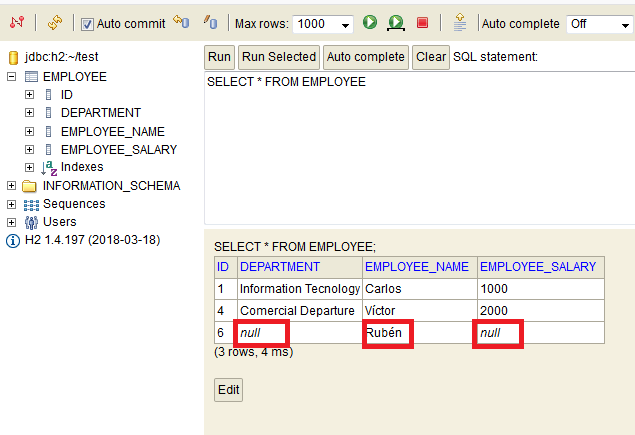
Para actualizar hay que indicar todos los datos del registro, si solo mandamos un valor insertará como si fuese un registro nuevo:



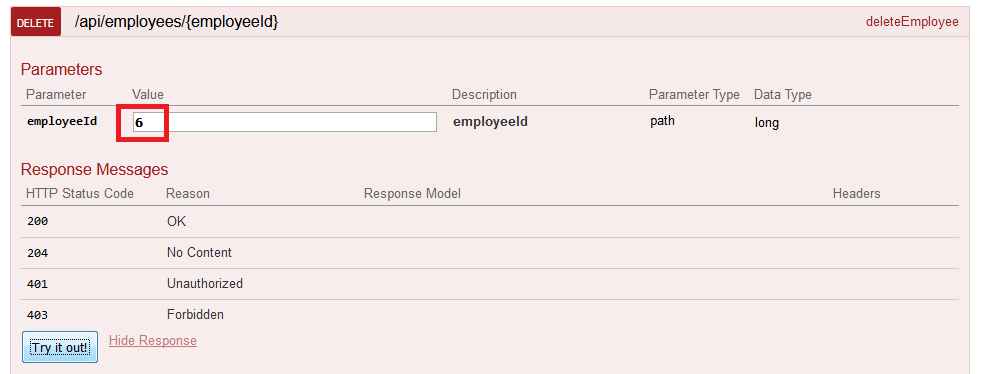
Nos da:



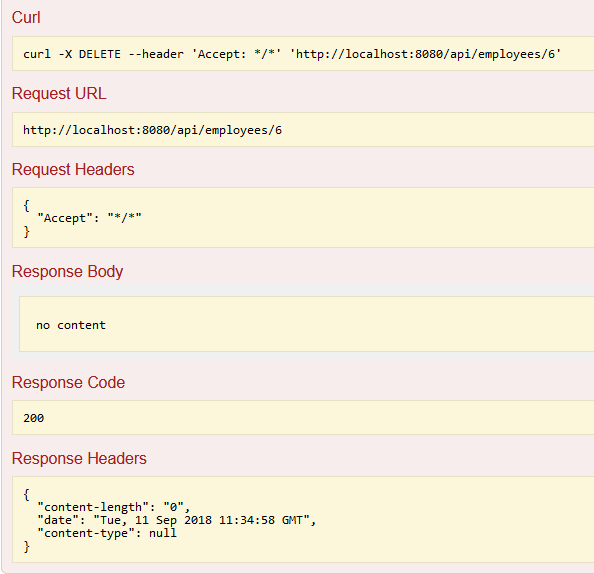
Pero lo que se inserta en H2 es:



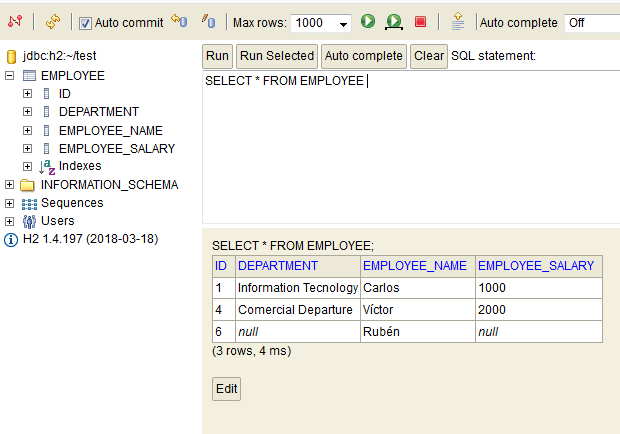
Y para borrar:



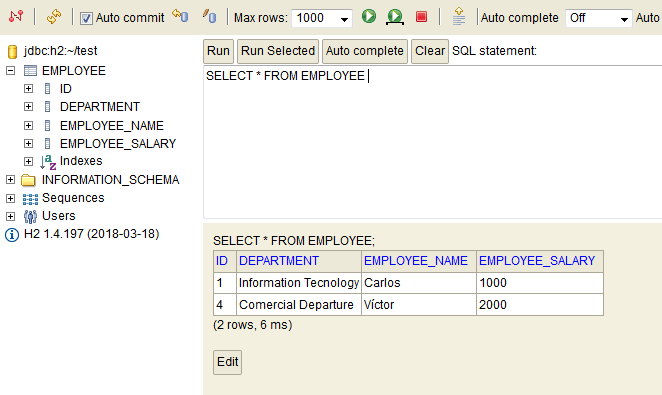
Nos da:



Y en BD H2, se elimina el registro con id=6, pasamos de:



a:



# Proyecto final

