
# Tarea 11 - Acceso a bases de datos

En esta última unidad vamos a implementar una pequeña aplicación que utilice una conexión contra una base de datos relacional, que estará montada sobre MariaDB. Sigue los siguientes pasos:

- Siguiendo los pasos de los contenidos, instala en tu equipo MariaDB y HeidiSQL como aplicación para acceso y administración de bases de datos en MariaDB.
- Realizada la instalación, inicia el SGBD MariaDB e inicia la aplicación HeidiSQL. Deberás introducir las credenciales de conexión a MariaDB.
- Crea una base de datos desde HeidiSQL denominada Concesionario.
- Dentro de esta base de datos deberás crea dos tablas:
  - Vehículos: almacenará una serie de vehículos.
    - Propietarios: almacenará los propietarios de los vehículos de la tabla anterior.

Utilizando HeidiSQL, inserta datos en ambas tablas que pueden servir para realizar pruebas. Introduce propietarios que pueden tener varios vehículos.

Para todo lo anterior, crearemos un documento llamado ***apellido1_apellido2_nombre_PROG_Tarea11_docBD*** donde documentaremos todo lo anterior. Es necesario añadir capturas para demostrar la realización de todos los apartados anteriores (instalación, configuración, creación de BD, conexión, creación de tablas e

inserción de datos). Toda captura debe estar acompañada de una explicación representativa de lo realizado.

Además deberá adjuntarse el script completo .sql con la creación de la BD, tablas, inserciones y reglas llamado “apellido1_apellido2_nombre_PROG_Tarea11_Script.sql".

Hasta este punto tenemos la parte de configuración del SGBD. El objetivo de nuestra aplicación es implementar varias clases que se encargue de realizar operaciones contra la base de datos. Para ello:

- Crea un nuevo proyecto denominado PROG11_Tarea.

    - Crea un paquete denominado com.prog11.bbdd.
        - Dentro del paquete se crea la clase ConnectionDB que se encargue de realizar la conexión contra la base de datos y el cierre de la misma. Para ello implementará los métodos openConnection() y closeConnection().
        - Crea una clase estática denominada PropietariosDAO.java que contenga los métodos necesarios para realizar operaciones contra la tabla propietarios.
            - Insertar un nuevo propietario: Recibe por parámetro los datos de un propietario a insertar, trata de insertarlo en la BBDD y devuelve 0 si la operación se realizó con éxito o -1 en caso contrario.
            - Recuperar vehículos de un propietario: Recibe por parámetro el DNI de un propietario y devuelve una lista con la matrícula, marca, número de kms y precio de todo sus vehículos. Retornará null si hubo problemas.
            - Eliminar propietario: Recibe por parámetro el DNI de un propietario y trata de eliminarlo de la BBDD. Devuelve el número de registros eliminados.
        - Crea una clase estática denominada VehiculosDAO.java que contenga los métodos necesarios para realizar operaciones contra la tabla Vehiculos. Deberá contener métodos para:
            - Insertar un nuevo vehículo: Recibe por parámetro los datos del vehículo a insertar, trata de insertarlo en la BBDD y devuelve 0 si

la operación se realizó con éxito o -1 en caso contrario.

- Actualizar propietario de vehículo: Recibe por parámetro la matrícula de un vehículo junto al identificador de un propietario y trata de actualizar el vehículo en la BBDD. Devuelve 0 si la operación se realizó con éxito o -1 si el vehículo no existe.
- Eliminar vehículo: Recibe por parámetro la matrícula de un vehículo y trata de eliminarlo de la BBDD. Devuelve 0 si la operación se realizó con éxito o -1 si el vehículo no existe.
- Recuperar todos los vehículos: No recibe parámetros y devuelve una lista con todos los vehículos del concesionario.Para cada vehículo, la lista contendrá una cadena de caracteres con los datos del mismo, incluido el nombre del propietario
- Recuperar vehículos por marca: Recibe una marca por parámetro y devuelve una lista con el listado de vehículos de la citada marca. Para cada vehículo, la lista contendrá una cadena de caracteres con los datos del mismo, incluido el nombre del propietario. Si no existen vehículos, devuelve el ArrayList vacío.
- Recuperar vehículos: No recibe parámetros (solo la conexión con la BBDD) y retorna una lista con la matrícula, marca, kilómetros y precio de cada vehículo.

Para los métodos que devuelven una lista, ten en cuenta que los elementos de la misma serán cadenas de caracteres.

- Crea otro paquete, denominado com.prog11.main que contendrá la clase Principal de la aplicación.
  - ❖ Dentro este paquete, crea una clase denominada Prog11_Main.java que contenga el método main. Inserta el código necesario en esta clase para poder probar cada uno de los métodos implementados en las clases de acceso a la BBDD, siguiendo el siguiente orden
  - Insertar varios vehículos y propietarios.

- Listar todos los vehículos.
- Actualizar propietario de un vehículo.
- Listar todos los vehículos.
- Eliminar un vehículo que exista.
- Eliminar un vehículo que no exista.
- Listar todos los vehículos.
- Listar los vehículos de una marca.
- Listar todos los vehículos de un propietario.
- Eliminar un propietario con vehículos.
- Eliminar un propietario sin vehículos.

**IMPORTANTE:**

- La conexión se creará en la clase principal y se pasará a los métodos que la necesiten como parámetro.
- Los métodos de las clases DAO no devuelven objetos de tipo ResultSet, deben devolver un arraylist de Strings.
- Las excepciones de los clases que interactúan con la BBDD deben tratarse dentro de la clase.
- **Entregar el script completo .sql con la creación de la BD, tablas, inserciones y reglas.**
- **El proyecto deberá incluir el conector/driver utilizado.**

**Criterios de puntuación.**
**Total 10 puntos RA8**
**Total 10 puntos RA9**

- Instalación y creación de bases de datos con credenciales y tablas. 4 puntos (RA8)
- Clase Principal:  2 puntos (RA8)
- Clase ConnectionDB:  4 puntos (RA8)
- Clase VehículosDAO: 5 puntos.(RA9)
- Clase PropietariosDAO: 5 puntos.(RA9)

Total: 10 puntos por cada RA trabajado.

- Todos los archivos .java deben estar correctamente comentados y estructurados. Se penalizará con hasta 2 puntos en cada RA por este motivo.

**Indicaciones de entrega**

Cada ejercicio estará contenido en un fichero cuyo nombre sea similar a **PROG11_tarea**. El tipo de archivo a entregar dependerá del ejercicio:
Se deberá entregar una carpeta comprimida con el **proyecto completo** donde se incluyan todos los archivos fuente demandados en el enunciado además del documento de texto.

**Es obligatorio adjuntar el documento explicativo, el script completo .sql y el driver/conector. De no hacerlo, la tarea será evaluada con un 0.**

**Asegúrate de que los archivos fuente (.java), contengan, como comentarios Java, tu nombre y apellidos y el número de ejercicio.**
**Los archivos fuente deben compilar y ejecutarse siguiendo las instrucciones de su enunciado. De no hacerlo, no se considerarán válidos y serán evaluados con un 0.**

En el supuesto que tengas/quieras adjuntar un documento de texto con explicaciones adicionales, o con la solución a alguno de los puntos pedidos, debes de seguir las siguientes consideraciones que se explican en el siguiente documento.
Crea una carpeta para la entrega y adjunta los archivos creados, y renombra la carpeta siguiendo las siguientes pautas:

*apellido1_apellido2_nombre_PROG_Tarea11*

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños.

# IMPORTANTE

Si la carpeta es muy pesada para el límite máximo de subida de la plataforma Moodle, súbela a tu unidad del Drive. Utiliza la cuenta de Educación, y en la entrega en Moodle adjunta la URL de tu entrega.

Para que la entrega se considere realizada en tiempo y forma, sube al Moodle el PDF con la solución de tu tarea, y adjunta el resto de los archivos que has creado al Drive, y comparte la carpeta mediante su enlace correspondiente. **No olvides ajustar los privilegios de la carpeta en Drive para que pueda acceder a tu trabajo.**

**Todos los ejercicios y/o /documentos demandados que no cumplan exactamente con las instrucciones y su formato de entrega no se considerarán válidos y serán evaluados con un 0.**