

PASOS PARA HACER FUNCIONAR EL SISTEMA:

1. DESCARGAR EL PROYECTO
2. UNA VEZ DESCARGADO EL PROYECTO, SE DEBE EJECUTAR EL SIGUIENTE SCRIP EN EL WORKBENCH DE MYSQL:

```
CREATE DATABASE db_proyectokodigo;
```

```
USE db_proyectokodigo;
```

3. VERIFICAR QUE EN INTELLIJIDEA EL ARCHIVO: application.yml se vea así: “create”. Además en el caso del PORT se usa el 3306, USER se designó que fuese: “root”, y la PASSWORD es vacía: “”. Por lo tanto, las variables de entorno ya están programadas para manejar esos datos.

```
spring:
  application:
    name: proyecto_Inventario
  jackson:
    default-property-inclusion: non_null
    property-naming-strategy: SNAKE_CASE
  datasource:
    url: ${DB_HOST}
    username: ${DB_USER}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: create
      show-sql: true

logging:
  level:
    org.springframework: DEBUG
    com.proyectoInventario_Kodigo.proyecto_Inventario: DEBUG
```

4. Ejecutar spring y esperar ya que por el DEBUG aparecerán muchos logs, hasta lograr esto:

```
: Started ProyectoInventarioApplication in 11.299 seconds (process running for 12.009)
: Application availability state LivenessState changed to CORRECT
: Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
```

5. Luego colocar en el application.yml la siguiente palabra: “update”:

```
jpa:
  hibernate:
    ddl-auto: update
  show-sql: true
```

6. Luego es necesario que en la base de datos se vea así:



7. Luego es necesario hacer estos insert:

```
INSERT INTO roles (name) VALUES ("ROLE_ADMIN");
```

```
INSERT INTO roles (name) VALUES ("ROLE_USER");
```

```
SELECT * FROM roles;
```

8. Luego es necesario ir a postman y colocar la siguiente URL:

POST: <http://localhost:8080/users>

Crea cualquier usuario:

```
{  
  "username": "admin",  
  "email": "admin@gmail.com",  
  "password": "12345",  
  "admin": true  
}
```

9. Luego hace login con la siguiente URL:

POST: <http://localhost:8080/login>

```
{  
  "username": "admin",  
  "password": "12345"  
}
```

10. Cada vez que haga “login” necesita ver lo siguiente:



The screenshot shows a REST client interface with the following details:

- Body** tab is selected.
- Headers (15)** and **Test Results** tabs are visible.
- Status:** 200 OK
- Time:** 690 ms
- Size:** 1.17 KB
- Save as example** button is present.
- JSON** format is selected for the response.
- The response body is a JSON object with the following structure:

```
1 {
2   "message": "Hello admin you have logged in successfullly!!",
3   "token": "eyJhbGciOiJIUzUxMiJ9.eyJhdXRob3JpdGllcyI6Ilt7XCJhdXRob3JpdHlcIjpcIlJPTeVfQURNSU5cIn0se1wiYXV0aG9yaXR5XCi6XCJST0xFOX1VTRVJcInIdIiwiaXNBZG1pbiI6dHJ1ZSwidXNlc
      m5hbWU1OiJhZG1pbiIsInN1YiI6ImFkbWluIiwiaWF0IjoxNzE5NDM5Njc3LCJleHAiOjE3MTk0NDMyNzd9.
      MZk8hqXPTgLT4mNWoFowj1PVT4eUD0MoHoXxbF-I-B3rbfk4oLb-1Awg16BqT_VUdELQR0DIhya05QHUqBo7dQ",
4   "username": "admin"
5 }
```

Ese token debe de usarlo en cada endpoint que desee realizar. Para eso se explica en el video y en la documentación. **Video en minuto 19:50** se empiezan a utilizar tanto en SWAGGER como en POSTMAN.