**SO Windows**

**🤔 1 Comando de ejecución del servidor MySQL como contenedor docker**

docker run -d `

--name hibernate-db `

-e MYSQL\_ROOT\_PASSWORD=root `

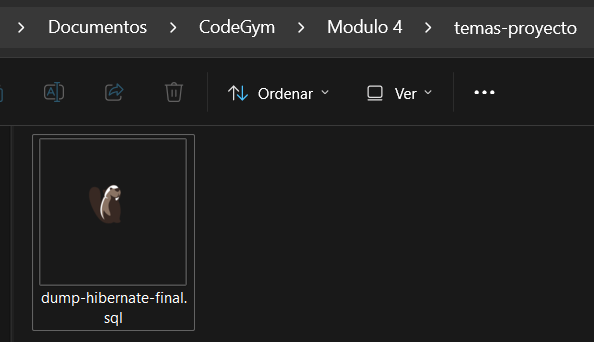
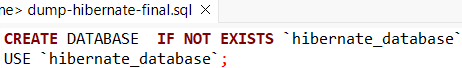
-e MYSQL\_DATABASE=hibernate\_database `

-p 3310:3306 `

mysql:8.0 `

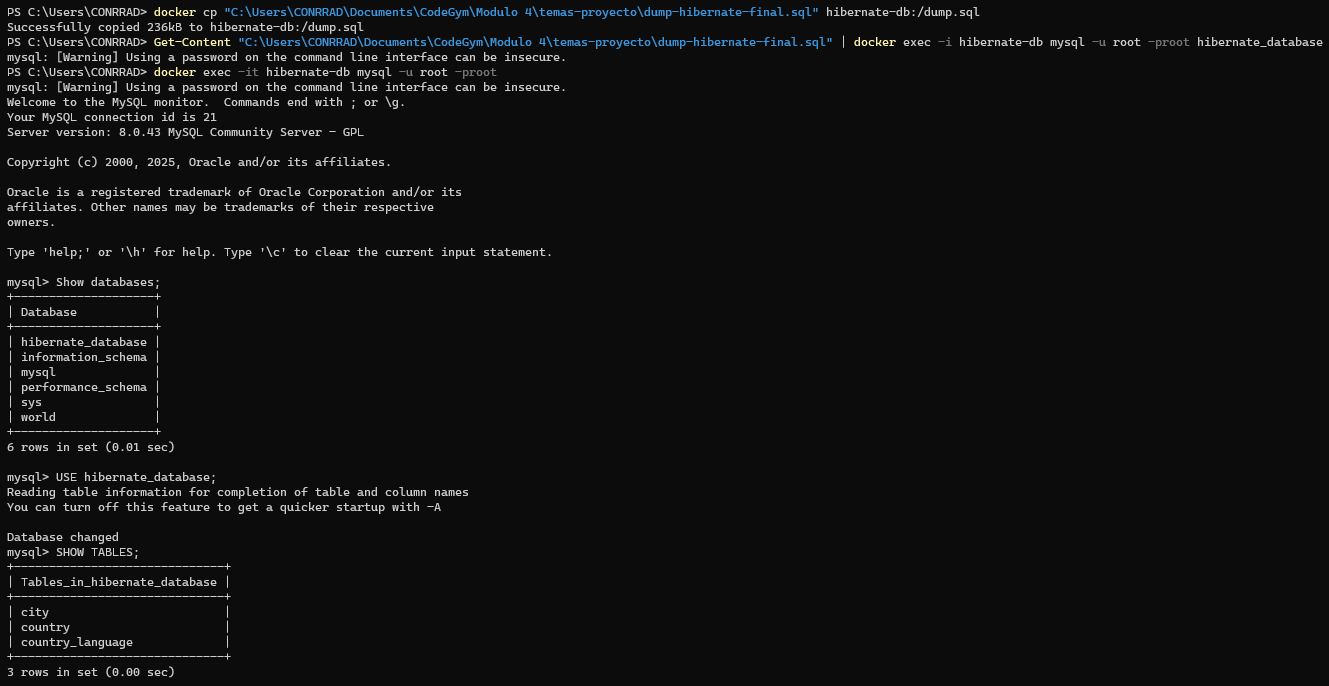
--default-authentication-plugin=mysql\_native\_password

**👌 2 Ubicación y modificación de la BD - dump-hibernate-final.sql**

**** 



**😺 3 Importación del archivo dump.sql en la base de datos**



1. docker cp "C:\Users\CONRRAD\Documents\CodeGym\Modulo 4\temas-proyecto\dump-hibernate-final.sql" hibernate-db:/dump.sql  
   Successfully copied 236kB to hibernate-db:/dump.sql
2. Get-Content "C:\Users\CONRRAD\Documents\CodeGym\Modulo 4\temas-proyecto\dump-hibernate-final.sql" | docker exec -i hibernate-db mysql -u root -proot hibernate\_database
3. docker exec -it hibernate-db mysql -u root –proot
4. mysql> Show databases;

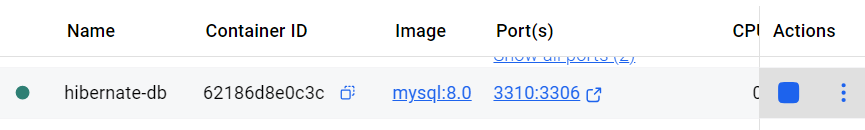
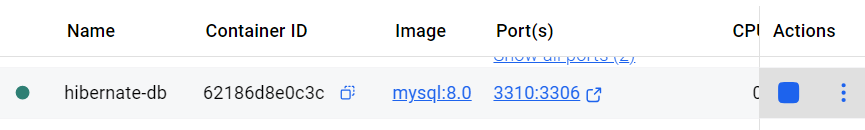
|  |
| --- |
| Database |
| hibernate\_database |
| information\_schema |
| mysql |
| performance\_schema |
| sys |
| world |

1. mysql> USE hibernate\_database;
2. mysql> SHOW TABLES;

|  |
| --- |
| Tables\_in\_hibernate\_database |
| city |
| country |
| country\_language |

**😺 4 DBaver**

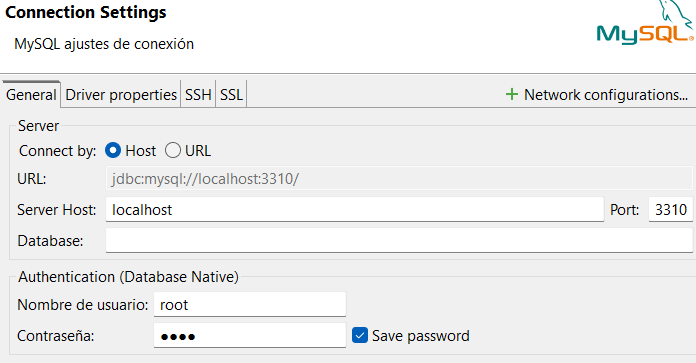
Docker container MySQL

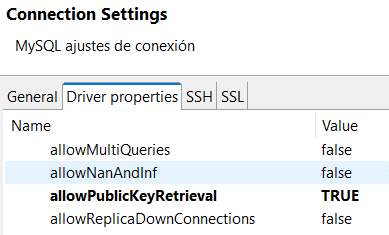


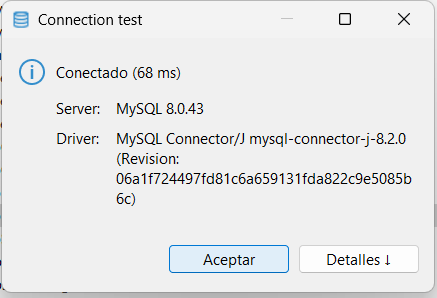
Selecciona la BD

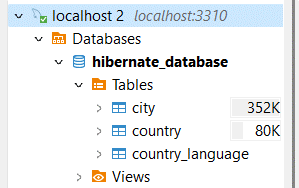


Puerto: 3310 Contraseña: root









**😺 4 Conexión a la BD MySQL - Dentro del main**

private SessionFactory prepareRelationalDb() {  
 final SessionFactory sessionFactory;  
 Properties properties = new Properties();  
 properties.put(Environment.*DIALECT*, "org.hibernate.dialect.MySQL8Dialect");  
 properties.put(Environment.*DRIVER*, "com.p6spy.engine.spy.P6SpyDriver");  
 properties.put(Environment.*URL*, "jdbc:p6spy:mysql://localhost:3310/hibernate\_database");  
 properties.put(Environment.*USER*, "root");  
 properties.put(Environment.*PASS*, "root");  
 properties.put(Environment.*CURRENT\_SESSION\_CONTEXT\_CLASS*, "thread");  
 properties.put(Environment.*HBM2DDL\_AUTO*, "validate");  
 properties.put(Environment.*STATEMENT\_BATCH\_SIZE*, "100");

**🤔 1 Comando de ejecución de Redis como contenedor docker**

|  |  |  |
| --- | --- | --- |
| Argumento | Descripción | Propósito |
| docker run | Inicia un nuevo contenedor. | El comando base de Docker. |
| -d | *Detached mode*. | Ejecuta el contenedor en **segundo plano**, liberando tu terminal. |
| --name redis-stack | Asigna un nombre. | Le da un nombre fácil de recordar al contenedor (redis-stack). |
| -p 6379:6379 | Mapeo de Puertos. | Mapea el puerto predeterminado de Redis (6379) del contenedor al puerto 6379 de tu PC. |
| -p 8001:8001 | Mapeo de Puertos. | Mapea el puerto de la Interfaz de Usuario Gráfica (RedisInsight) del contenedor al puerto 8001 de tu PC. |
| redis/redis-stack:latest | Imagen a Usar. | La imagen específica que contiene el servidor Redis y la interfaz RedisInsight. |

docker run -d `

--name redis-stack `

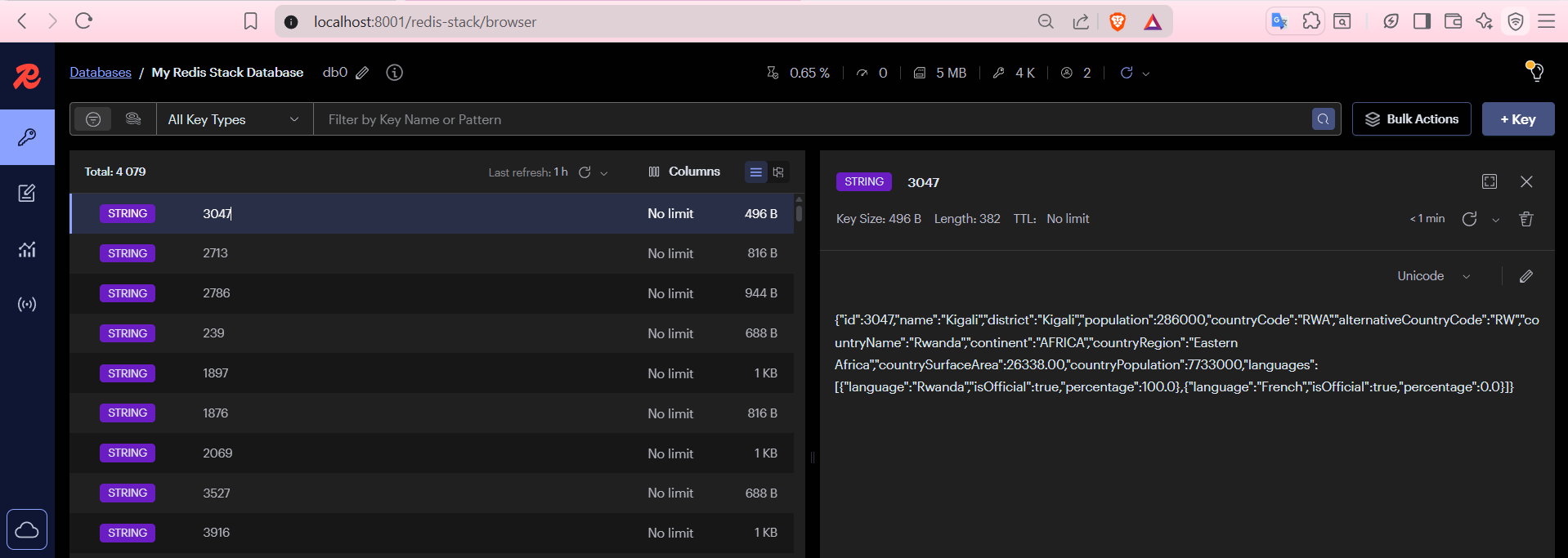
-p 6379:6379 `

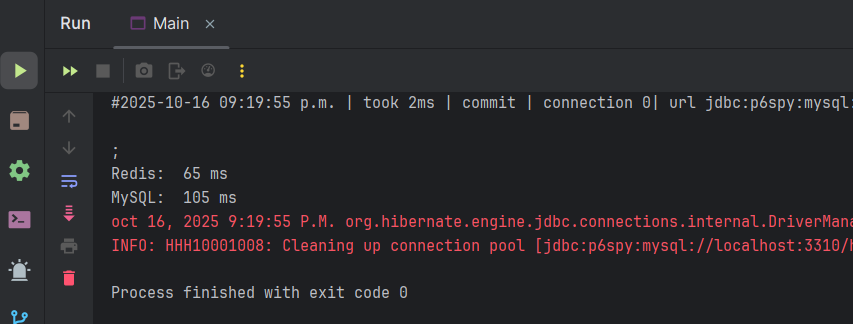
-p 8001:8001 `

redis/redis-stack:latest

**👌 Configuracion del cliente Redis - Dentro del main**

private RedisClient prepareRedisClient() {  
 RedisClient redisClient =  
 RedisClient.*create*(RedisURI.*create*("localhost", 6379));  
 try (StatefulRedisConnection<String, String> connection = redisClient.connect()) {  
 System.*out*.println("\nConnected to Redis");  
 }  
 return redisClient;  
}





**Resultado:  
Tiempo de diferencia**