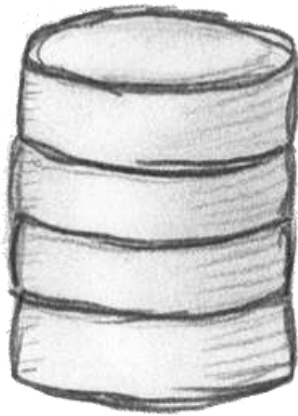


# MyISAM vs InnoDB



Se va a tratar de explicar las características mas importantes que hay a la hora de trabajar con bases de datos entre los motores **Myisam vs Innodb**, qué **ventajas** tiene Myisam frente a Innodb o **desventajas**, así como las diferencias existentes, muy enfocado al **desarrollo web**.

A la hora de abarcar un **proyecto web** que sobre todo va a tener diversas ejecuciones o comunicaciones **con bases de datos**, es muy importante **conocer las capacidades del servidor** (alojamiento) que va a gestionar nuestro desarrollo, no solo en capacidad de almacenamiento sino también en cuanto a versión de software y base de datos que van a manejarlo.

Los motores más populares y usados en desarrollo web son **MyISAM e InnoDB**, su correcta elección definirá como se gestionarán los recursos en cuanto a **velocidad**, consumo de esos **recursos y calidad** de servicio.

Cada proyecto tiene su casuística, a la que debemos prestar atención, conociendo el **número de usuarios que accederán** o pueden acceder **simultáneamente** a realizar altas, bajas, etc., o bien si tenemos miles de accesos solamente a consulta.

En el presente artículo no vamos a entrar al detalle de que diseño de base de datos o sistema de desarrollo es más idoneo a la hora de manipular la información, evidentemente precisa de un capítulo a parte, pero ni que decir tiene que es un punto muy importante, el cual se ha de tratar con detenimiento.

## Características de MyISAM:

- Se establece **por defecto cuando se crea una tabla**, salvo que se indique lo contrario.
- Soporta transacciones.
- Realizar bloqueo de registros.
- **Soporta un gran número de consultas SQL**, lo que se refleja en una **velocidad de carga muy rápida** para nuestra web.

Como desventaja, señalamos que **no realiza bloqueo de tablas**, esto puede ser un problema si como se ha mencionado anteriormente hay un acceso simultáneo al mantenimiento de registros por parte de varios usuarios.

## Características de InnoDB

- **Bloqueo de registros**. Importante para accesos múltiples al mantenimiento de tablas, es decir, **ejecuciones de sentencias tipo INSERT o UPDATE**, éstas ejecuciones tienen una velocidad optimizada.
- Capacidad para **soportar transacciones e integridad de datos**, es decir previene el alta de datos no adecuados.
- Aplica las características propias de **ACID** (Atomicity, Consistency, Isolation and Durability), consistentes en garantizar la integridad de las tablas.

Como desventaja, marcamos que al ser un tipo de motor que define un sistema más complejo de diseño de tablas, **reduce el rendimiento en velocidad** para desarrollo que requieren de un **elevado número de consultas**.



## RECOMENDACIONES

Un solo gestor de mantenimiento para una plataforma que requerirá muchas consultas o visitas: **MyISAM**

Necesitas velocidad y mínimo consumo de recursos en servidor, espacio, RAM, etc.: **MyISAM**

Varios o muchos gestores de mantenimiento: **InnoDB**

Desarrollo donde se prioriza el diseño relacional de bases de datos: **InnoDB**

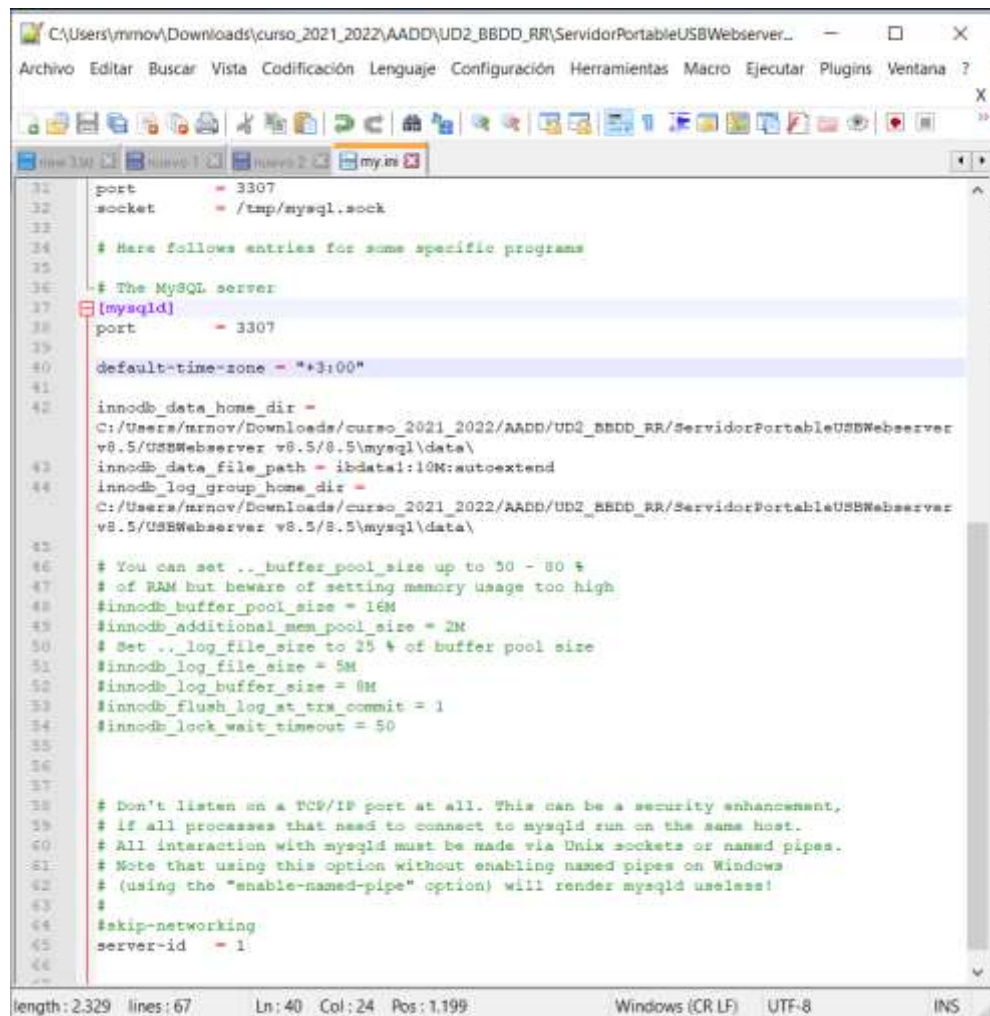
Independientemente del sistema que se elija, hay que hacer un buen diseño de la **estructura y funcionalidad** de la base de datos, si bien este tema requiere un capítulo aparte, os damos unas nociones a tener en cuenta.

La información o datos no deben almacenarse de cualquier manera. Hay que buscar el mayor aprovechamiento de los recursos que tenemos a nuestra disposición, tanto a nivel de **almacenamiento** como de **rendimiento**.

Hay que mantener la consistencia de la información durante todo el ciclo de vida de la base de datos, más aún si los datos que se manejan son críticos, por ejemplo los salarios de una organización.

Los primeros factores que realizará el analista, serán el análisis del sistema que servirá de modelo, la observación de los elementos que lo componen y la descomposición en partes mucho más pequeñas.

# HORA



```
32 port = 3307
33 socket = /tmp/mysql.sock
34
35 # Here follows entries for some specific programs
36
37 # The MySQL server
38 [mysqld]
39 port = 3307
40
41 default-time-zone = "+3:00"
42
43 innodb_data_home_dir =
44 C:/Users/mrmov/Downloads/curso_2021_2022/AADD/UD2_BBDD_RR/ServidorPortableUSBWebserver
45 v8.5/USBWebserver v8.5/8.5/mysql\data\
46 innodb_data_file_path = ibdata1:10M:autoextend
47 innodb_log_group_home_dir =
48 C:/Users/mrmov/Downloads/curso_2021_2022/AADD/UD2_BBDD_RR/ServidorPortableUSBWebserver
49 v8.5/USBWebserver v8.5/8.5/mysql\data\
50
51 # You can set .._buffer_pool_size up to 30 - 80 %
52 # of RAM but beware of setting memory usage too high
53 #innodb_buffer_pool_size = 16M
54 #innodb_additional_mem_pool_size = 2M
55 # Set .._log_file_size to 25 % of buffer pool size
56 #innodb_log_file_size = 5M
57 #innodb_log_buffer_size = 8M
58 #innodb_flush_log_at_trx_commit = 1
59 #innodb_lock_wait_timeout = 50
60
61 # Don't listen on a TCP/IP port at all. This can be a security enhancement,
62 # if all processes that need to connect to mysqld run on the same host.
63 # All interaction with mysqld must be made via Unix sockets or named pipes.
64 # Note that using this option without enabling named pipes on Windows
65 # (using the "enable-named-pipe" option) will render mysqld useless!
66 #
67 #skip-networking
68 server-id = 1
```

```
select now();
```

```
SELECT @@session.time_zone;
```

```
SELECT @@global.time_zone;
```

```
SET GLOBAL time_zone = '+3:00';
```

# Procedimientos Almacenados (Rutinas) En PhpMyAdmin

Un procedimiento almacenado es una rutina en phpmyadmin, definida en una consulta,

Para acceder a esta herramienta, en la barra de menú seleccionamos **Rutinas**:



Al abrir el panel de rutinas, se mostrará la siguiente ventana:

A screenshot of the 'Agregar rutina' (Add routine) form in phpMyAdmin. The form has a title bar 'Agregar rutina' with a close button. It contains several sections: 'Detalles' (selected), 'Nombre de rutina' (text input), 'Tipo' (dropdown menu showing 'PROCEDURE'), 'Parámetros' (table with columns: Dirección, Nombre, Tipo, Longitud/Valores, Opciones), 'Definición' (large text area), 'Es determinístico' (checkbox), 'Definidor' (text input), 'Tipo de seguridad' (dropdown menu showing 'DEFINER'), 'Acceso de datos SQL' (dropdown menu showing 'NO SQL'), and 'Comentario' (text input). At the bottom right are 'Continuar' and 'Cerrar' buttons.

	Dirección	Nombre	Tipo	Longitud/Valores	Opciones
1	IN		I		

Su sintaxis es la siguiente:

- **Nombre de rutina:** es el nombre de la rutina o procedimiento almacenado
- **Tipo :** Existen 2 tipos de rutinas; Procedure y Function
- **Parametros:** Son las columnas o nombre de los campos de la tabla
- **Definicion:** En este campo se coloca la consulta a ejecutar.
- **Definidor:** aca se coloca el nombre del administrador y el tipo de servidor.
- **Tipo de Seguridad:** Existen 2 tipos de seguridad; Definir y Invoker

- **Acerca de datos SQL:** se selecciona CONTAINS SQL

## Ejemplo:

Crear una rutina para insertar datos

**Editar rutina**

**Detalles**

Nombre de rutina: Ingresarpedido

Tipo: PROCEDURE

**Parámetros**

Dirección	Nombre	Tipo	Longitud/Valores	Opciones
IN	codigo_pedido	CHAR		Juego de caract. Eliminar
IN	nombre_cliente	VARCHAR	100	Juego de caract. Eliminar
IN	descripcion	VARCHAR	50	Juego de caract. Eliminar
IN	cantidad	INT		Eliminar
IN	mesa	INT		Eliminar

**Definición**

```
insert into pedidos values(codigo_pedido,nombre_cliente,descripcion,cantidad,mesa)
```

Es determinístico: ☒

Ajustar privilegios: ☒

Definidor: root@localhost

Tipo de seguridad: DEFINER

Acceso de datos SQL: CONTAINS SQL

Comentario:

Continuar Cerrar

Una vez llenado los campos, le damos clic en **Continuar**. Para ejecutar el procedure, seleccionamos de nuevo la opción **Rutinas** de la barra de menú:



Se mostrará el panel con las rutinas creadas, le damos clic en **Ejecutar**:

Ejecutar rutina `ingresarpedido`

Parámetros de rutina

Nombre	Tipo	Función	Valor
codigo_pedido	CHAR		
nombre_cliente	VARCHAR		
descripcion	VARCHAR		
cantidad	INT		
mesa	INT		

Continuar
Cerrar

Para ingresar datos se utiliza la columna **Valor** y pulsamos en **Continuar**:

Ejecutar rutina `ingresarpedido`

Parámetros de rutina

Nombre	Tipo	Función	Valor
codigo_pedido	CHAR		G01
nombre_cliente	VARCHAR		Raul benites
descripcion	VARCHAR		Gaseosa Inka kola
cantidad	INT		2
mesa	INT		4

Continuar
Cerrar

# Ejecutar Procedimiento Almacenado Mysql desde Java

Es posible invocar procedimientos almacenados posterior a la conexión a la base de datos, haciendo uso de un objeto **CallableStatement**, pasarle parámetros, ejecutarlo y obtener valores de retorno (para el caso de funciones o si tiene un procedimiento almacenado que tiene valores\*\* in /out\*\* como en **Oracle** o en este caso **Mysql**):

- **Tabla de mensajes**



Mostrando filas 0 - 1 (total de 2. La consulta tardó 0.0008 segundos)

SELECT \* FROM `mensajes`

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Sort by key: Ninguna

+ Opciones

	id_mensaje	mensaje	autor_mensaje	fecha_mensaje
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	Este es un mensaje de prueba	desaextremo	2020-11-04 22:40:35
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	Otro Mensaje de prueba	desaextremo	2020-11-04 22:40:35

- **Procedimiento mysql:** El siguiente procedimiento almacenado recibe como parámetro de entrada el **código** o **id** del mensaje, y el texto del mensaje asociado al **id** recibido como parámetro de salida o retorno.

```
DELIMITER $$
```

```
DROP PROCEDURE IF EXISTS `mensajes_app`.`getMensaje` $$
CREATE PROCEDURE `mensajes_app`.`getMensaje`
    (IN param_id_mensaje INT, OUT param_mensaje VARCHAR(280))
BEGIN
    SELECT mensaje INTO param_mensaje
    FROM mensajes
    WHERE id_mensaje = param_id_mensaje;
END $$
```

```
DELIMITER ;
```



- Proceso creado en Mysql



- Ejecutar proceso des PHPMYADMIN

Seleccionar carpeta Rutinas



Ingresar un valor para el parámetro código y hacer clic en el botón **Continuar**



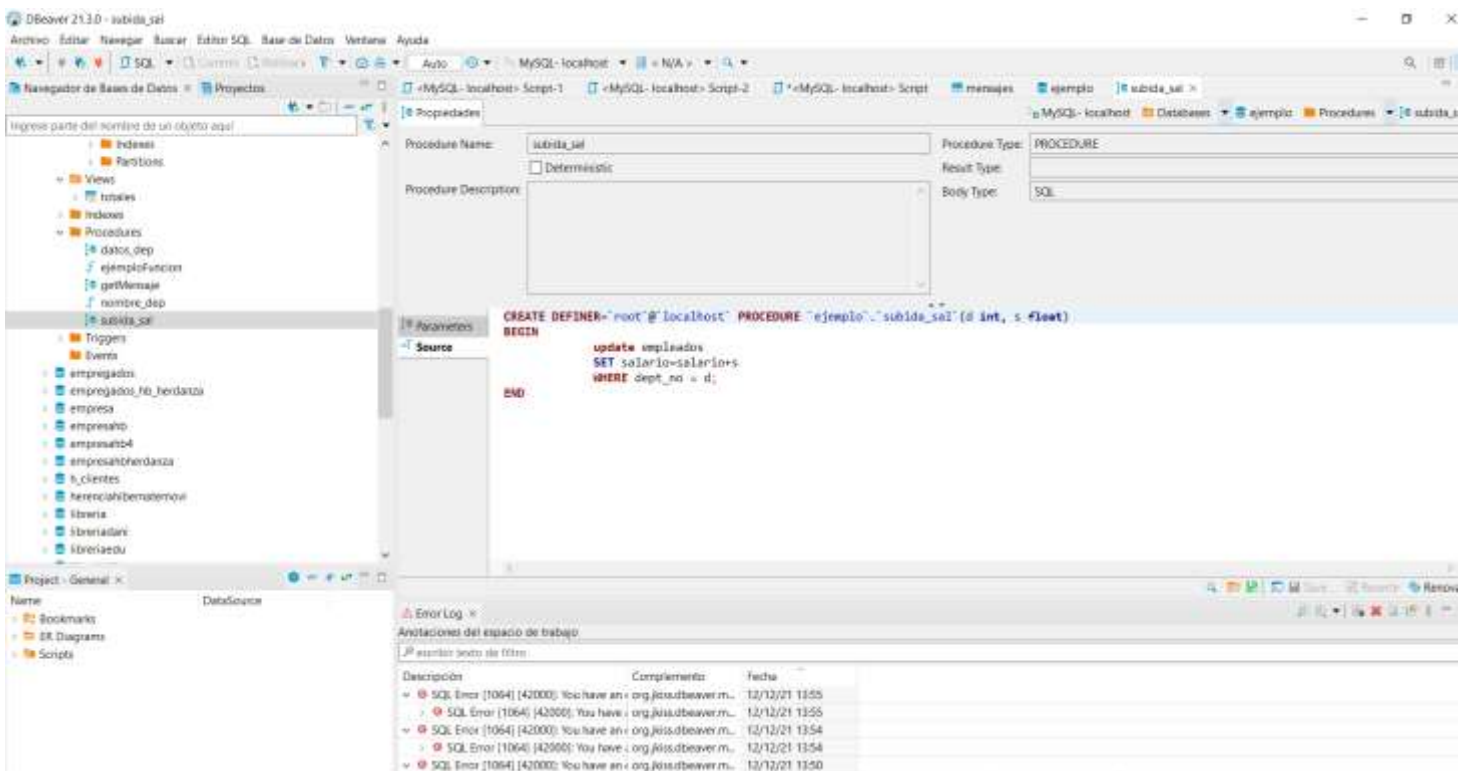
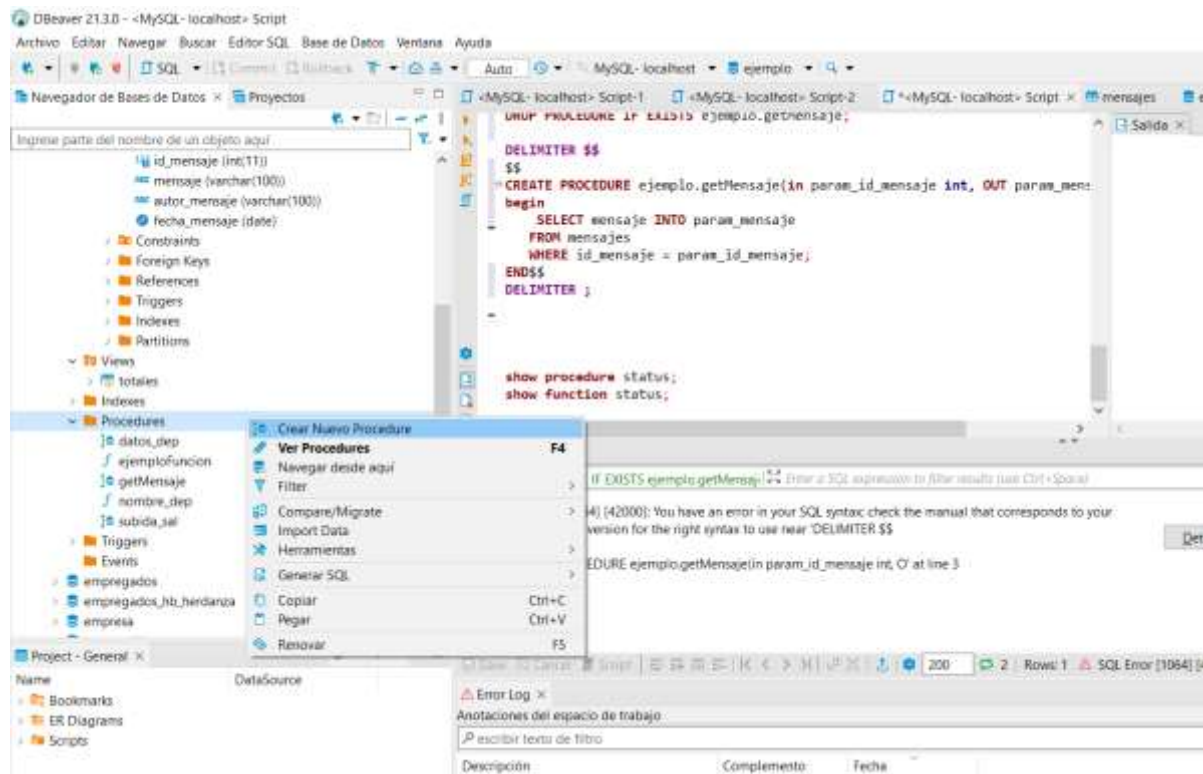
- Ver resultados

<title>localhost / 127.0.0.1 / mensajes\_app | phpMyAdmin 5.0.2</title> ✕

Parámetros de rutina

Nombre	Tipo	Función	Valor
param_id_mensaje	INT	<input type="text" value="v"/>	<input type="text" value="1"/>

- DBeaver



- **Código Java**

```
/*
 * Esta clase se utiliza para invocar el procedimiento almacenado de
 * Mysql
 */
package com.desaextremo.mensajes.test;

import com.desaextremo.mensajes.Conexion;
import java.sql.CallableStatement;
import java.sql.Connection;

/**
 *
 * @author mrnovoa
 */
public class InvocaProceso {

    public static void main(String[] args) {
        //crea conexion
        Conexion conexion = new Conexion();

        //se utiliza para ejecutar proceso
        CallableStatement stmt = null;

        int idMensaje = 1;
        String textoMensaje= null;

        try ( Connection cnx = conexion.get_connection()) {

            System.out.println("Creando sentencia...");
            //el sql de invocación utiliza la notacion call + nombre
            //proceso + parametros requeridos pro el proceso ? separados por ,
            String sql = "{call getMensaje (?, ?)}";
            stmt = cnx.prepareCall(sql);

            //Bind IN parameter first, then bind OUT parameter

            stmt.setInt(1, idMensaje); // Asigna 1 al ide de mensaje
            // El segundo parametro es de salida; pero debe
            registrarse
            stmt.registerOutParameter(2, java.sql.Types.VARCHAR);

            //Invocar el metodo execute parr ejecutar el procedimiento
            //almacenado 'getMensaje' y recuperar el resultado
            System.out.println("Ejecutando el procedimiento almacenado
            getMensaje...");
            stmt.execute();

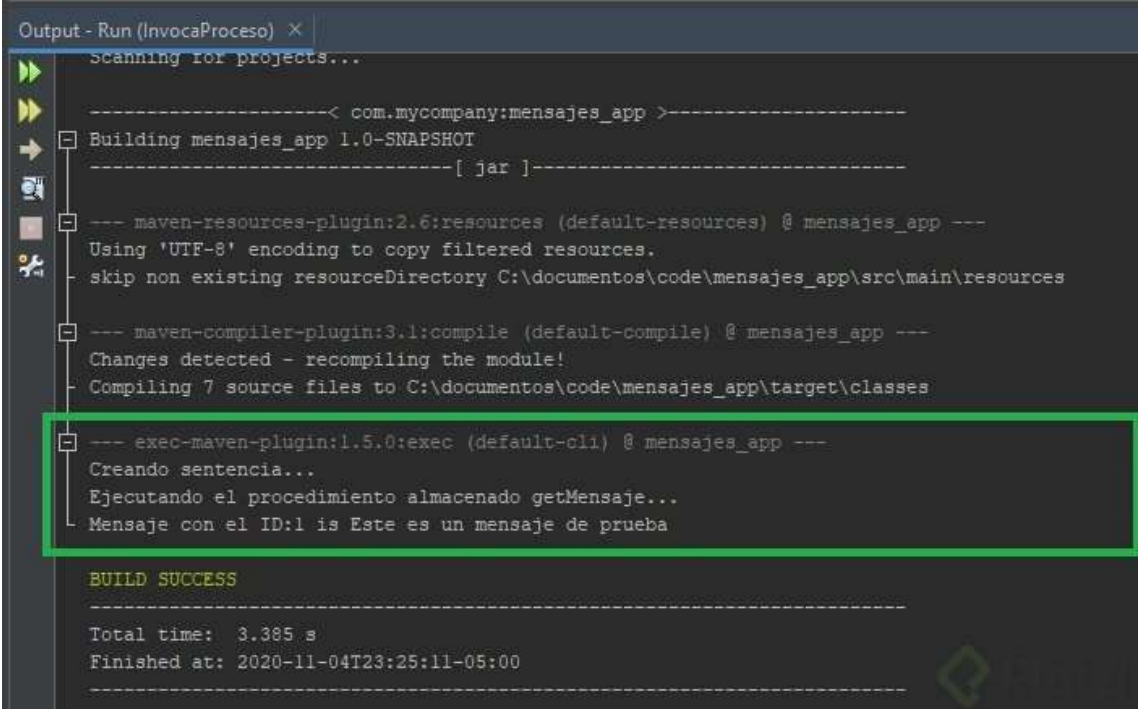
            //Recupera el texto del mensaje y lo imprime en la consola
            textoMensaje = stmt.getString(2);

            System.out.println("Mensaje con el ID:"
                               + idMensaje + " is " + textoMensaje);
            //cerrar recursos
            stmt.close();
            cnx.close();

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
}  
}
```

- Resultado en consola



```
Output - Run (InvocaProceso) X  
Scanning for projects...  
  
-----< com.mycompany:mensajes_app >-----  
Building mensajes_app 1.0-SNAPSHOT  
-----[ jar ]-----  
  
--- maven-resources-plugin:2.6:resources (default-resources) @ mensajes_app ---  
Using 'UTF-8' encoding to copy filtered resources.  
- skip non existing resourceDirectory C:\documentos\code\mensajes_app\src\main\resources  
  
--- maven-compiler-plugin:3.1:compile (default-compile) @ mensajes_app ---  
Changes detected - recompiling the module!  
- Compiling 7 source files to C:\documentos\code\mensajes_app\target\classes  
  
--- exec-maven-plugin:1.5.0:exec (default-cli) @ mensajes_app ---  
Creando sentencia...  
Ejecutando el procedimiento almacenado getMensaje...  
Mensaje con el ID:1 is Este es un mensaje de prueba  
  
BUILD SUCCESS  
-----  
Total time: 3.385 s  
Finished at: 2020-11-04T23:25:11-05:00  
-----
```