**TABLA DE CONTENIDO**

[**¿Qué es JPA ( Java Persistence API)?**](#_s6e8iyhqoyf2) **2**

[**Bases de datos relacionales**](#_fcggf4wqpran) **3**

[**Presentación del proyecto**](#_7fbhpyox4ac6) **4**

[**Diseñar y crear BD**](#_6rs7b6ehep84) **4**

[**Instalación de MySQL**](#_snp8f6fxj2ln) **4**

[Instalación de MySQL](#_kguav1r1szfu) 4

[**Diseñando la base de datos**](#_6biikz6ahn30) **8**

[**Creando nuestra base de datos usando DDL**](#_ivwn93sge3xe) **8**

[**Instalación de Netbeans**](#_2u8om3kbgvwl) **10**

[**Realizar operaciones CRUD y generar conexión**](#_intksc5orsy5) **10**

[**Creación del proyecto en Java**](#_h7vcwmyk2au7) **10**

[**Conexión a MySQL desde Java**](#_t7sc6hk1mlms) **12**

[**Flujo y lógica de la aplicación**](#_7hn25tecwz7f) **16**

[**CRUD: inserción de datos**](#_xbxikyg6gwz5) **22**

[**CRUD: lectura de datos**](#_rcgy8gbj958s) **27**

[**BIBLIOGRAFÍA**](#_vxt3fozg2ecx) **35**

# **¿Qué es JPA ( Java Persistence API)?**

La persistencia de datos se puede realizar actualmente de dos formas: una es en la memoria RAM y otra es en el disco duro mediante archivos o mediante bases de datos.

El almacenamiento en memoria RAM suele ser más rápido, pero tiene un problema y es que la memoria RAM es volátil (cuando la memoria se apaga se pierden los datos).

En el disco duro tenemos una persistencia duradera y tenemos dos formas: los archivos y las bases de datos. Pero con los archivos es un poco más difícil tomar información de varios archivos para procesarla. Mientras que las BBDD son la respuesta a tener mucha información segregada.

JPA es un ORM (Object Relational Mapping) que tiene como objetivo lograr la persistencia de datos entre una aplicación desarrollada en java y una base de datos.

JPA busca traducir el modelado de las clases de Java a un modelado relacional en una base de datos ,posibilitando al programador elegir qué clases u objetos quiere persistir.

**¿Cómo funciona JPA?**

JPA se vale de una serie de mapeos que se deben realizar sobre cada uno de los elementos de la clase, los mismos se representan mediante annotations(@).

JPA cuenta con proveedores JPA , entre ellos : Eclipselink, Hibernate, Toplink entre otros.



**HISTORIA:**

**¿Cómo surgieron las primeras bases de datos relacionales?**

Las primeras bases de datos fueron las bibliotecas hace cientos de años. Fueron los primeros lugares donde se empezó a catalogar, clasificar y ordenar la información.

Cuando empezó la industria de la computación (años 40 y 50) un grupo de personas expertas en el tema llamados CODASYL indagaron sobre cómo hacer sistemas de información que permitiera almacenar la información y procesarla de una manera más ágil. De este grupo un profesional llamado Edgar Codd sugirió el modelo relacional (que es la forma en que la información se pueda almacenar por medio de relaciones a través de tablas).

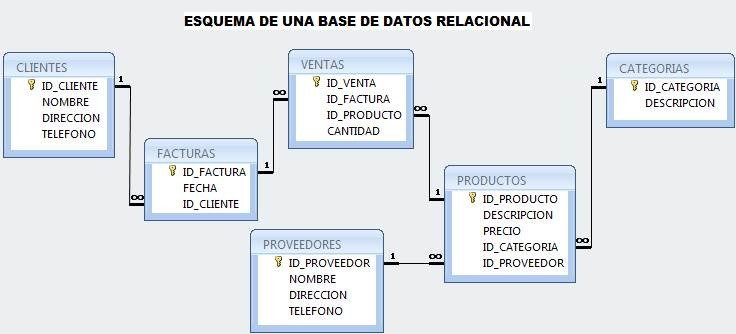
Luego a partir de los años (60’s 70’s)empezaron a aparecer las primeras bases de datos relacionales y también apareció el lenguaje SQL que fue la respuesta a cómo tener un estándar para poder trabajar con la información en las BBDD (cómo consultar la información, cómo guardarla, cómo editarla).

La característica principal de las BBDD relacionales es que se relacionan entre si por medio de una jerarquía. Tenemos las bases de datos que a su vez se descomponen en tablas y estas tablas trabajan con columnas y con filas. La unión de estas se llaman tuplas. Entonces la información que tenemos en estas tablas están relacionadas de una u otra forma.

A través de ese estándar han surgido muchas BBDD relacionales. Luego aparecieron las BBDD no relacionales llegando a los años 2000 's que tienen una forma diferente de almacenar la información.

# **Bases de datos relacionales**

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos.

**Conociendo algunas bases de datos relacionales: MySQL, PostgreSQL, Oracle**

Las bases de datos relacionales se pueden encontrar dentro de las más utilizadas MySQL , PostgreSQL Y Oracle quienes manejan en estándar de modelo entidad relación y el concepto transaccional de manipulación de datos.

A Continuación los enlaces para la descarga de estas bases de datos , para este módulo se utilizará MYSQL con PHPMyadmin el cual es un servidor que permite administrar de manera visual las bases de datos creadas en MySQL.

**SQLite Home Page**

https://www.sqlite.org

**MySQL**

https://www.mysql.com

**MariaDB Foundation - MariaDB.org**

https://mariadb.org

**PostgreSQL: The world's most advanced open source database**

https://www.postgresql.org

**Base de datos - Oracle | Oracle España**

https://www.oracle.com/es/database/

**Sql Server**

<https://www.microsoft.com/es-es/sql-server/sql-server-downloads>

# **Presentación del proyecto**

El proyecto que se realizará en este módulo será una aplicación de recepción de mensajes estilos Twitter , donde el usuario podrá crear un mensaje ,editarlo, listarlos y posteriormente eliminarlo , este proyecto se realizará por medio de JAVASE es decir no tendrá interfaz , todos los datos serán capturados por consola.

# **Diseñar y crear BD**

# **Instalación de MySQL**

Xampp es un paquete que contiene varias herramientas para crear aplicaciones web: un servidor web apache, la base de datos MySQL, PHPMyAdmin y PHP.

En nuestro proyecto vamos a usar MySQL como base de datos relacional y apache junto con phpmyadmin para gestionar nuestra base de datos desde una interfaz web.

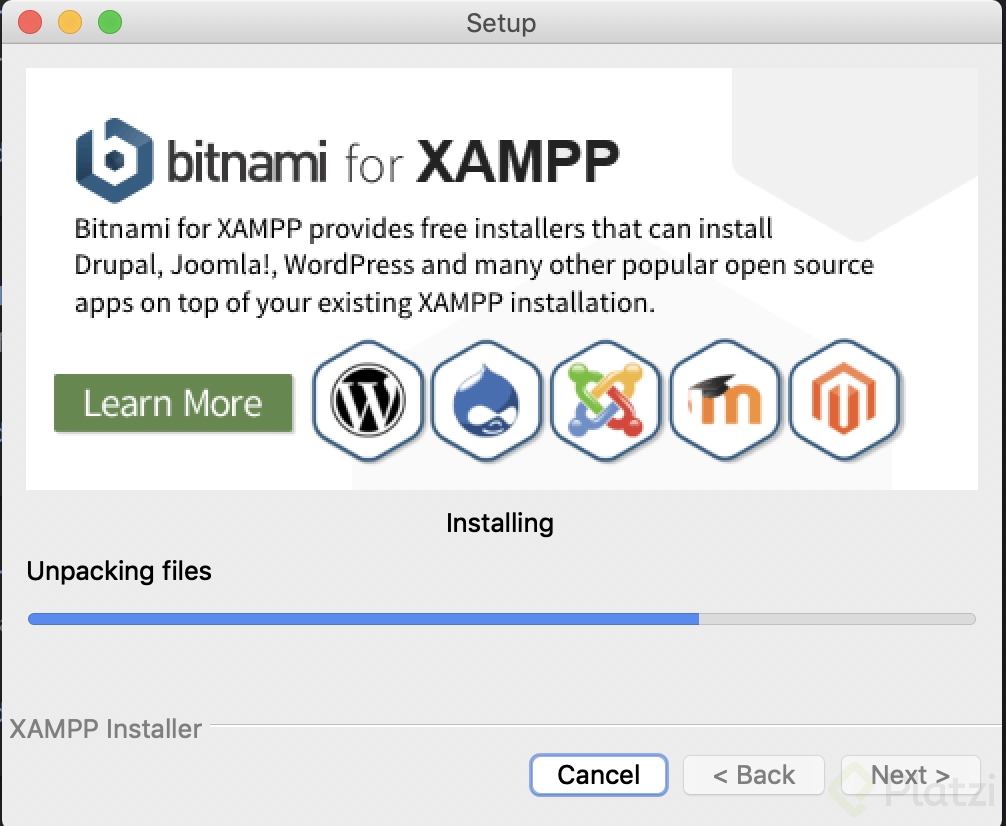
Para instalarlo [vamos al sitio web](https://www.apachefriends.org/es/download.html) y seleccionamos el instalador de acuerdo a nuestro sistema operativo.

## **Instalación de MySQL**

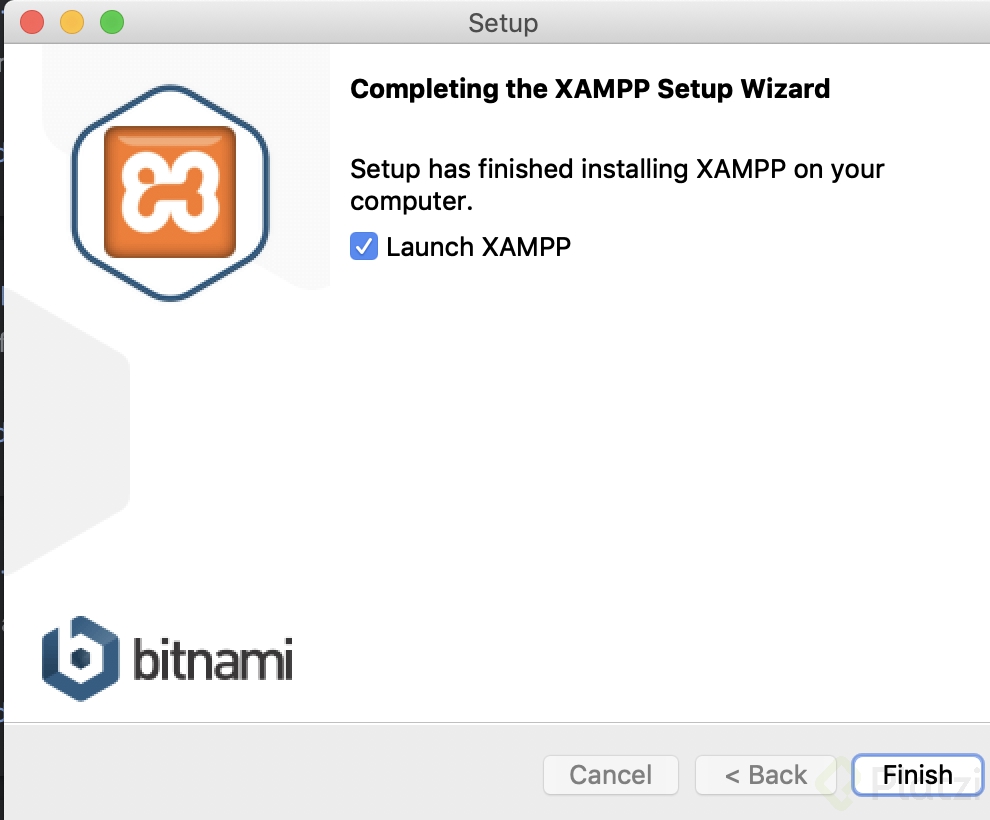
Para instalar nuestra base de datos utilizaremos un gestor llamado **XAMPP**, un software que contiene empaquetado el servidor web apache, mysql como base de datos, php como lenguaje de backend y proftpD como servidor ftp. En nuestro caso, solo necesitamos usar mysql como base de datos y apache para gestionar nuestra base de datos desde la herramienta web **PhpMyAdmin.**

En el sitio web ofrecen varios [instaladores](https://www.apachefriends.org/es/download.html), pero vamos a buscar el instalador clásico que está disponible para [Windows](https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/7.3.11/xampp-windows-x64-7.3.11-0-VC15-installer.exe/download), [Linux](https://sourceforge.net/projects/xampp/files/XAMPP%20Linux/7.3.11/xampp-linux-x64-7.3.11-0-installer.run/download) y [Mac](https://sourceforge.net/projects/xampp/files/XAMPP%20Mac%20OS%20X/7.3.11/xampp-osx-7.3.11-0-installer.dmg/download).

Una vez descargamos el instalador, realizamos la instalación (generalmente es un siguiente, siguiente, muy común).



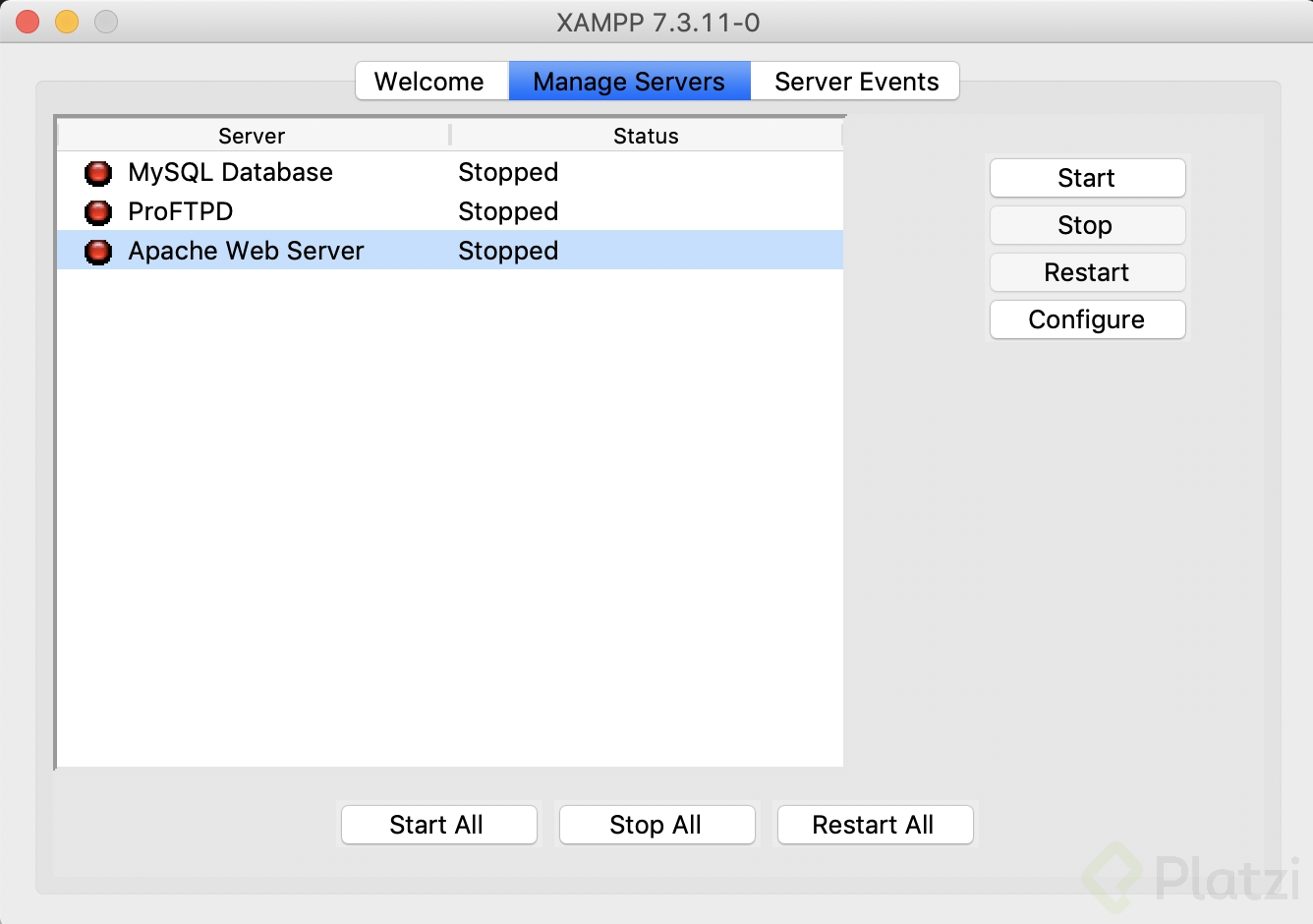
Al finalizar el instalador podemos seleccionar “Lanzar XAMPP”:



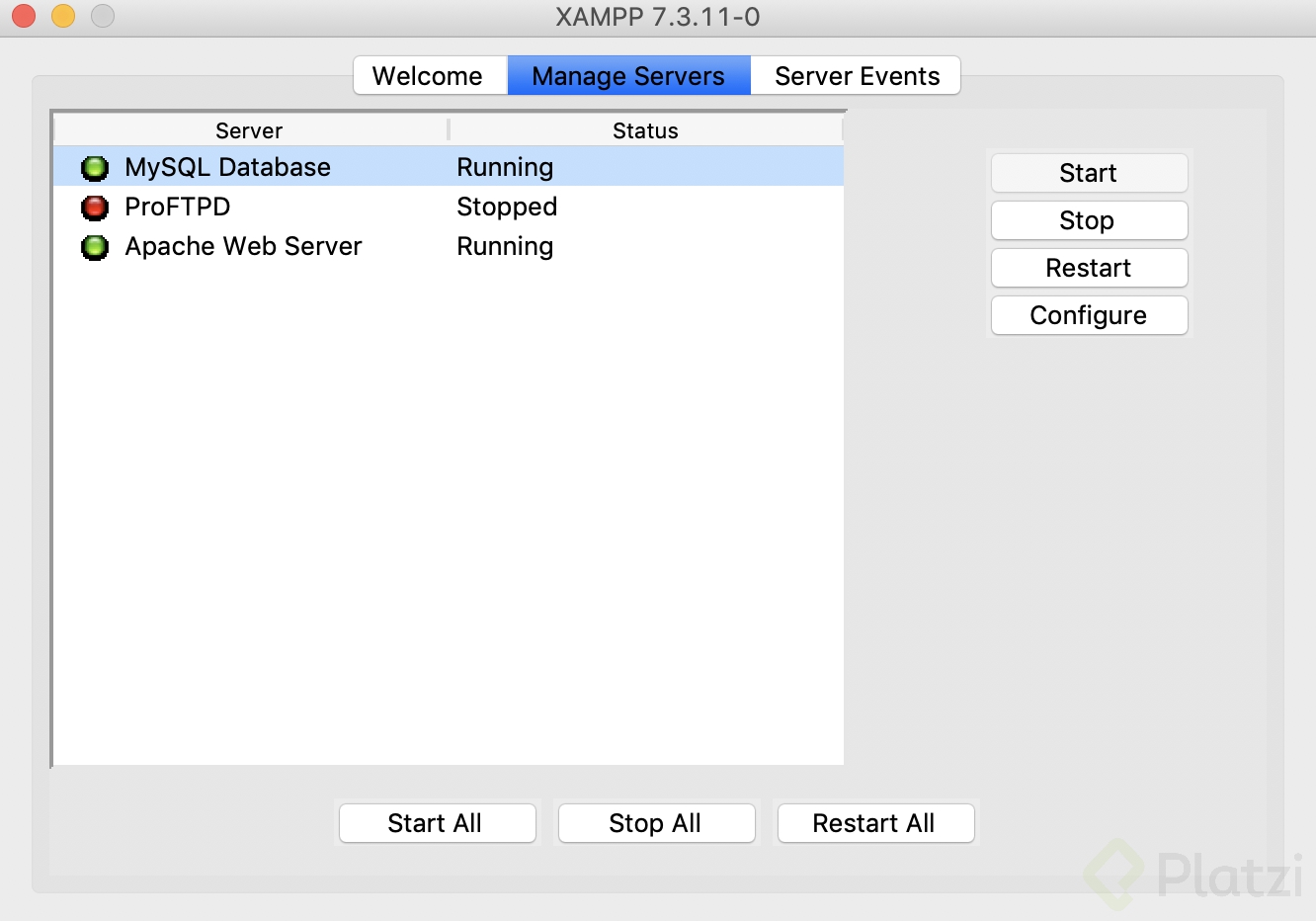
Allí nos aparece el gestor Xampp, vamos a ir a la pestaña **Manage Servers**:



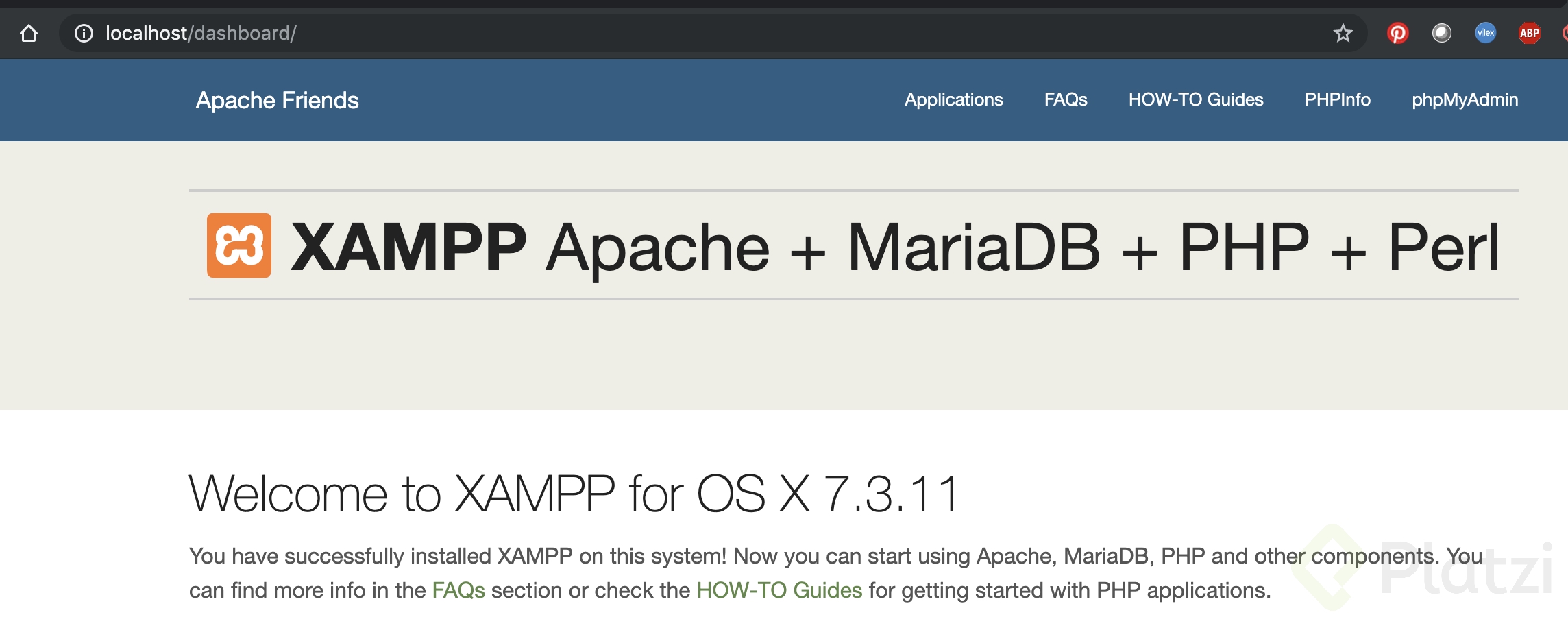
En dicha pestaña vamos a seleccionar el servidor web y de base de datos y vamos a seleccionar en **Start**:



Nos daremos cuenta si estos servicios están encendidos porque los íconos quedan en color verde:

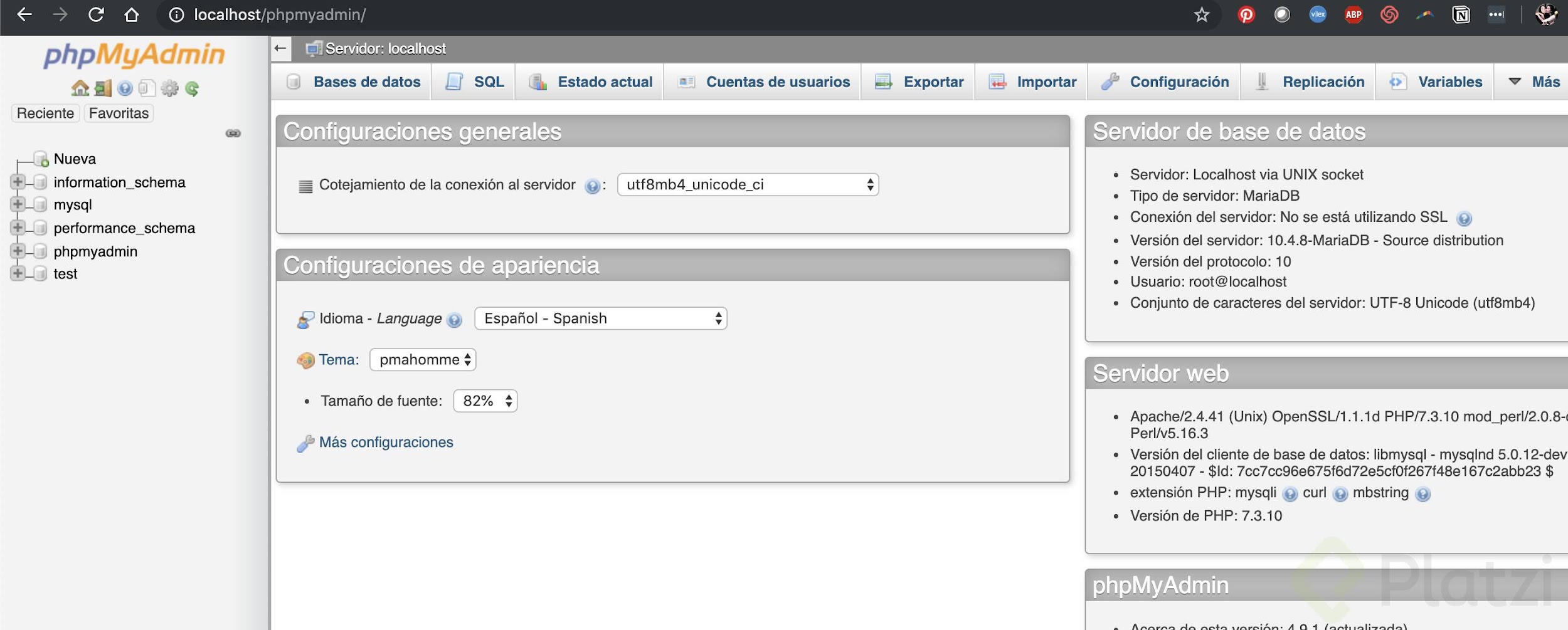


En nuestro navegador vamos a la dirección<http://localhost> o<http://localhost/dashboard/> y tendremos un menú como el siguiente:



Vamos a seleccionar la última opción en el menú superior **PhpMyAdmin**,

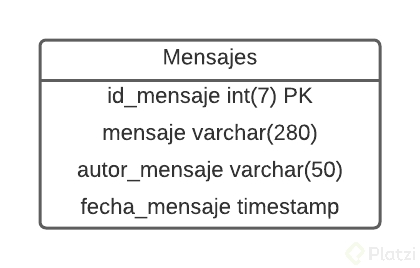
que nos debe abrir un enlace similar a este:<http://localhost/phpmyadmin>.



Y con esto podremos gestionar nuestra base de datos MySQL.

# **Diseñando la base de datos**

Para el diseño de la base de datos podemos utilizar draw.io o fluid char, estas son herramientas de uso libre que permite la construcción del modelo de la aplicación a desarrollar.



# **Creando nuestra base de datos usando DDL**

Levantamos los servicios de xampp y creamos la base de datos , luego creamos las tablas y sus respectivos valores .

Sentencias de DDL (Data Definition Language)

CREATE: Utilizado para crear nuevas tablas, campos e índices.

ALTER: Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

DROP: Empleado para eliminar tablas e índices.

TRUNCATE: Empleado para eliminar todos los registros de una tabla.

COMMENT: Utilizado para agregar comentarios al diccionario de datos.

RENAME: Tal como su nombre lo indica es utilizado para renombrar objetos.

¿Qué es DML? Lenguaje de Manipulación de Datos

DML significa Data Manipulation Language o Lenguaje de Manipulación de Datos, en español. Este lenguaje permite realizar diferentes acciones a los datos que se encuentran en una base de datos.

Permite recuperar, almacenar, modificar, eliminar, insertar y actualizar datos de una base de datos.

Elementos del DML (Data Manipulation Language)

SELECT: Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.

INSERT: Utilizado para cargar de datos en la base de datos en una única operación.

UPDATE: Utilizado para modificar los valores de los campos y registros especificados

DELETE: Utilizado para eliminar registros de una tabla de una base de datos.

¿Qué es DCL? Lenguaje de Control de Datos

Permite crear roles, permisos e integridad referencial, así como el control al acceso a la base de datos.

Elementos del DCL (Data Control Language)

GRANT: Usado para otorgar privilegios de acceso de usuario a la base de datos.

REVOKE: Utilizado para retirar privilegios de acceso otorgados con el comando GRANT.

¿Qué es TCL? Lenguaje de Control Transaccional

Permite administrar diferentes transacciones que ocurren dentro de una base de datos.

Elementos del TCL\*\*(Transactional Control Language)\*\*

COMMIT: Empleado para guardar el trabajo hecho.

ROLLBACK: Utilizado para deshacer la modificación que hice desde el último COMMIT.

create table mensajes\_app.mensajes(

ID\_MENSAJE INT(7) AUTO\_INCREMENT,

MENSAJE VARCHAR(280) NOT NULL,

AUTOR\_MENSAJE VARCHAR(50) NOT NULL,

FECHA\_MENSAJE timestamp NOT NULL,

PRIMARY KEY(ID\_MENSAJE)

);

# **Instalación de Netbeans**

<https://netbeans.apache.org/download/index.html>

Para realizar la instalación de Netbeans seguir el siguiente vídeo pracitco.

Instalacion Windows

<https://www.youtube.com/watch?v=2Et13pH2484&ab_channel=CharlyCimino>

Instalación en Linux

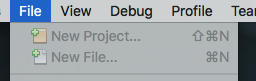
<https://www.youtube.com/watch?v=zCJpPZ548Oo&ab_channel=solvetic.com>

# **Realizar operaciones CRUD y generar conexión**

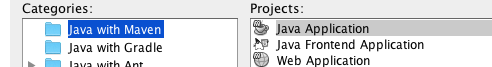
# **Creación del proyecto en Java**

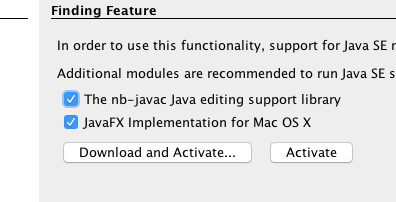
Para crear el proyecto en Netbeans realizamos los siguientes pasos:

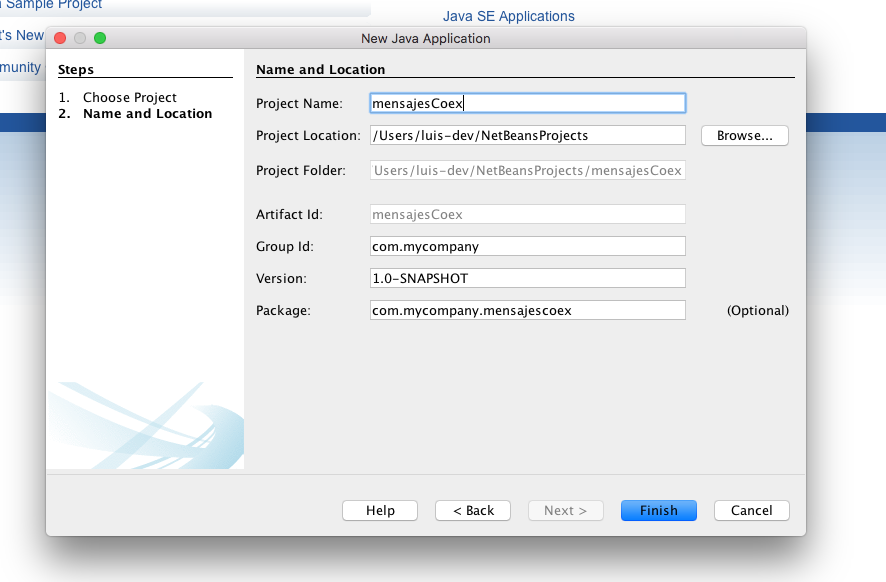
1: Pestaña

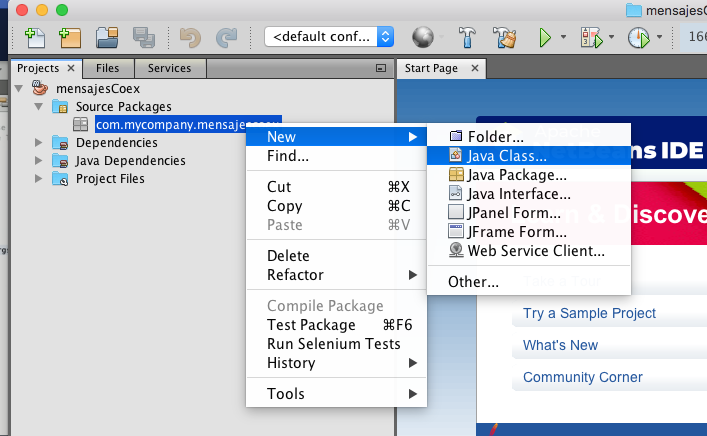


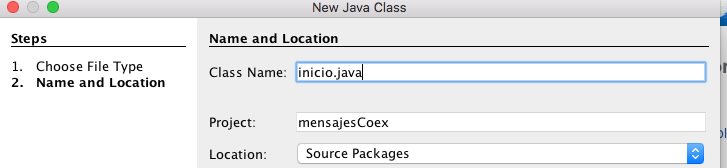
2.

3.

4. (Si es la primera vez)

5. 

6. 

7. 

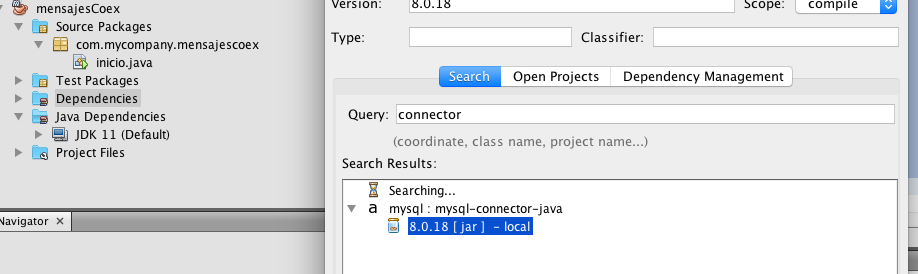
# **Conexión a MySQL desde Java**

Para realizar la conexión realizaremos la importación de las dependencias .jar a nuestro proyecto esta conexión la debemos realizar en la carpeta dependencies , clic derecho agregar dependencia.

si no tenemos la librería de dependencias la podemos descargar por este enlace.

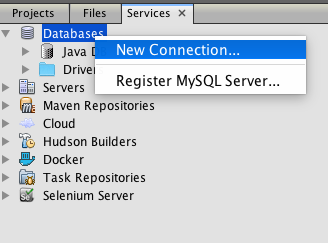
<https://dev.mysql.com/downloads/connector/j/>

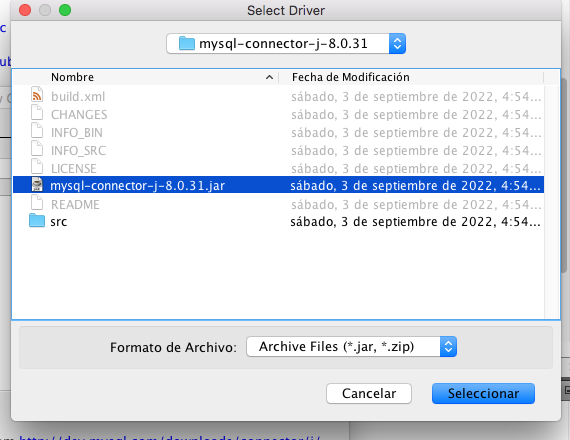


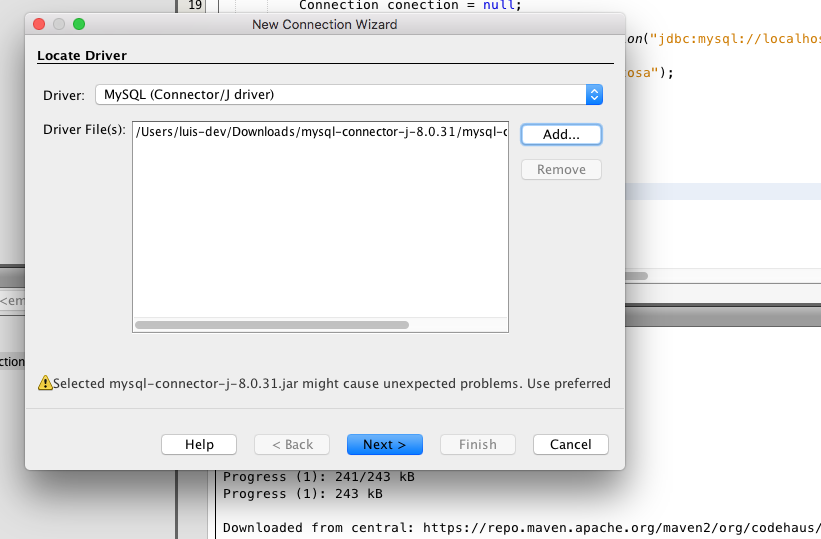


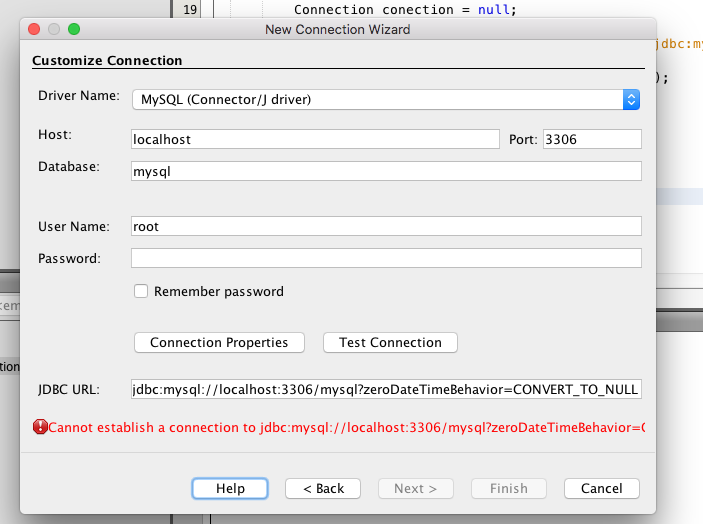
otra forma es en la pestaña services

clic derecho y agregamos conexión .



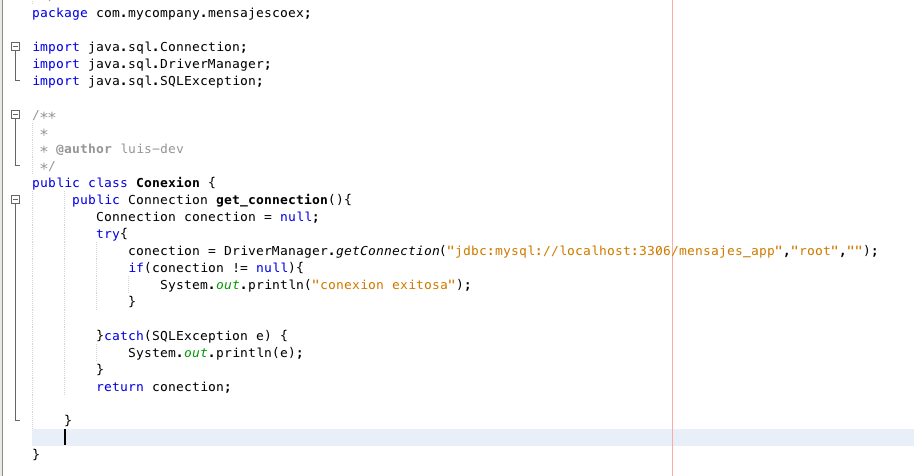




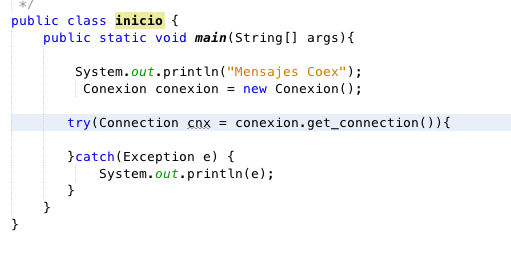


En esta ventana debemos tener activados los servicios de xampp y revisar si tenemos contraseña o usuario junto con el nombre de la base de datos y el puerto de enlace .

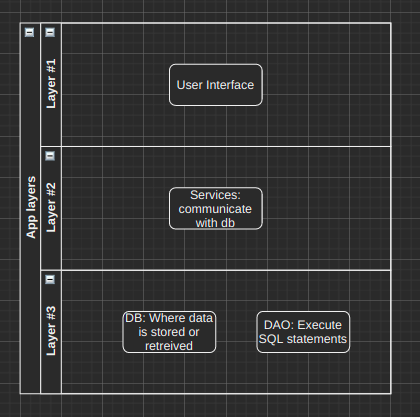
Luego de realizar este proceso debemos indicar un método estándar de conexión a nuestra base de datos esto lo realizaremos en un nuevo archivo llamado conexion y aquí daremos los parámetros para la configuración de la conexión a mysql.



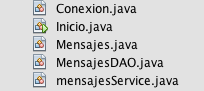
Para realizar la prueba nos dirigimos a nuestra clase inicio y aqui damos los parámetros para cerrar el objeto conexión y poder llamar el método de la clase Conexion.



# **Flujo y lógica de la aplicación**



Lo primero que debemos realizar es identificar cuales son las clases que vamos a utilizar , en este caso vamos a crear las siguientes clases.



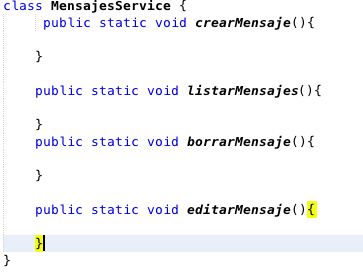
Después de crear las diferentes clases, crearemos la lógica del negocio.

Primero en la clase Mensajes crearemos , los constructores de los atributos que tendrá en sí nuestra entidad Mensajes y luego crearemos los getter y setter para administrar los comportamientos del mensaje.

| public class Mensajes {    int id\_mensaje;  String mensaje;  String autor\_mensaje;  String fecha\_mensaje;  public Mensajes(){  }  public Mensajes(String mensaje, String autor\_mensaje, String fecha\_mensaje) {  this.mensaje = mensaje;  this.autor\_mensaje = autor\_mensaje;  this.fecha\_mensaje = fecha\_mensaje;  }    public int getId\_mensaje() {  return id\_mensaje;  }  public void setId\_mensaje(int id\_mensaje) {  this.id\_mensaje = id\_mensaje;  }  public String getMensaje() {  return mensaje;  }  public void setMensaje(String mensaje) {  this.mensaje = mensaje;  }  public String getAutor\_mensaje() {  return autor\_mensaje;  }  public void setAutor\_mensaje(String autor\_mensaje) {  this.autor\_mensaje = autor\_mensaje;  }  public String getFecha\_mensaje() {  return fecha\_mensaje;  }  public void setFecha\_mensaje(String fecha\_mensaje) {  this.fecha\_mensaje = fecha\_mensaje;  } |
| --- |



El siguiente paso es ir a la clase MensajesService y crear los métodos correspondientes a crear listar borrar editar.

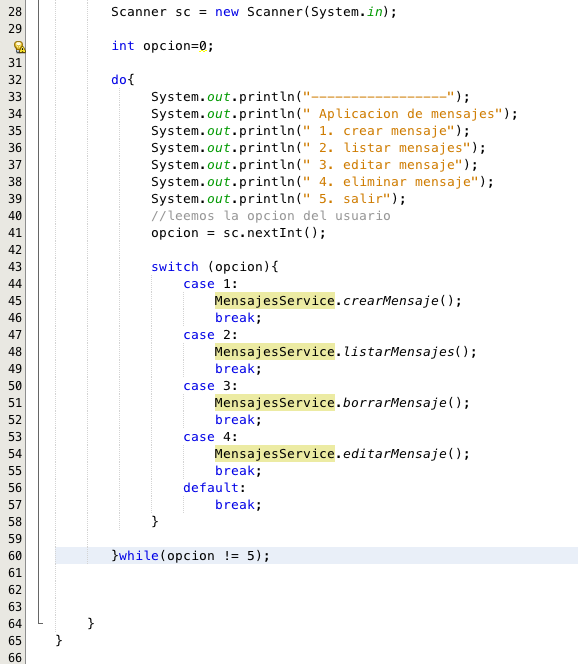


| class MensajesService {  public static void crearMensaje(){    }    public static void listarMensajes(){    }  public static void borrarMensaje(){    }    public static void editarMensaje(){    }  } |
| --- |

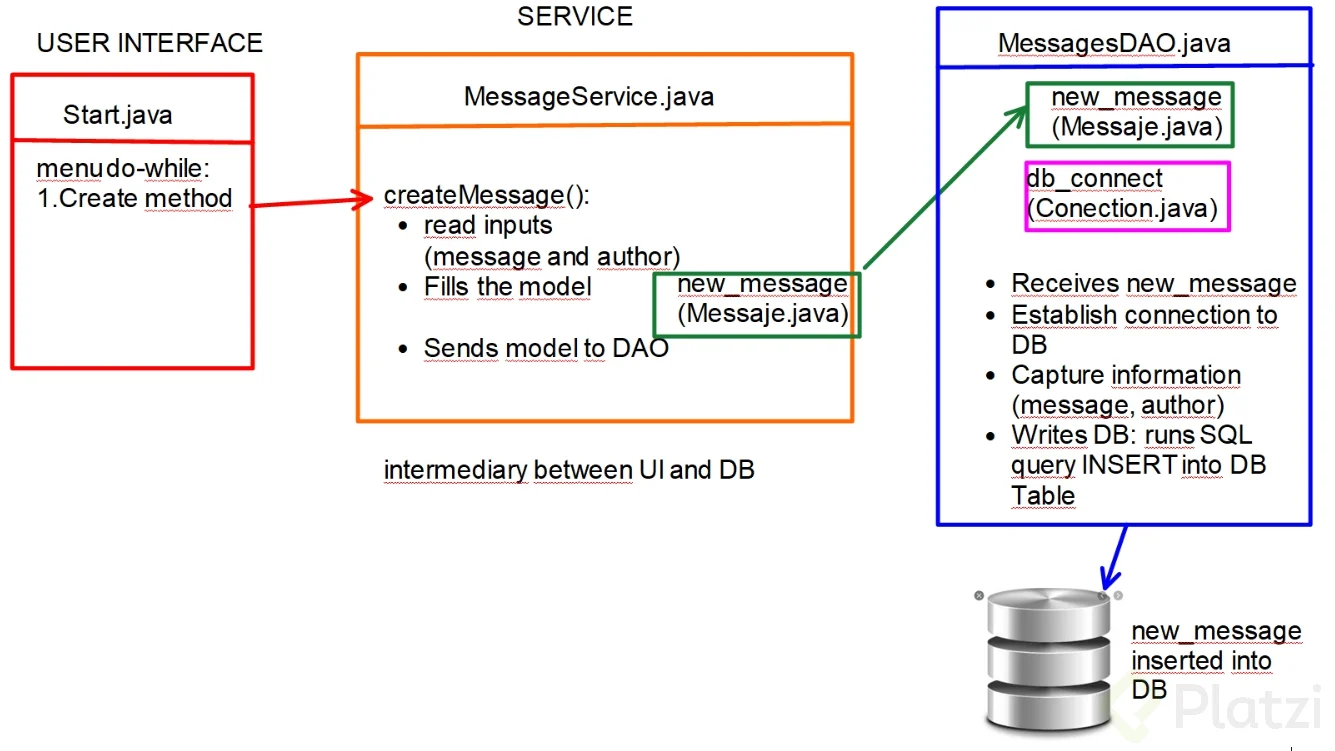
Luego de tener nuestra capa de mensajes y nuestra capa de service procedemos a crear en el inicio del programa un menú para seleccionar los diferentes métodos de nuestro CRUD.

Para este caso comentaremos el código anterior de conexión y luego importamos java util scanner que nos permite agregar información por teclado.





| LÍNEA | EXPLICACIÓN |
| --- | --- |
| 28 | Instanciamos el método scanner para poder recibir los datos desde la consola. |
| 30 | se crea la opción opción iniciando en cero ,para posteriormente ser usada. |
| 32 | iniciamos un bucle do - while |
| 33-40 | Se generan los mensajes en pantalla para la selección del menú del programa. |
| 41 | los datos almacenados en la consola se almacenan en la variable opción esta se guarda en memoria para ser usada después. |
| 43 | creamos un ciclo switch con los datos almacenados en opción |
| 45 | si opción es igual a 1 llamamos al método crearMensaje ubicando en MnesajeService |
| 48 | si opción es igual a 1 llamamos al método crearMensaje ubicando en MnesajeService |
| 51 | si opción es igual a 1 llamamos al método crearMensaje ubicando en MnesajeService |
| 54 | si opción es igual a 1 llamamos al método crearMensaje ubicando en MnesajeService |
| 56 | si ningún valor entra el condicional se rompe |
| 60 | Termina do - while opcion sea diferente de 5 |



El siguiente paso es clave para el proceso de datos CRUD , ya que la capa Service será la encargada de pedirnos los datos de los mensajes, el autor y está enviará sus parámetros a la capa DAO la cual será encargada de conectarse a la base de datos y enviar los datos del mensaje .

# **CRUD: inserción de datos**

Para el proceso de insertar datos es necesario tener en cuenta las siguientes líneas de código que dan como referencia la recepción de los datos que se almacenan en el menú principal y luego pasan por la capa DAO que se comunica directamente con la base de datos.

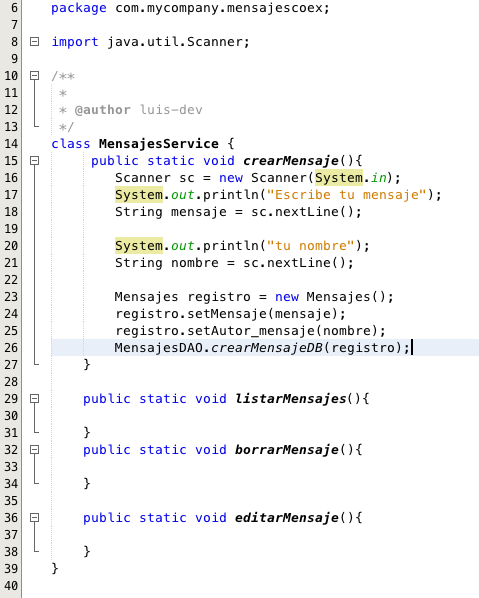


| Línea | Explicación |
| --- | --- |
| 7 | Se importa el módulo connection |
| 8 | Se importa el preparedStatement para ser usado previamente en el código |
| 9 | Se importa el SQLException para ser usado previamente en el código |
| 16 | Se crea la instancia de Conexion con el nombre db\_connect para realizar la conexión a la base de datos. |
| 18 | Permite manejar errores en tiempo de ejecución , en este caso atrapa la conexión , si no tiene error entra . |
| 19 | Guarda la conexión en la variable ps. |
| 20 | inicio de otro try si la conexión se establece |
| 21 | trae el query en sql para el insert de datos |
| 22 | trae el objeto prepareStatement y guarda el query para almacenar los datos. |
| 23 | Se trae el objeto de la clase mensaje .getMensaje y lo almacena en la posición 1 |
| 24 | Se trae el objeto de la clase mensaje .getAutor\_mensaje y lo almacena en la posición 2 |
| 25 | nos devuelve un entero que nos dice el número de registros a los que ha afectado la operación |
| 26 | Genera el mensaje creado |
| 28 | **Catch** señala un bloque de instrucciones a intentar ( **try** ), y especifica una respuesta si se produce una excepción ( **catch** ). |
| 31 | **Catch** señala un bloque de instrucciones a intentar ( **try** ), y especifica una respuesta si se produce una excepción ( **catch** ). |

| package com.mycompany.mensajescoex;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.SQLException;  /\*\*  \*  \* @author luis-dev  \*/  public class MensajesDAO {  public static void crearMensajeDB(Mensajes mensaje){  Conexion db\_connect = new Conexion();    try(Connection conexion = db\_connect.get\_connection()) {  PreparedStatement ps=null;  try{  String query="INSERT INTO mensajes ( mensaje, autor\_mensaje) VALUES (?,?)";  ps=conexion.prepareStatement(query);  ps.setString(1, mensaje.getMensaje());  ps.setString(2, mensaje.getAutor\_mensaje());  ps.executeUpdate();  System.out.println("mensaje creado");    }catch(SQLException ex){  System.out.println(ex);  }  }catch(SQLException e){  System.out.println(e);  }  } |
| --- |

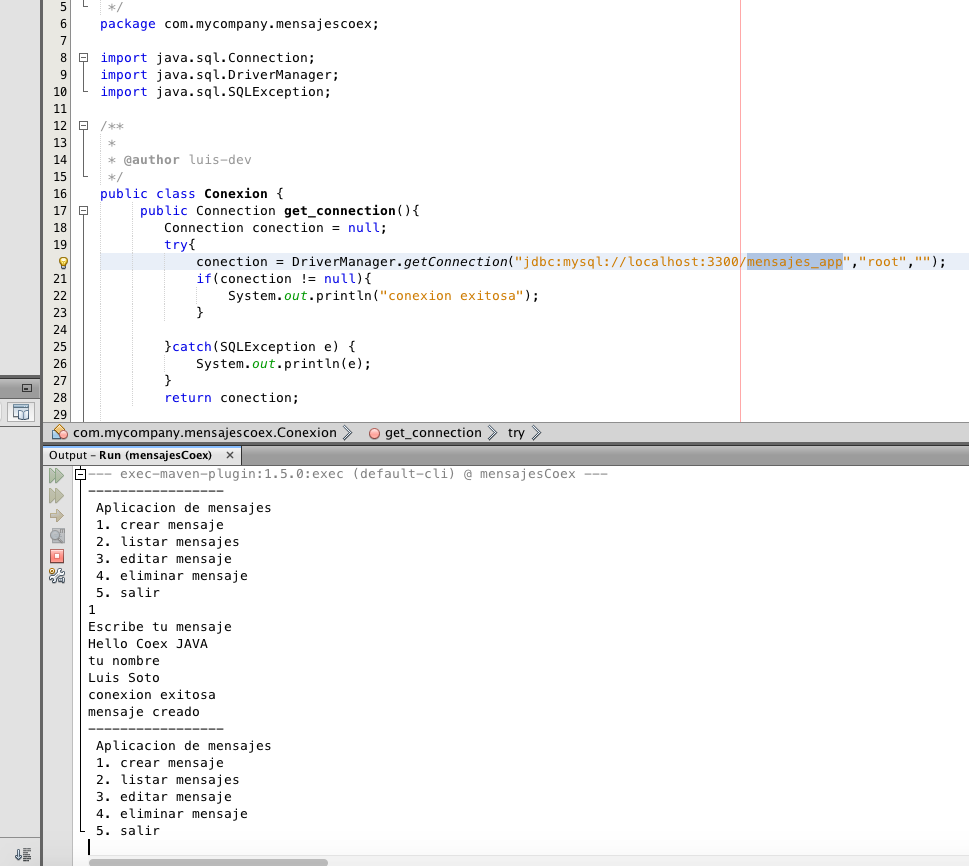
Ahora realizaremos el ingreso del código a la clase mensajesService, quién es el encargado de esperar la solicitud del menú de inicio luego este recibe los valores por teclado al momento de crear el mensaje y este se comunica con MensajesDAO que toma el objeto y lo envía a la base de datos por medio de la instrucción INSERT en SQL .

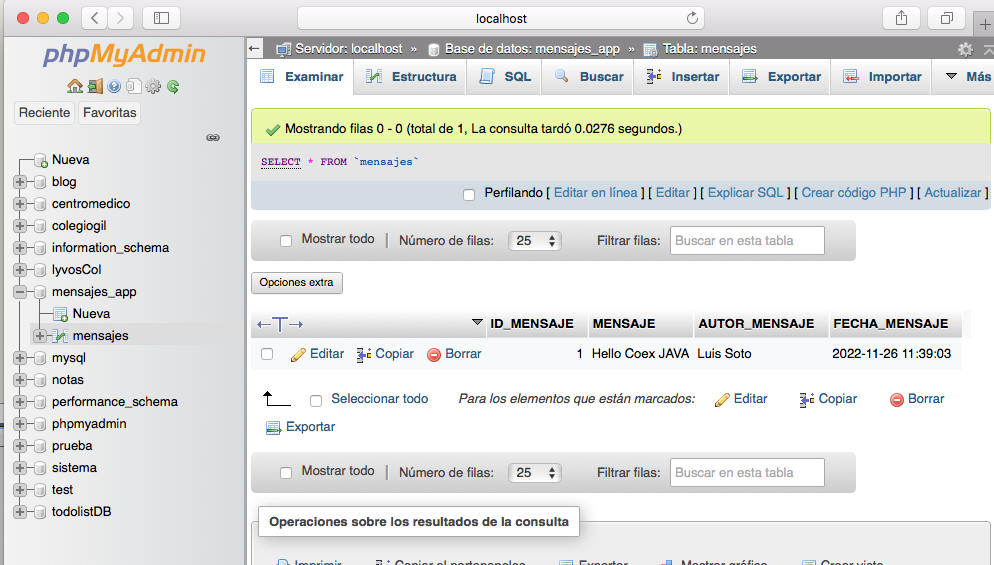
| package com.mycompany.mensajescoex;  import java.util.Scanner;  /\*\*  \*  \* @author luis-dev  \*/  class MensajesService {  public static void crearMensaje(){  Scanner sc = new Scanner(System.in);  System.out.println("Escribe tu mensaje");  String mensaje = sc.nextLine();    System.out.println("tu nombre");  String nombre = sc.nextLine();    Mensajes registro = new Mensajes();  registro.setMensaje(mensaje);  registro.setAutor\_mensaje(nombre);  MensajesDAO.crearMensajeDB(registro);  }    public static void listarMensajes(){    }  public static void borrarMensaje(){    }    public static void editarMensaje(){    }  } |
| --- |



| Línea | Explicación |
| --- | --- |
| 8 | Importa el Scanner para poder recibir por teclado. |
| 16 | Instanciamos la librería scanner para poder ser utilizada, se asigna una variable a la instancia con el nombre de sc. |
| 17 | Se muestra en pantalla el mensaje para que el usuario escriba el mensaje. |
| 18 | la instrucción nextLine() para almacenar los datos en la variable mensaje . |
| 20 | Se muestra en pantalla el mensaje para que el usuario escriba el nombre. |
| 21 | la instrucción nextLine() para almacenar los datos en la variable nombre . |
| 23 | Instanciamos los valores de la clase Mensaje con el fin de obtener los parámetro del mensaje. |
| 24 | Al objeto registro le pasamos los parámetros de mensaje con el set dejándolos fijos y estos se conectan con la clase mensajes. |
| 25 | Al objeto registro le pasamos los parámetros de autor con el set dejándolos fijos y estos se conectan con la clase mensajes. |
| 26 | Se envían los parámetros de crear mensaje para que mensajes DAO reciba estos datos del objeto y posteriormente sean almacenados en la base de datos. |

Realizamos la prueba para revisar el proceso de los datos.

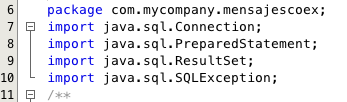


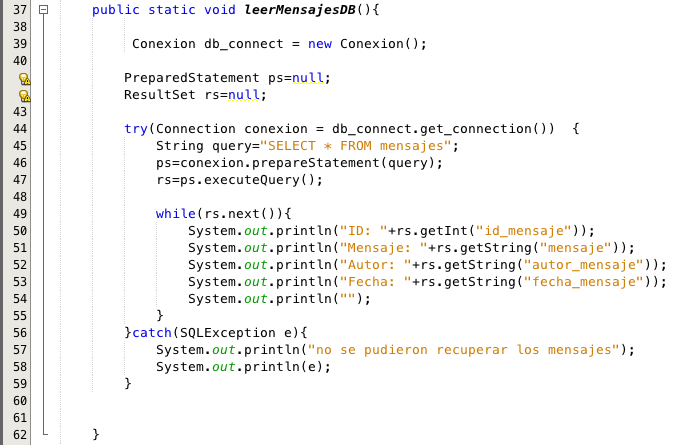


# **CRUD: lectura de datos**

En este espacio veremos cómo realizaremos la lectura de los datos almacenados en la base de datos .

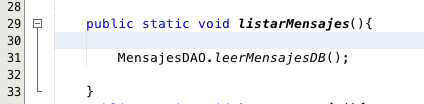
| Conexion db\_connect = new Conexion();    PreparedStatement ps=null;  ResultSet rs=null;    try(Connection conexion = db\_connect.get\_connection()) {  String query="SELECT \* FROM mensajes";  ps=conexion.prepareStatement(query);  rs=ps.executeQuery();    while(rs.next()){  System.out.println("ID: "+rs.getInt("id\_mensaje"));  System.out.println("Mensaje: "+rs.getString("mensaje"));  System.out.println("Autor: "+rs.getString("autor\_mensaje"));  System.out.println("Fecha: "+rs.getString("fecha\_mensaje"));  System.out.println("");  }  }catch(SQLException e){  System.out.println("no se pudieron recuperar los mensajes");  System.out.println(e);  } |
| --- |



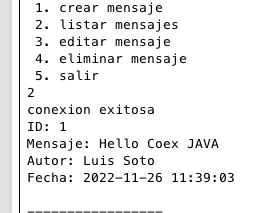


| LÍNEA | EXPLICACIÓN |
| --- | --- |
| 9 | Se realiza la importación de resultSet |
| 39 | Se crea el objeto conexión en la base de datos para verificación de los datos |
| 41 | Se crea la variable ps para de modo vacía para luego ser utilizada por el query de SQL |
| 42 | Usamos el resultSet para luego utilizar el métodos que trae para poder manipular las consultas en SQL. |
| 44 | Creamos el objeto conexión para traer la información de la consulta. |
| 45 | Creamos el query de la instrucción SQL para traer todos los datos que están almacenados en la base de datos. |
| 46 | cargamos lo datos del query a nuestra variable ps . |
| 47 | cargados con resultset los datos encontrados en la consulta y se los pasamos al valor rs para que se organice y se listen todos los datos de la base de datos. |
| 50 | Imprimimos los datos del ID y con getIt concatenamos el el ID de los registros de la base de datos con id\_mensaje. |
| 52 | Imprimimos los datos del Mensaje y con getString concatenamos el el ID de los registros de la base de datos con mensaje. |
| 53 | Imprimimos los datos del autor y con getString concatenamos el el autor de los registros de la base de datos con autor\_mensaje. |
| 54 | Imprimimos los datos del fecha y con getString concatenamos la fecha de los registros de la base de datos con fecha\_mensaje. |
| 57 | si los valores no son atrapados con el try hacemos la excepción e imprimimos los valores no se pudieron recuperar. |

El paso siguiente es conectarnos con mensajesService de esta manera creamos la instancia que traerá los valores del objeto DAO.

****

probamos el ejercicio.

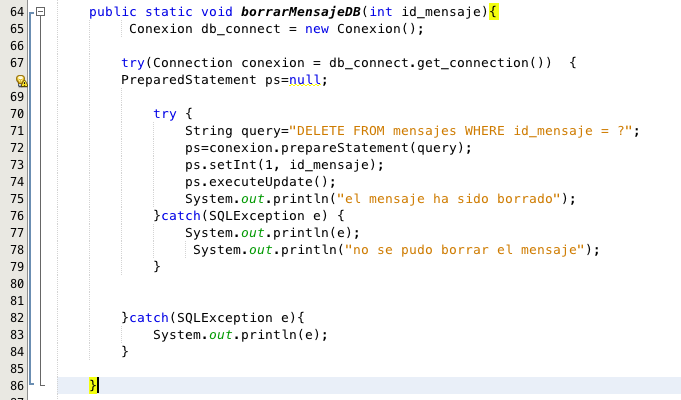


**CRUD: eliminación de datos**

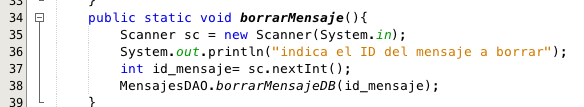
Para el proceso de eliminación tendremos en cuenta los valores que se van a registrar en la base de datos y como sabemos el tiene un id que de forma automática está incrementando el valor de los datos que tenemos , este será nuestro punto de referencia para poder eliminar los datos que tenemos en nuestra base de datos.

a continuación el código.

| public static void borrarMensajeDB(int id\_mensaje){  Conexion db\_connect = new Conexion();    try(Connection conexion = db\_connect.get\_connection()) {  PreparedStatement ps=null;    try {  String query="DELETE FROM mensajes WHERE id\_mensaje = ?";  ps=conexion.prepareStatement(query);  ps.setInt(1, id\_mensaje);  ps.executeUpdate();  System.out.println("el mensaje ha sido borrado");  }catch(SQLException e) {  System.out.println(e);  System.out.println("no se pudo borrar el mensaje");  }      }catch(SQLException e){  System.out.println(e);  }    } |
| --- |



| LÍNEA | EXPLICACIÓN |
| --- | --- |
| 65 | Se realiza la conexión a la base de datos creando el objeto db\_connect |
| 67 | se crea el try para comprobar que la conexión si este establecida |
| 68 | se crea la variable ps para comprobar los datos de la base de datos |
| 71 | se gestiona el query encargado de eliminar los datos cuando sea igual al id |
| 72 | ps carga los datos que trae la consulta |
| 73 | En esta línea se hace la verificación de los datos de manera que por medio del setInt podamos hacer referencia al primer valor de la consulta en este caso id\_mensaje. |
| 74 | ps.execute() es un método que trae los datos de resultset encargado de actualizar después de la consulta |
| 75 | Imprime por pantalla el mensaje se borró |
| 78 | Si el programa gestiona algún error de conexión se carga el catch que envía un mensaje que no se pudo eliminar. |

Ahora vamos a mostrar en pantalla cuando el usuario pueda elegir el valor a eliminar.

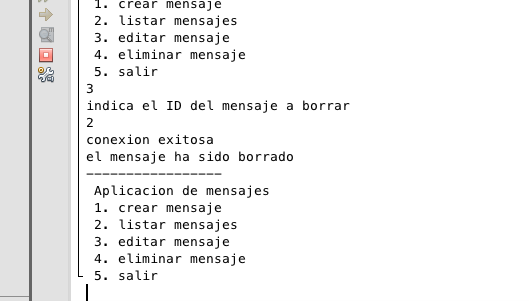
Como podemos ver el sistema en la línea 35 crea el objeto sc para poder ser invocado cuando se ingresen datos por consola.

Línea 36 , se solicita al usuario que seleccione el usuario a borrar

cargamos lo que el usuario digita por consola en la línea 37

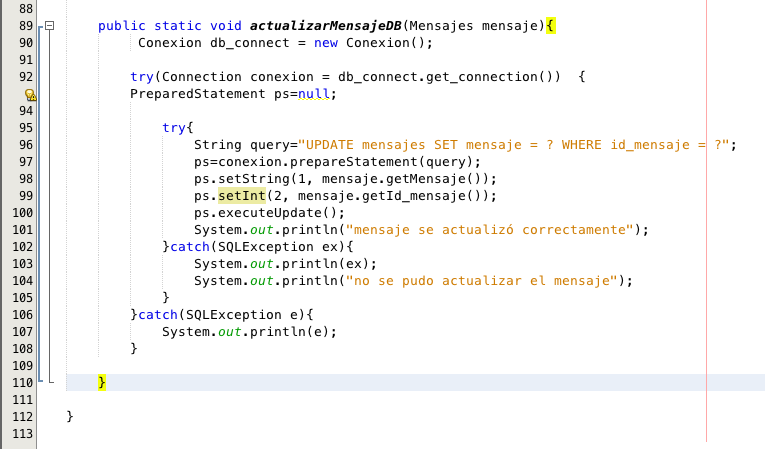
En la página 38 enviamos la información a la capa DAO para poder realizar el método allí ejecutado.

Probamos.

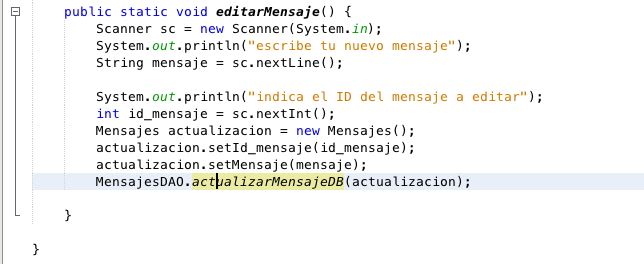


**CRUD: actualización de datos**

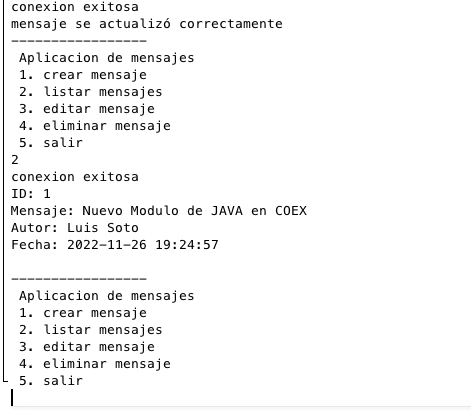
| public static void actualizarMensajeDB(Mensajes mensaje){  Conexion db\_connect = new Conexion();    try(Connection conexion = db\_connect.get\_connection()) {  PreparedStatement ps=null;    try{  String query="UPDATE mensajes SET mensaje = ? WHERE id\_mensaje = ?";  ps=conexion.prepareStatement(query);  ps.setString(1, mensaje.getMensaje());  ps.setInt(2, mensaje.getId\_mensaje());  ps.executeUpdate();  System.out.println("mensaje se actualizó correctamente");  }catch(SQLException ex){  System.out.println(ex);  System.out.println("no se pudo actualizar el mensaje");  }  }catch(SQLException e){  System.out.println(e);  }  }  } |
| --- |



| LÍNEA | EXPLICACIÓN |
| --- | --- |
| 90 | Se realiza la conexión a la base de datos creando el objeto db\_connect |
| 92 | se crea el try para comprobar que la conexión si este establecida |
| 96 | se crea la variable ps para comprobar los datos de la base de datos |
| 97 | se gestiona el query encargado de ACTUALIZAR los datos cuando sea igual al id |
| 98 | ps carga los datos que trae la consulta |
| 99 | En esta línea se hace la verificación de los datos de manera que por medio del setInt podamos hacer referencia al primer valor de la consulta en este caso id\_mensaje. |
| 100 | ps.execute() es un método que trae los datos de resultset encargado de actualizar después de la consulta |
| 101 | Imprime por pantalla el mensaje se actualizo correctamente |
| 104 | Si el programa gestiona algún error de conexión se carga el catch que envía un mensaje que no se pudo actualizar. |



Como vemos en estas líneas de código vamos a recibir los datos por consola del usuario .



# 

# 

# **BIBLIOGRAFÍA**

ECODEUP. (2022, 26 y 11). *patrones de diseño con java* .ecodeup. URL

<https://www.ecodeup.com/patrones-de-diseno-en-java-mvc-dao-y-dto/>

ORACLE. (2022, 22 y 11). *what is a relational database* .oracle . URL

<https://www.oracle.com/co/database/what-is-a-relational-database/>

YOUTUBE. (2022, 20 y 11). *aprende a programar con java* .todo code. URL

<https://www.youtube.com/watch?v=ppDv4N5A31E&list=PLQxX2eiEaqbwNP20GMMCjRslRq2lOLWlg&index=23&ab_channel=TodoCode>

MAKIGAS. (2022, 16 y 11). *patrones de diseño con java* .makigas. URL

<https://www.youtube.com/watch?v=NjY-WA-jeJ8&ab_channel=makigas%3Aaprendeaprogramar>

PLATZI. (2022, 10 y 11). *curso persistencia en java* .platzi . URL

<https://www.youtube.com/watch?v=NjY-WA-jeJ8&ab_channel=makigas%3Aaprendeaprogramar>