

Acceso a datos

Tarea Unidad 5

Contenido

Estructura de la aplicación	2
Localización de ficheros	6
EJERCICIO 1.	9
EJERCICIO 2.	13
EJERCICIO 3.	15
Aclaraciones y Explicaciones Generales de la tarea	22

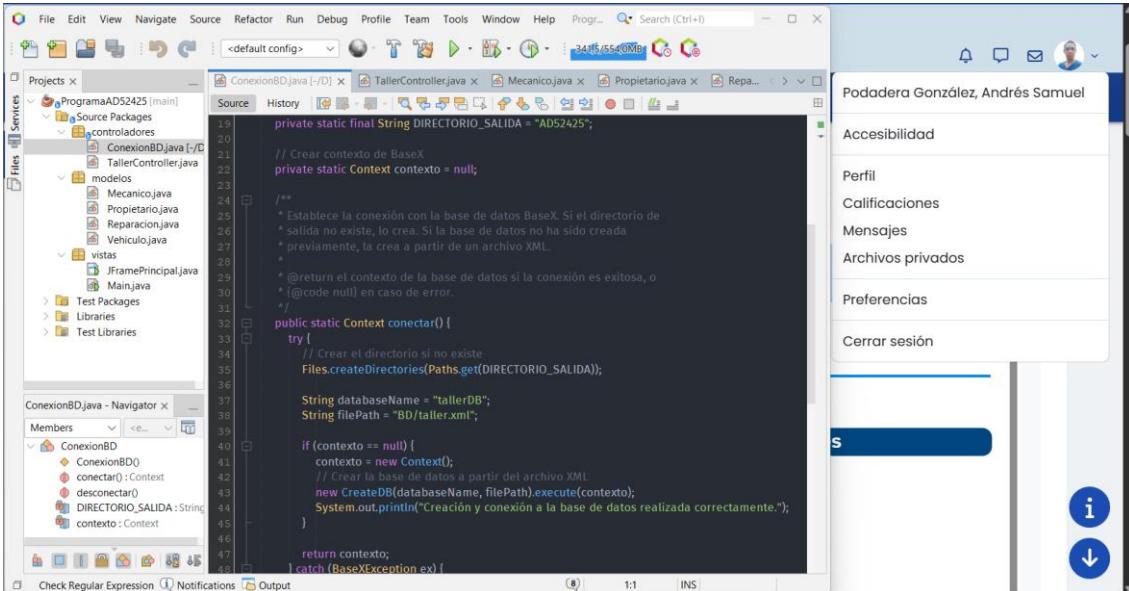
Estructura de la aplicación

Lo primero será realizar una pequeña revisión de la estructura del proyecto, como está organizado y que se hace en cada parte.

La estructura de aplicación sigue el patrón MVC en la que el paquete Modelos contiene las entidades de la base de datos, el paquete de controladores contiene nuestro controlador de taller y la clase ConexionBD para conectar a la base de datos y el paquete vistas contiene nuestro formulario principal y la clase Main, el punto de arranque de nuestra aplicación. En un desglose más específico enumero cada una de ellas:

ConexionBD.java

Esta clase es la encargada de localizar el archivo XML y crear la base de datos en memoria a partir de los datos del archivo. También se encarga de cerrar la conexión cuando la aplicación termina.



```
private static final String DIRECTORIO_SALIDA = "AD52425";
// Crear contexto de BaseX
private static Context contexto = null;

/**
 * Establece la conexión con la base de datos BaseX. Si el directorio de
 * salida no existe, lo crea. Si la base de datos no ha sido creada
 * previamente, la crea a partir de un archivo XML.
 */
public static Context conectar() {
    try {
        // Crear el directorio si no existe
        Files.createDirectories(Paths.get(DIRECTORIO_SALIDA));

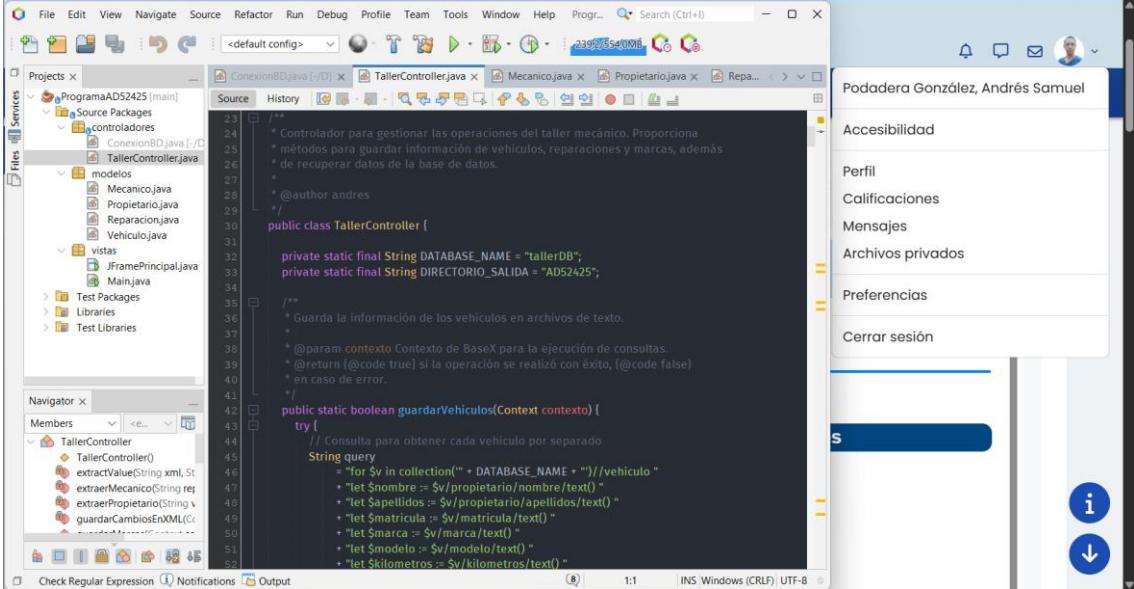
        String databaseName = "tallerDB";
        String filePath = "BD/taller.xml";

        if (contexto == null) {
            contexto = new Context();
            // Crear la base de datos a partir del archivo XML.
            new CreateDB(databaseName, filePath).execute(contexto);
            System.out.println("Creación y conexión a la base de datos realizada correctamente.");
        }
        return contexto;
    } catch (BaseXException ex) {
        throw new RuntimeException(ex);
    }
}
```

TallerController.java

Esta clase contiene toda la lógica de nuestra aplicación. Contiene métodos para acceder a la información, insertarla, modificarla o eliminarla. Todas las operaciones se realizan mediante la ejecución de sentencias XQuery sobre la base de datos en memoria.

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025



The screenshot shows the Eclipse IDE interface with the TallerController.java file open in the central editor window. The code implements a controller for managing car operations in a garage. It includes methods for saving vehicle, repair, and owner information, as well as retrieving data from the database. The code is annotated with Javadoc comments and includes a constructor and several instance variables.

```
/*
 * Controlador para gestionar las operaciones del taller mecánico. Proporciona
 * métodos para guardar información de vehículos, reparaciones y marcas, además
 * de recuperar datos de la base de datos.
 *
 * @author andres
 */
public class TallerController {

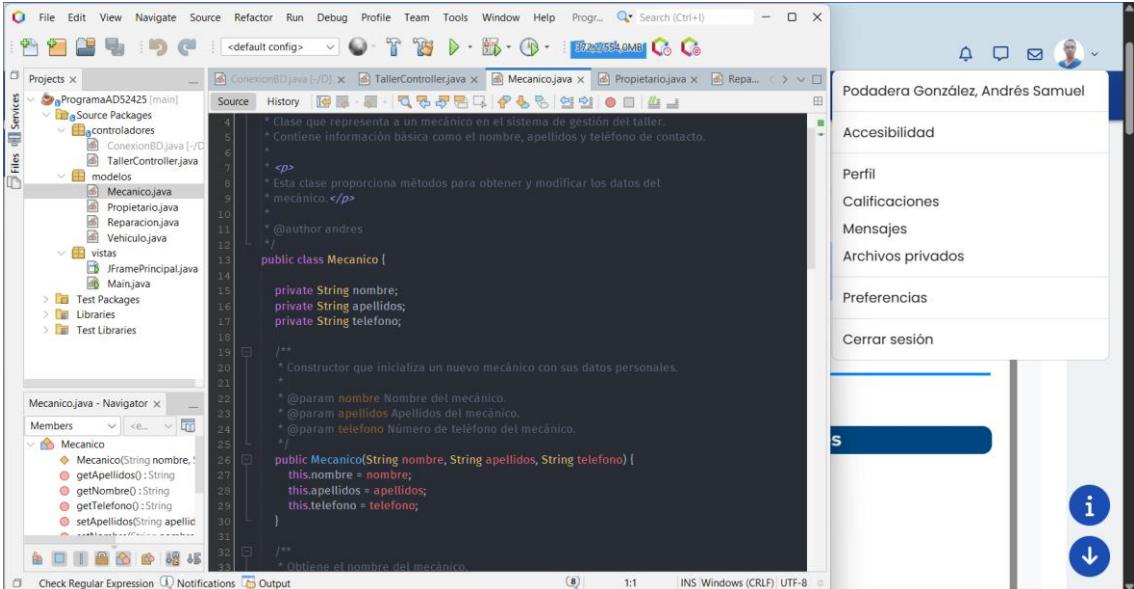
    private static final String DATABASE_NAME = "tallerDB";
    private static final String DIRECTORIO_SALIDA = "ADS2425";

    /**
     * Guarda la información de los vehículos en archivos de texto.
     *
     * @param contexto Contexto de BaseX para la ejecución de consultas.
     * @return {@code true} si la operación se realizó con éxito, {@code false}
     * en caso de error.
     */
    public static boolean guardarVehiculos(Context contexto) {
        try {
            // Consulta para obtener cada vehículo por separado
            String query =
                "for $v in collection(\"" + DATABASE_NAME + "\")/vehiculo "
                + "let $nombre := $v/propietario/nombre/text() "
                + "let $apellidos := $v/propietario/apellidos/text() "
                + "let $matricula := $v/matricula/text() "
                + "let $marca := $v/marca/text() "
                + "let $modelo := $v/modelo/text() "
                + "let $kilometros := $v/kilometros/text() "

```

Mecánico.java

Es la representación en código Java de nuestra entidad Mecánico de la base de datos, contiene los atributos y métodos para almacenar y gestionar su información.



The screenshot shows the Eclipse IDE interface with the Mecanico.java file open in the central editor window. This class represents a mechanic in the system. It has attributes for name, surnames, and phone number, and methods for initializing a new mechanic and getting their name. The code includes Javadoc comments and a constructor.

```
/*
 * Clase que representa a un mecánico en el sistema de gestión del taller.
 * Contiene información básica como el nombre, apellidos y teléfono de contacto.
 */


<p>



Esta clase proporciona métodos para obtener y modificar los datos del
mecánico.</p>
*/
@author andres
*/
public class Mecanico {

    private String nombre;
    private String apellidos;
    private String telefono;

    /**
     * Constructor que inicializa un nuevo mecánico con sus datos personales.
     *
     * @param nombre Nombre del mecánico.
     * @param apellidos Apellidos del mecánico.
     * @param telefono Número de teléfono del mecánico.
     */
    public Mecanico(String nombre, String apellidos, String telefono) {
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.telefono = telefono;
    }

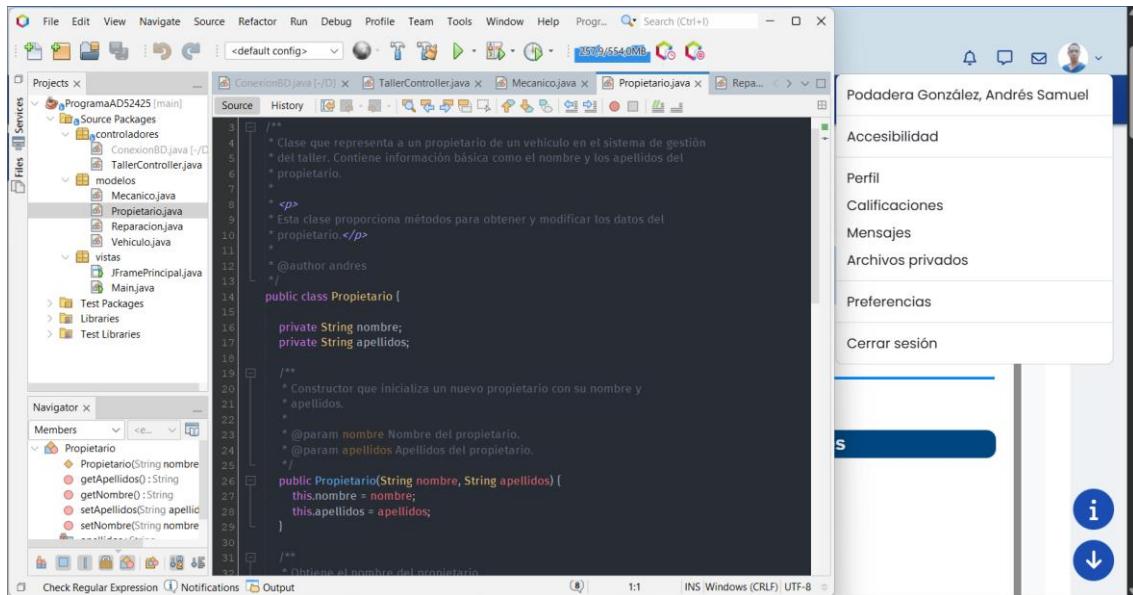
    /**
     * Obtiene el nombre del mecánico.
     */


```

Propietario.java

Es la representación en código Java de nuestra entidad Propietario de la base de datos, contiene los atributos y métodos para almacenar y gestionar su información.

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025



```
/*
 * Clase que representa a un propietario de un vehículo en el sistema de gestión
 * del taller. Contiene información básica como el nombre y los apellidos del
 * propietario.
 *
 * <p>
 * Esta clase proporciona métodos para obtener y modificar los datos del
 * propietario.</p>
 *
 * @author andres
 */
public class Propietario {

    private String nombre;
    private String apellidos;

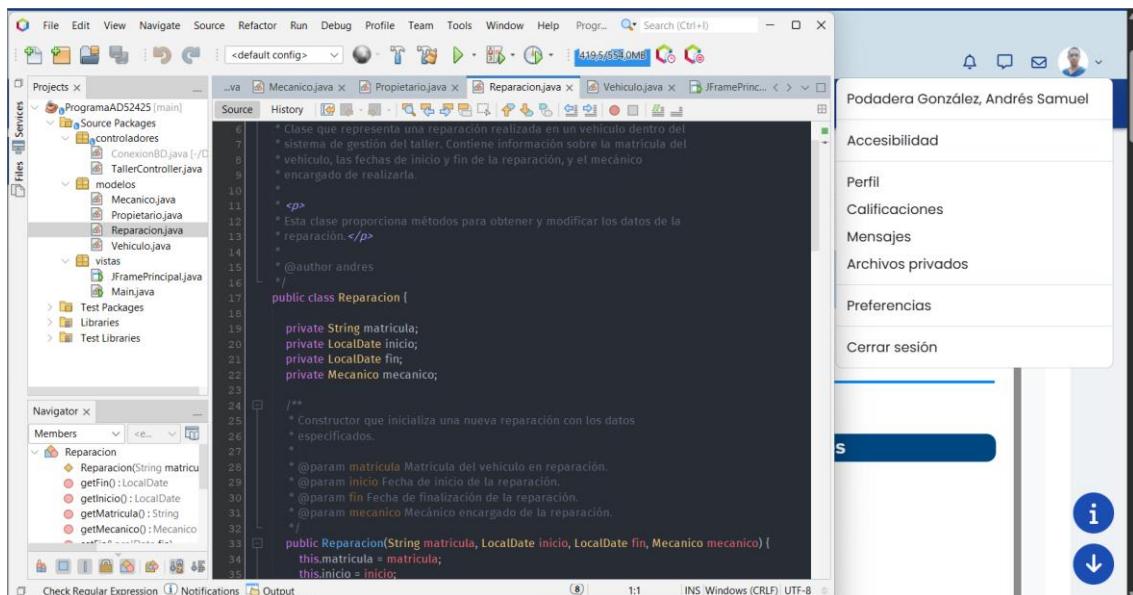
    /**
     * Constructor que inicializa un nuevo propietario con su nombre y
     * apellidos.
     *
     * @param nombre Nombre del propietario.
     * @param apellidos Apellidos del propietario.
     */
    public Propietario(String nombre, String apellidos) {
        this.nombre = nombre;
        this.apellidos = apellidos;
    }

    /**
     * Obtiene el nombre del propietario.
     *
     * @return El nombre del propietario.
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Establece el nombre del propietario.
     *
     * @param nombre El nombre del propietario.
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

Reparacion.java

Es la representación en código Java de nuestra entidad Reparación de la base de datos, contiene los atributos y métodos para almacenar y gestionar su información.



```
/*
 * Clase que representa una reparación realizada en un vehículo dentro del
 * sistema de gestión del taller. Contiene información sobre la matrícula del
 * vehículo, las fechas de inicio y fin de la reparación, y el mecánico
 * encargado de realizarla.
 *
 * <p>
 * Esta clase proporciona métodos para obtener y modificar los datos de la
 * reparación.</p>
 *
 * @author andres
 */
public class Reparacion {

    private String matricula;
    private LocalDate inicio;
    private LocalDate fin;
    private Mecanico mechanico;

    /**
     * Constructor que inicializa una nueva reparación con los datos
     * especificados.
     *
     * @param matricula Matrícula del vehículo en reparación.
     * @param inicio Fecha de inicio de la reparación.
     * @param fin Fecha de finalización de la reparación.
     * @param mechanico Mecánico encargado de la reparación.
     */
    public Reparacion(String matricula, LocalDate inicio, LocalDate fin, Mecanico mechanico) {
        this.matricula = matricula;
        this.inicio = inicio;
        this.fin = fin;
        this.mechanico = mechanico;
    }

    /**
     * Obtiene la matrícula del vehículo en reparación.
     *
     * @return La matrícula del vehículo.
     */
    public String getMatricula() {
        return matricula;
    }

    /**
     * Establece la matrícula del vehículo en reparación.
     *
     * @param matricula La matrícula del vehículo.
     */
    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    /**
     * Obtiene la fecha de inicio de la reparación.
     *
     * @return La fecha de inicio de la reparación.
     */
    public LocalDate getInicio() {
        return inicio;
    }

    /**
     * Establece la fecha de inicio de la reparación.
     *
     * @param inicio La fecha de inicio de la reparación.
     */
    public void setInicio(LocalDate inicio) {
        this.inicio = inicio;
    }

    /**
     * Obtiene la fecha de finalización de la reparación.
     *
     * @return La fecha de finalización de la reparación.
     */
    public LocalDate getFin() {
        return fin;
    }

    /**
     * Establece la fecha de finalización de la reparación.
     *
     * @param fin La fecha de finalización de la reparación.
     */
    public void setFin(LocalDate fin) {
        this.fin = fin;
    }

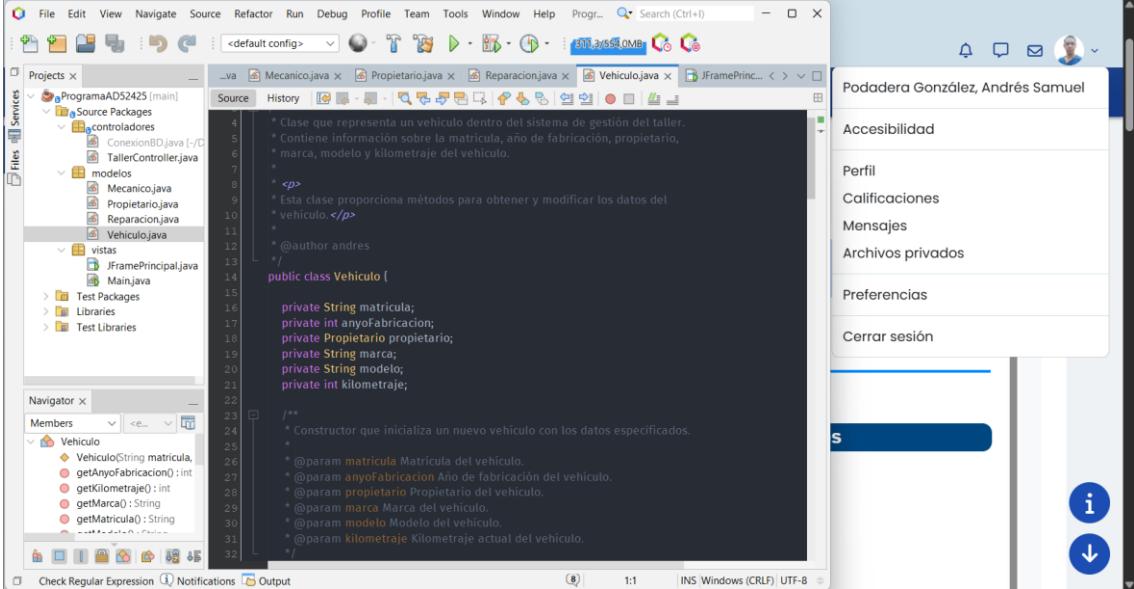
    /**
     * Obtiene el mecanico encargado de la reparación.
     *
     * @return El mecanico encargado de la reparación.
     */
    public Mecanico getMechanico() {
        return mechanico;
    }

    /**
     * Establece el mecanico encargado de la reparación.
     *
     * @param mechanico El mecanico encargado de la reparación.
     */
    public void setMechanico(Mecanico mechanico) {
        this.mechanico = mechanico;
    }
}
```

Vehículo.java

Es la representación en código Java de nuestra entidad Vehiculo de la base de datos, contiene los atributos y métodos para almacenar y gestionar su información.

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025

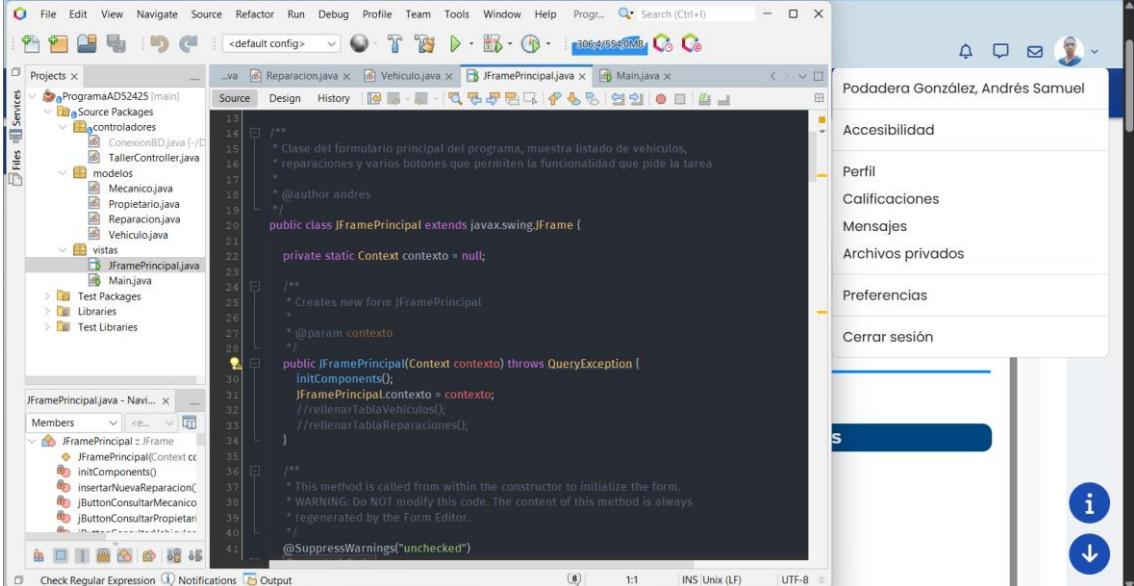


The screenshot shows the IntelliJ IDEA interface with the Vehiculo.java file open in the editor. The code defines a class Vehiculo with attributes like matrícula, año de fabricación, propietario, marca, modelo, and kilometraje. It includes a constructor and several methods. The Navigator tool window on the left shows the class structure. A sidebar on the right displays user information and navigation links.

```
4 * Clase que representa un vehículo dentro del sistema de gestión del taller.  
5 * Contiene información sobre la matrícula, año de fabricación, propietario,  
6 * marca, modelo y kilometraje del vehículo.  
7 *  
8 * <p>  
9 * Esta clase proporciona métodos para obtener y modificar los datos del  
10 * vehículo.</p>  
11 *  
12 * @author andres  
13 */  
  
public class Vehiculo {  
  
    private String matricula;  
    private int anyoFabricacion;  
    private Propietario propietario;  
    private String marca;  
    private String modelo;  
    private int kilometraje;  
  
    /**  
     * Constructor que inicializa un nuevo vehículo con los datos especificados.  
     *  
     * @param matricula Matrícula del vehículo.  
     * @param anyoFabricacion Año de fabricación del vehículo.  
     * @param propietario Propietario del vehículo.  
     * @param marca Marca del vehículo.  
     * @param modelo Modelo del vehículo.  
     * @param kilometraje Kilometraje actual del vehículo.  
     */  
  
}
```

JFramePrincipal.java

Esta clase contiene el único formulario de nuestra aplicación. Este JFrame hecho en Swing nos va a permitir realizar todas las operaciones que nos piden en los ejercicios de la tarea.



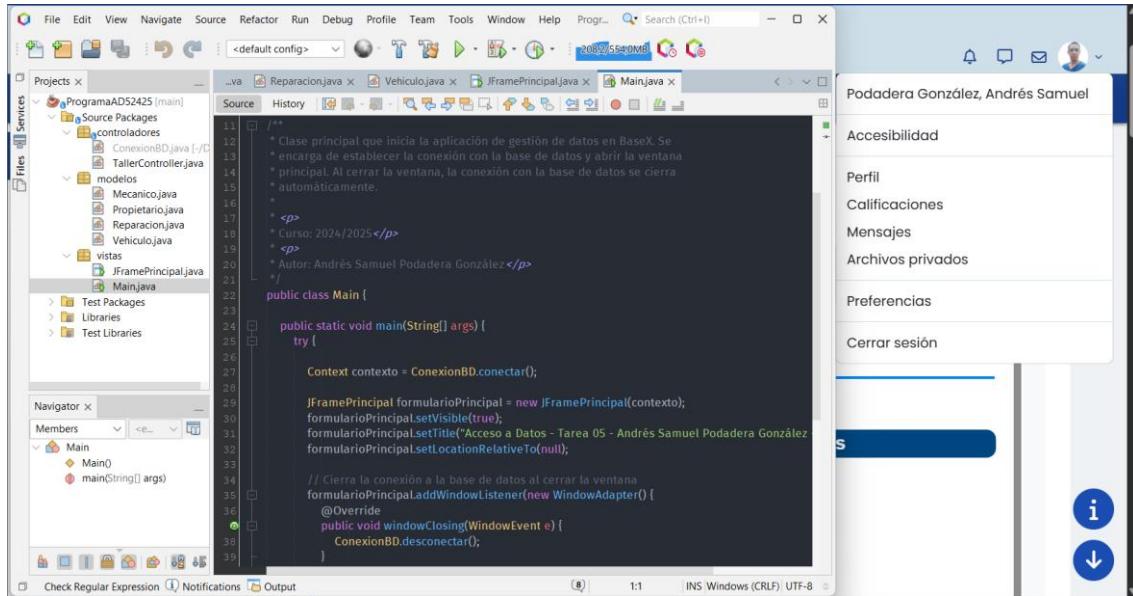
The screenshot shows the IntelliJ IDEA interface with the JFramePrincipal.java file open in the editor. This is a Swing application window with various components like buttons and tables. The code includes annotations such as @author andres and @SuppressWarnings("unchecked"). A sidebar on the right shows user information and navigation links.

```
13 /**  
14  * Clase del formulario principal del programa, muestra listado de vehículos,  
15  * reparaciones y varios botones que permiten la funcionalidad que pide la tarea  
16  *  
17  * @author andres  
18 */  
  
public class JFramePrincipal extends javax.swing.JFrame {  
  
    private static Context contexto = null;  
  
    /**  
     * Creates new form JFramePrincipal  
     *  
     * @param contexto  
     */  
    public JFramePrincipal(Context contexto) throws SQLException {  
        initComponents();  
        JFramePrincipal.contexto = contexto;  
        //rellenarTablaVehiculos();  
        //rellenarTablaReparaciones();  
    }  
  
    /**  
     * This method is called from within the constructor to initialize the form.  
     * WARNING: Do NOT modify this code. The content of this method is always  
     * regenerated by the Form Editor.  
     */  
    @SuppressWarnings("unchecked")
```

Main.java

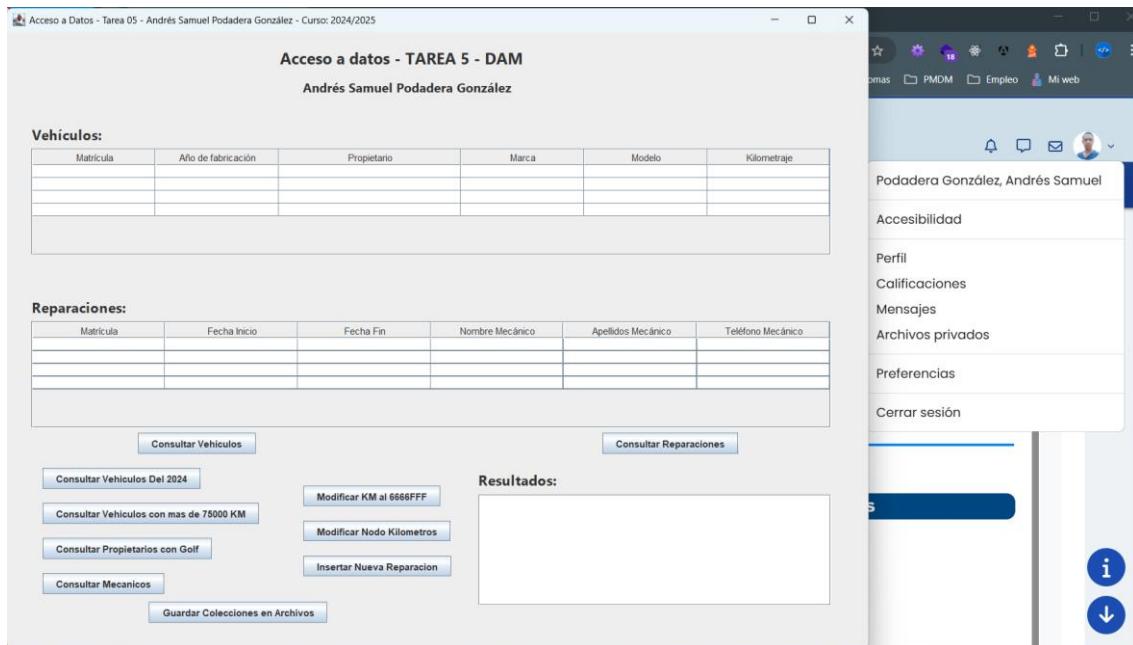
Es el punto de arranque de la aplicación, nos permite crear y lanzar el JFrame con un título y tamaño específicos.

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025



Aplicación en funcionamiento

Nuestra aplicación una vez arrancada se presenta de esta forma:



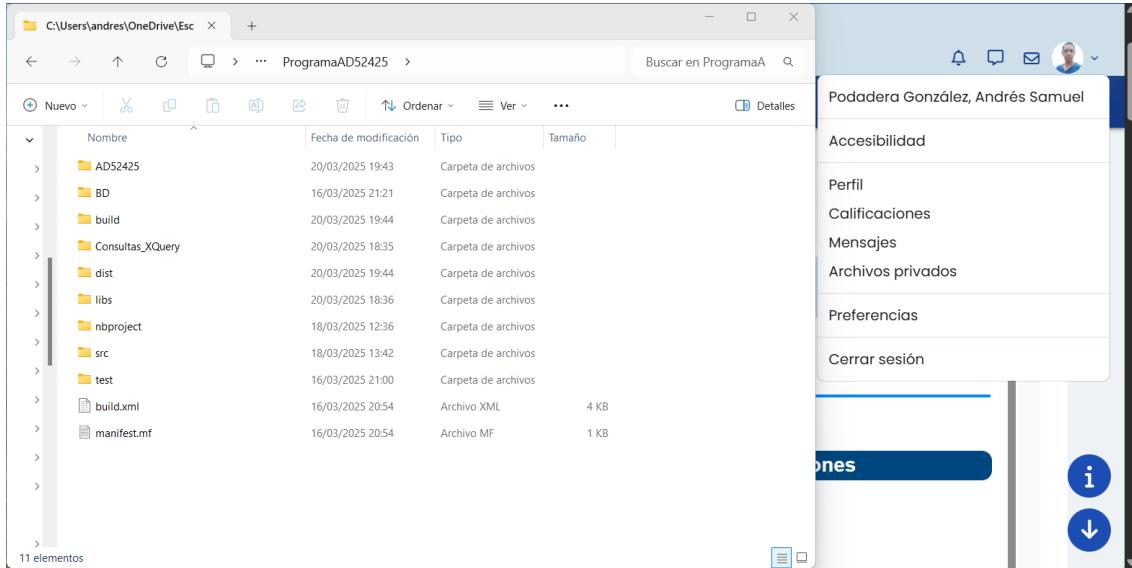
En la tabla Vehículos, listamos los vehículos, en la tabla Reparaciones listamos las reparaciones registradas, el conjunto de botones nos permiten realizar las consultas de vehículos y reparaciones, además de realizar las operaciones que nos solicitan en la tarea como insertar nuevos nodos, modificarlos o eliminarlos. También nos permiten guardar la información del XML en archivos de texto separados.

Localización de ficheros

Dentro del nuestro proyecto los ficheros más importantes se encuentran dispuestos en:

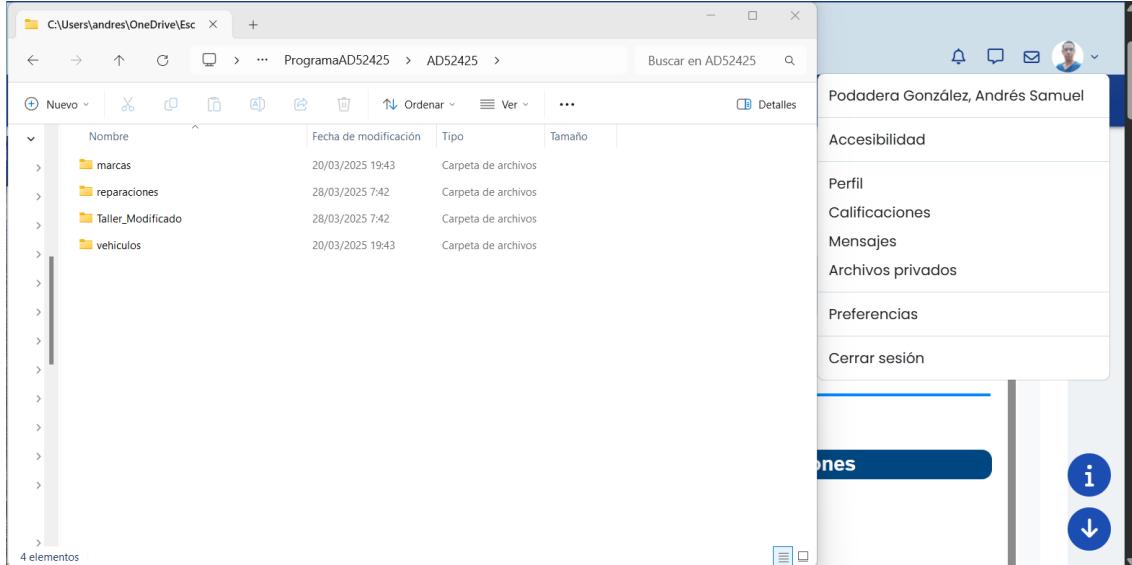
Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025

Estructura principal:



Carpeta AD52425:

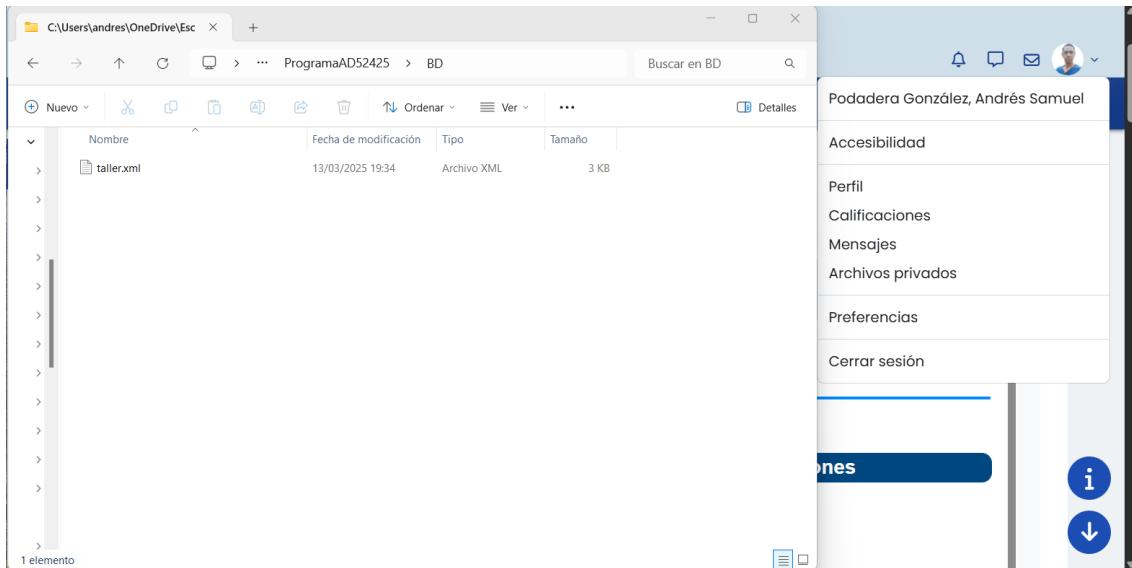
Contiene las carpetas y archivos que se generan en el Ejercicio 1, en el que extraemos del XML, las marcas, reparaciones y vehículos. Taller_modificado contendrá el archivo XML modificado cuando en el último ejercicio nos pidan modificar los nodos y almacenar los cambios (he preferido no modificar el archivo original)



Carpeta BD:

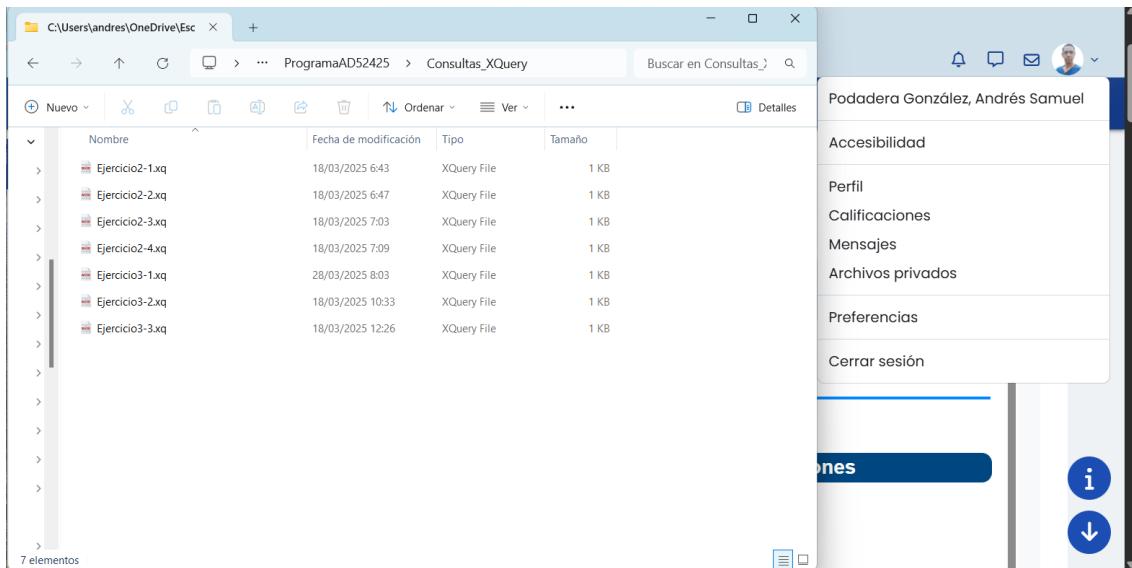
Contiene el archivo XML con la información que tratamos en esta tarea.

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025



Carpeta Consultas_XQuery:

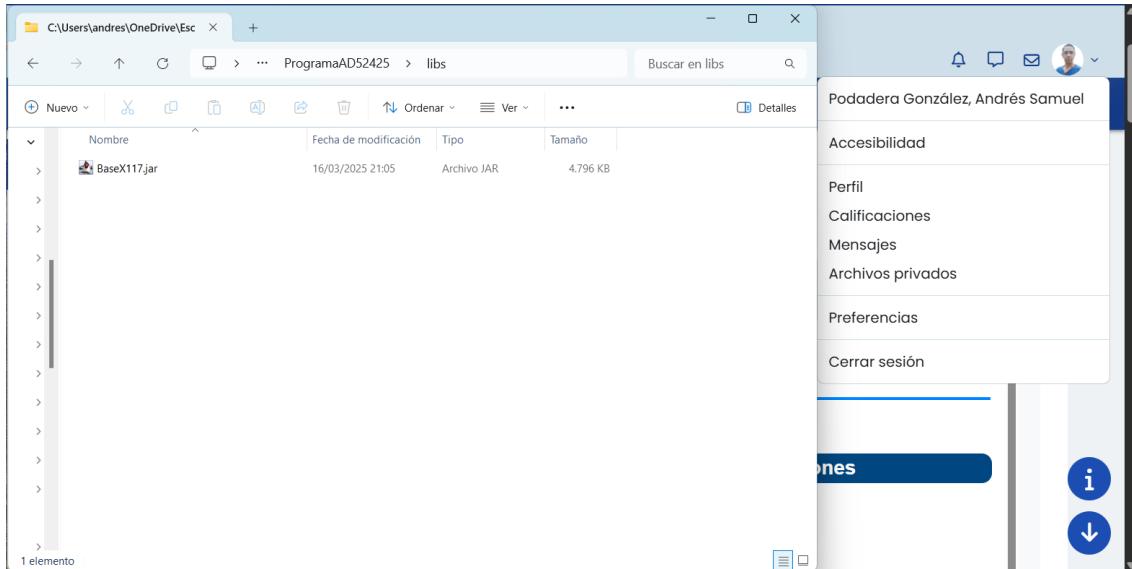
Contiene todas las consultas que se piden en los diferentes ejercicios de esta tarea, construidas y probadas en BaseX. La mayoría de ellas contienen comentarios de aclaración y explicación de las operaciones que realizan.



Carpeta libs:

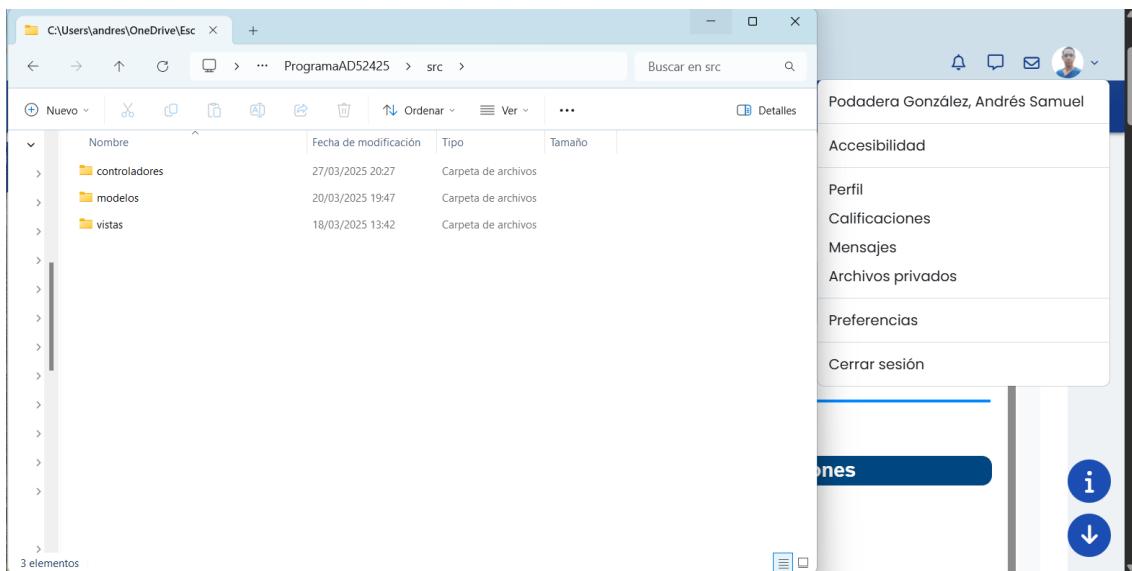
Contiene el jar de la librería BaseX, necesario para realizar la tarea.

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025



Carpeta src:

Contiene los paquetes o carpetas con el código fuente escrito en java de nuestra aplicación.



EJERCICIO 1.

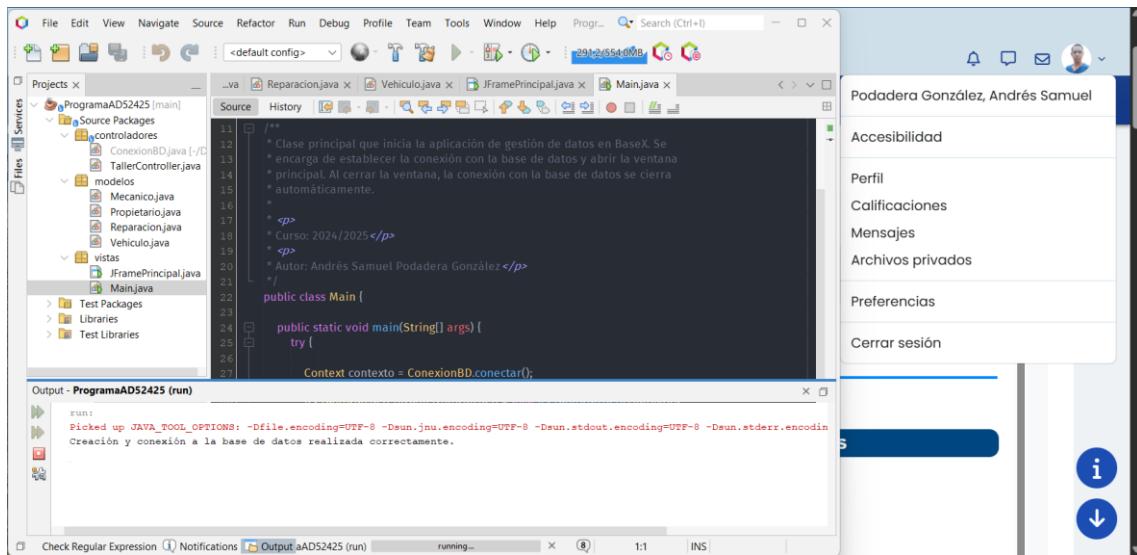
Es necesario crear una aplicación con interfaz gráfica (usando Swing) en base al driver Basex.jar donde se realicen operaciones de lectura, inserción, modificación y borrado.

Utilizando la base de datos [taller.xml](#), extrae las colecciones 'Vehículos', 'Reparaciones' y 'Marcas' para realizar una copia local en el directorio de tu proyecto (AD52425) de manera que:

- Guarda cada vehículo, reparación o marca en un fichero de nombre vehiculox, reparacionx, marcas (x será un número secuencial), dentro del directorio de tu proyecto (si no existe el directorio, obviamente ha de crearse).

Para comprobar que la aplicación realiza la operación que se pide comenzamos arrancando la aplicación y comprobando que dentro del proyecto no existe la carpeta AD222425 ya que es lo que se pide en este apartado, que la aplicación sea capaz de crear carpetas y ficheros con la información del archivo XML.

Empezaremos arrancando la aplicación y haciendo clic en los botones de “Consultar Vehículos” y “Consultar Reparaciones”. Cuando nuestra aplicación arranca, busca el archivo taller.xml en el directorio correspondiente, crea la base de datos en memoria y queda a la espera que se realicen acciones desde la interfaz. Si todo ha salido como se espera deberíamos ver un mensaje de creación y conexión a la base de datos en la consola de NetBeans y la interfaz debería mostrarnos los datos consultados:



The screenshot shows the application's graphical user interface. It includes sections for "Vehículos:" and "Reparaciones:" containing tables of data. Below these are several buttons for querying the database. A central "Resultados:" area is shown with a large empty box. Two red arrows point from the "Consultar Vehículos" and "Consultar Reparaciones" buttons towards the "Pulsa aquí" button in the center, indicating they all lead to the same results area.

Matrícula	Año de fabricación	Propietario	Marca	Modelo	Kilometraje
1111BBB	2024	Juan Martínez López	Peugeot	308	20000
2222BBB	2024	Ana María García	Citroën	C5	54000
3333CCC	2023	José Luis Medina	Vauxhall	XR	70000
4444CCC	2023	Julio Martín Martín	Renault	Megane	85000
5555DDO	2022	Alicia Puertas Gómez	Volkswagen	Golf	100000
6666FFF	2020	Eva Pío Ponce	BMW	X3	90000
7777FFF	2021	Eduardo Cuevas Torres	Audi	Q5	150000

Matrícula	Fecha Inicio	Fecha Fin	Nombre Mecánico	Apellidos Mecánico	Teléfono Mecánico
2222BBB	2024-11-02	2024-12-10	Pepe	Grillo	972987654
2222BBB	2025-01-01	Sin Terminar	Lucas	García	950456987
3333CCC	2024-12-20	2025-01-20	Emilio	Trujillo	958741632
4444CCC	2024-02-20	Sin Terminar	Lucas	García	950456987

Pulsa aquí

Consultar Vehículos Consultar Reparaciones

Resultados:

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025

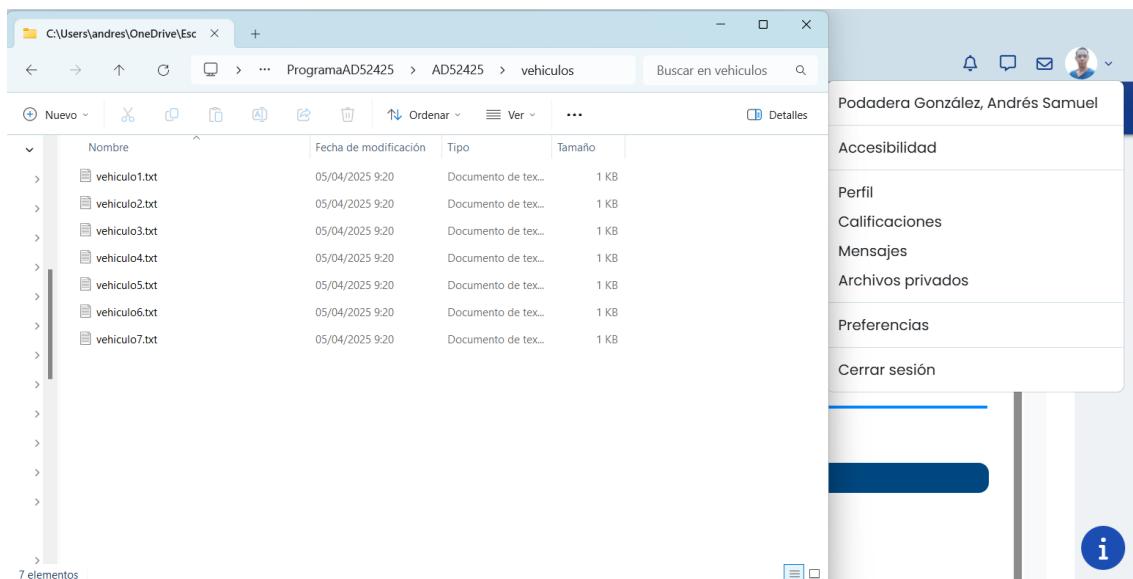
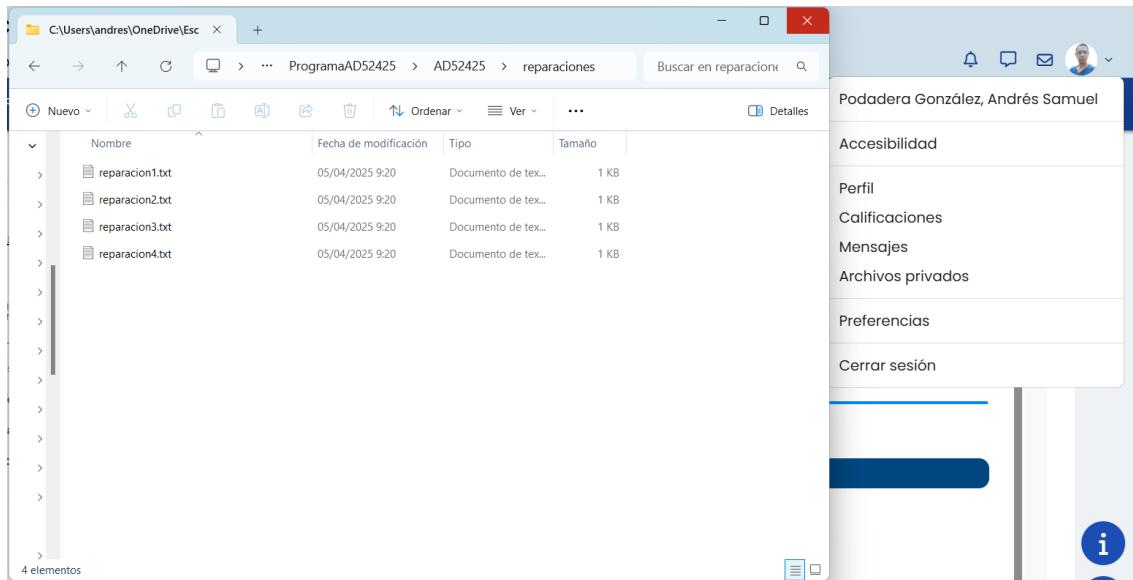
En este punto nuestra aplicación está conectada y vemos que es capaz de acceder y recuperar la información. Para realizar este ejercicio tan sólo tendremos que pulsar sobre el botón con el texto “Guardar Colecciones en Archivos”, aquí el programa creará el directorio AD52425 sino no existe, las carpetas de marcas, reparaciones y vehículos con los respectivos ficheros txt que contiene la información clasificada.

The screenshot shows the application's main window titled "Acceso a datos - TAREA 5 - DAM" with the subtitle "Andrés Samuel Podadera González". It displays two tables: "Vehículos" and "Reparaciones". Below the tables are several search buttons. At the bottom left is a button labeled "Guardar Colecciones en Archivos". To the right is a sidebar menu with items like "Perfil", "Calificaciones", and "Archivos privados". A red arrow points to the "Guardar Colecciones en Archivos" button.

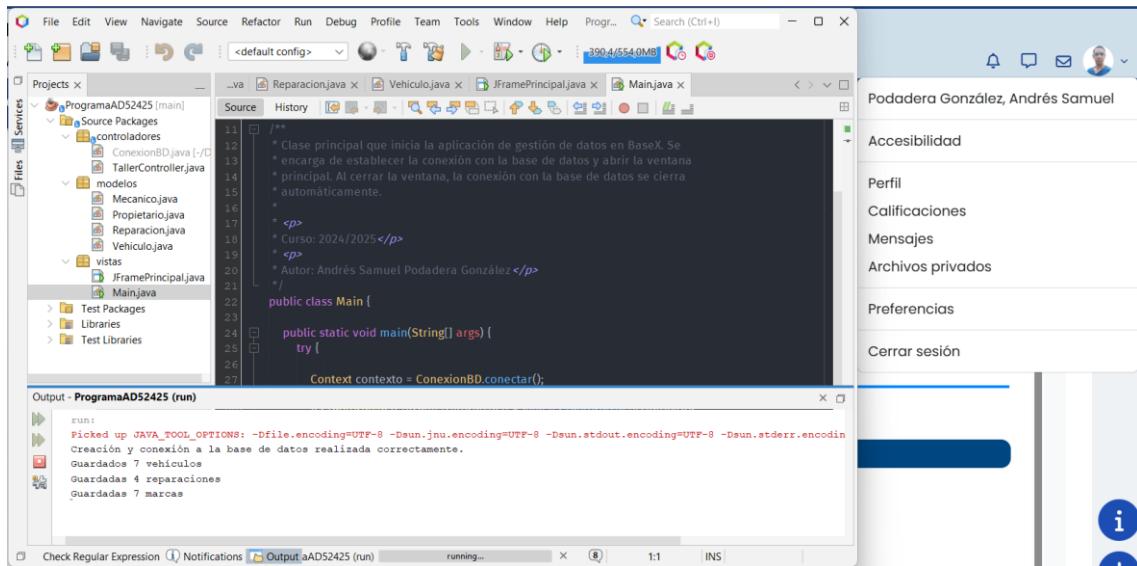
Y se generan los archivos:

The screenshot shows a file explorer window displaying a folder structure. The path is "C:\Users\andres\OneDrive\Escritorio\ProgramaAD52425\AD52425\marcas". Inside this folder, there are seven files named "marca1.txt" through "marca7.txt", each with a size of 1 KB. To the right of the file explorer is a sidebar menu with options like "Perfil", "Calificaciones", and "Archivos privados".

Nombre: Andrés Samuel Podadera González
Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
Curso: 2024/2025



Además, en la consola de NetBeans debe aparecer un mensaje indicando la cantidad de vehículos, reparaciones y marcas que se han almacenado en los archivos.



EJERCICIO 2.

Realiza estas consultas a la BD taller.xml y ejecútalas desde la aplicación Java. Guardar cada consulta en un script .xq, que llamarás desde tu programa:

- a. Matrícula de los vehículos cuyo año de fabricación (afabricación) sea el 2024

Para realizar esta actividad pulsaremos en el botón:

Matrícula	Año de fabricación	Propietario	Marca	Modelo	Kilometraje
1111BBB	2024	Juan Martinez Lopez	Peugeot	308	20000
2222BBB	2024	Ana Mora Garcia	Citroen	C5	54000
3333CCC	2023	Jesse Luna Molina	Lexus	NX	70000
4444CCC	2023	Julio Marin Marin	Renault	Megane	85000
5555DDD	2022	Alicia Puentes Gomez	Volkswagen	Golf	100000
6666FFF	2020	Eva Pio Ponce	BMW	X3	80000
7777FFF	2021	Eduardo Cuevas Torres	Audi	Q5	150000

Reparaciones:					
Matrícula	Fecha Inicio	Fecha Fin	Nombre Mecánico	Apellidos Mecánico	Teléfono Mecánico
2222BBB	2024-11-02	2024-12-10	Pepito	Grillo	972987654
2222BBB	2025-01-01	Sei Terminar	Lucas	Garcia	956789012
3333CCC	2024-12-20	2025-01-20	Emilio	Trujillo	958741632
4444CCC	2024-02-20	Sei Terminar	Lucas	Garcia	950456907

Consultar Vehiculos
Consultar Reparaciones

Consultar Vehiculos Del 2024
Pulsar aquí

Consultar Vehiculos con mas de 75000 KM
Modificar KM al 6666FFF

Consultar Propietarios con Golf
Modificar Nodo Kilometros

Consultar Mecanicos
Insertar Nueva Reparacion

Guardar Colecciones en Archivos

Resultados:
 Matrícula: 1111BBB
 Matrícula: 2222BBB

Y en el cuadro de texto de la derecha, con título “Resultados”, nos aparecen las matrículas de los vehículos:

Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025

Vehículos:

Matrícula	Año de fabricación	Propietario	Marca	Modelo	Kilometraje
1111BBB	2024	Juan Martínez López	Peugeot	308	20000
2222BBB	2024	Ana Mora García	Citroën	C5	54000
3333CCC	2023	José Luna Molina	Lexus	NX	70000
4444CCC	2023	Julio Marín Martín	Renault	Megane	85000
5555DDD	2022	Alicia Puentes Gómez	Volkswagen	Golf	100000
6666FFF	2020	Eva Pío Ponce	BMW	X3	90000
7777FFF	2021	Eduardo Cuevas Torres	Audi	Q5	150000

Reparaciones:

Matrícula	Fecha Inicio	Fecha Fin	Nombre Mecánico	Apellidos Mecánico	Teléfono Mecánico
2222BBB	2024-01-01	2024-12-10	Pepito	Grillo	972987654
2222BBB	2025-01-01	Sin Terminar	Lucas	García	950456987
3333CCC	2024-12-20	2025-01-20	Emilio	Trujillo	958741632
4444CCC	2024-02-20	Sin Terminar	Lucas	García	950456987

Consultar Vehículos **Consultar Reparaciones**

Consultar Vehículos Del 2024 **Modificar KM al 6666FFF**
Consultar Vehículos con mas de 75000 KM **Modificar Nodo Kilometros**
Consultar Propietarios con Golf **Insertar Nueva Reparacion**
Consultar Mecánicos **Guardar Colecciones en Archivos**

Resultados:
 Matrícula: 1111BBB
 Matrícula: 2222BBB

b. Marca y modelo de los vehículos que tengan menos de 75000 kilómetros.

Para realizar este apartado pulsaremos sobre el botón con texto “Consultar Vehículos con mas de 75000KM” y en el cuadro de texto nos deberían aparecer los resultados:

Vehículos:

Matrícula	Año de fabricación	Propietario	Marca	Modelo	Kilometraje
1111BBB	2024	Juan Martínez López	Peugeot	308	20000
2222BBB	2024	Ana Mora García	Citroën	C5	54000
3333CCC	2023	José Luna Molina	Lexus	NX	70000
4444CCC	2023	Julio Marín Martín	Renault	Megane	85000
5555DDD	2022	Alicia Puentes Gómez	Volkswagen	Golf	100000
6666FFF	2020	Eva Pío Ponce	BMW	X3	90000
7777FFF	2021	Eduardo Cuevas Torres	Audi	Q5	150000

Reparaciones:

Matrícula	Fecha Inicio	Fecha Fin	Nombre Mecánico	Apellidos Mecánico	Teléfono Mecánico
2222BBB	2024-11-02	2024-12-10	Pepito	Grillo	972987654
2222BBB	2025-01-01	Sin Terminar	Lucas	García	950456987
3333CCC	2024-12-20	2025-01-20	Emilio	Trujillo	958741632
4444CCC	2024-02-20	Sin Terminar	Lucas	García	950456987

Consultar Vehículos **Consultar Reparaciones**

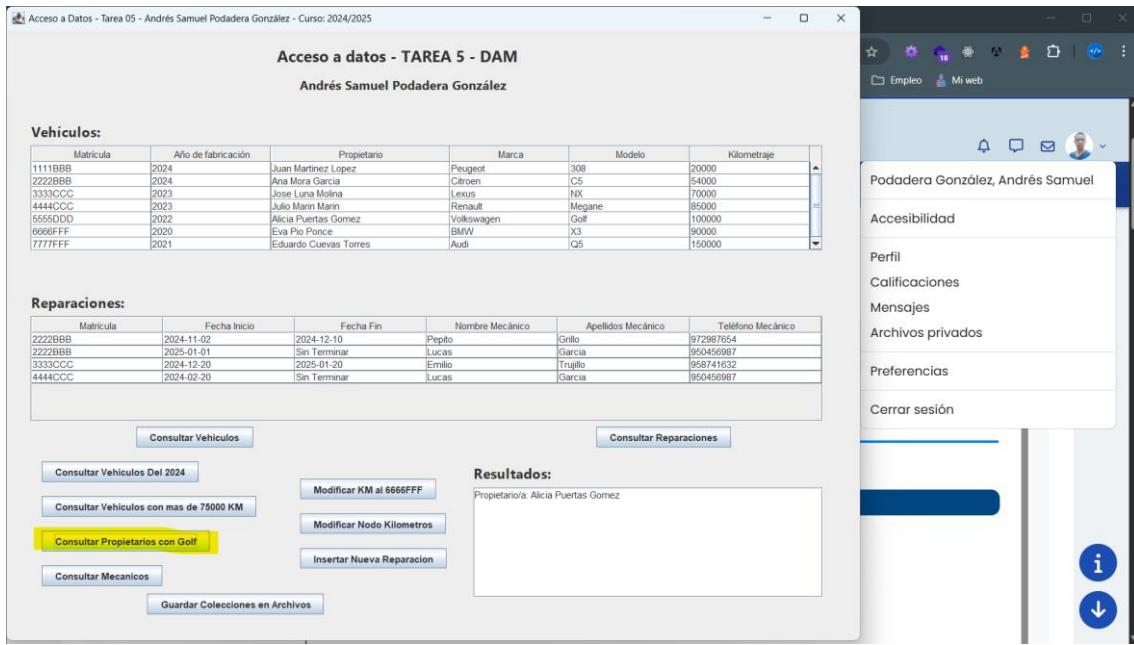
Consultar Vehículos Del 2024 **Modificar KM al 6666FFF**
Consultar Vehículos con mas de 75000 KM **Modificar Nodo Kilometros**
Consultar Propietarios con Golf **Insertar Nueva Reparacion**
Consultar Mecánicos **Guardar Colecciones en Archivos**

Resultados:
 Marca: Peugeot | Modelo: 308
 Marca: Citroën | Modelo: C5
 Marca: Lexus | Modelo: NX

c. Nombre y apellidos de los propietarios que tengan un Golf.

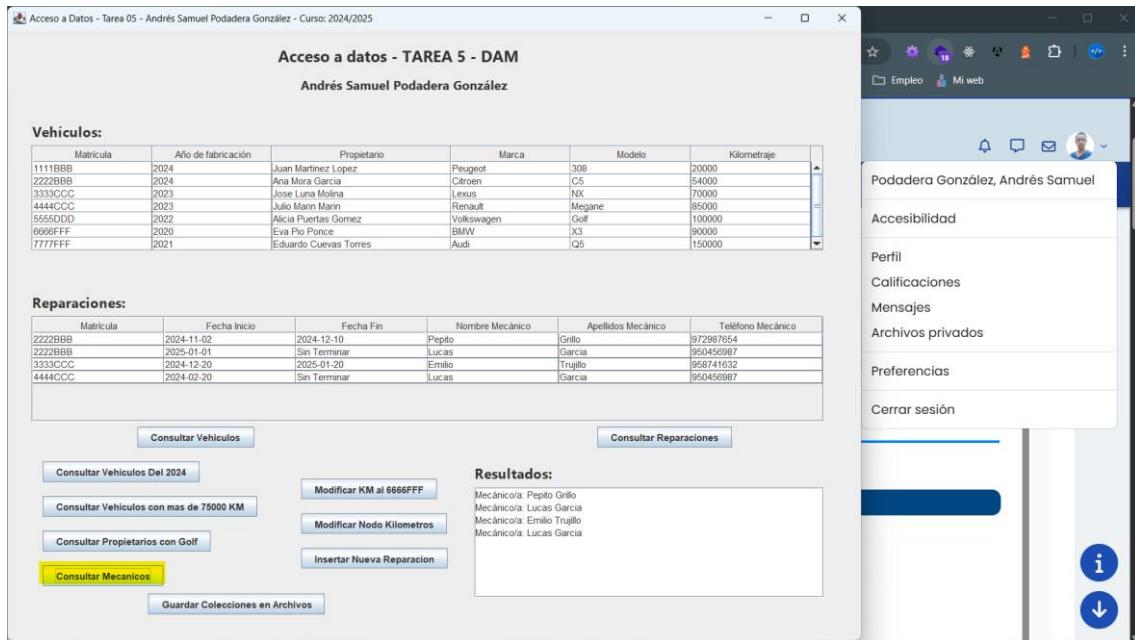
Para este caso haremos clic en el botón con texto “Consultar Propietarios con Golf” y los resultados deberían aparecer en el cuadro de texto.

Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025



d. Nombre y apellidos de todos los mecánicos de vehículos.

Para este apartado, haremos clic en el botón con texto “Consultar Mecánicos” y nos deberían aparecer los mecánicos en el cuadro de texto:



EJERCICIO 3.

Crea estas consultas de actualización .xq para la BD taller.xml y comprueba su resultado desde la aplicación Java:

- a. Sustituir el valor del nodo kilómetros a 110000 en el producto cuya matrícula=6666FFF.

Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025

Para este apartado primero observamos en la tabla que el registro contiene una cifra de kilómetros de 90000.

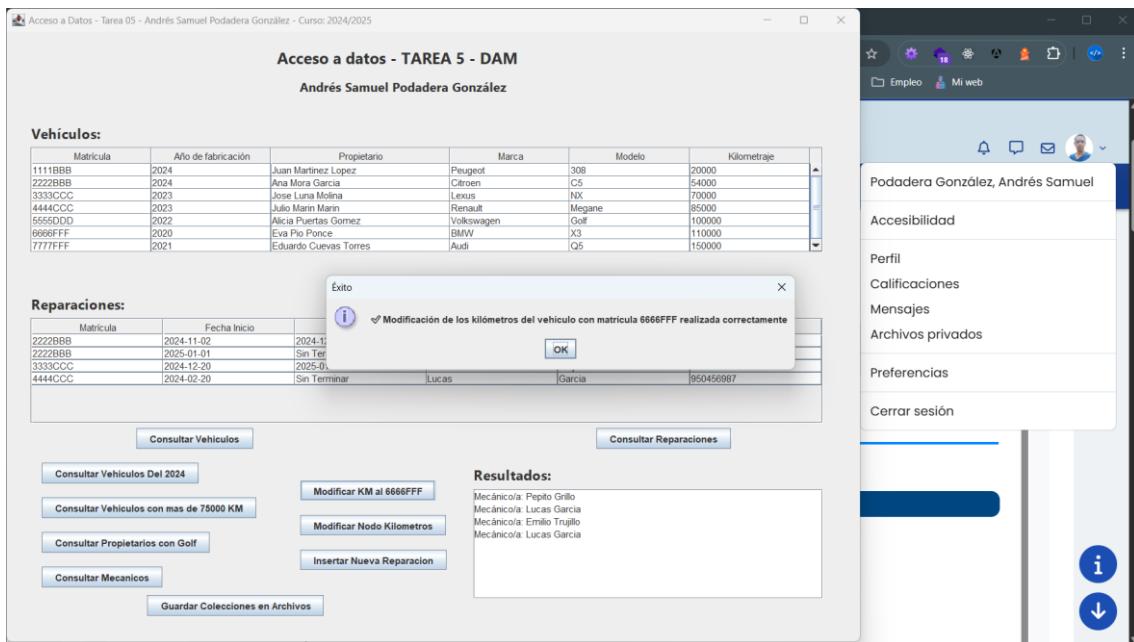
The screenshot shows a Windows desktop environment with the application window titled 'Acceso a datos - TAREA 5 - DAM' in the foreground. The application interface includes sections for 'Vehículos' and 'Reparaciones'. In the 'Vehículos' section, there is a table with columns: Matrícula, Año de fabricación, Propietario, Marca, Modelo, and Kilometraje. One row in the table has the Matrícula '6666FFF' and the Kilometraje '90000' highlighted with a yellow box. Below the table are several buttons: 'Consultar Vehículos', 'Consultar Reparaciones', 'Consultar Vehículos Del 2024', 'Consultar Vehículos con mas de 75000 KM', 'Consultar Propietarios con Golf', 'Consultar Mecánicos', and 'Guardar Colecciones en Archivos'. To the right of the application window is a sidebar with user profile information and navigation links like 'Empleo', 'Mi web', 'Perfil', 'Calificaciones', 'Mensajes', 'Archivos privados', 'Preferencias', and 'Cerrar sesión'.

Al pulsar en el botón con texto “Modificar KM al 6666FFF”:

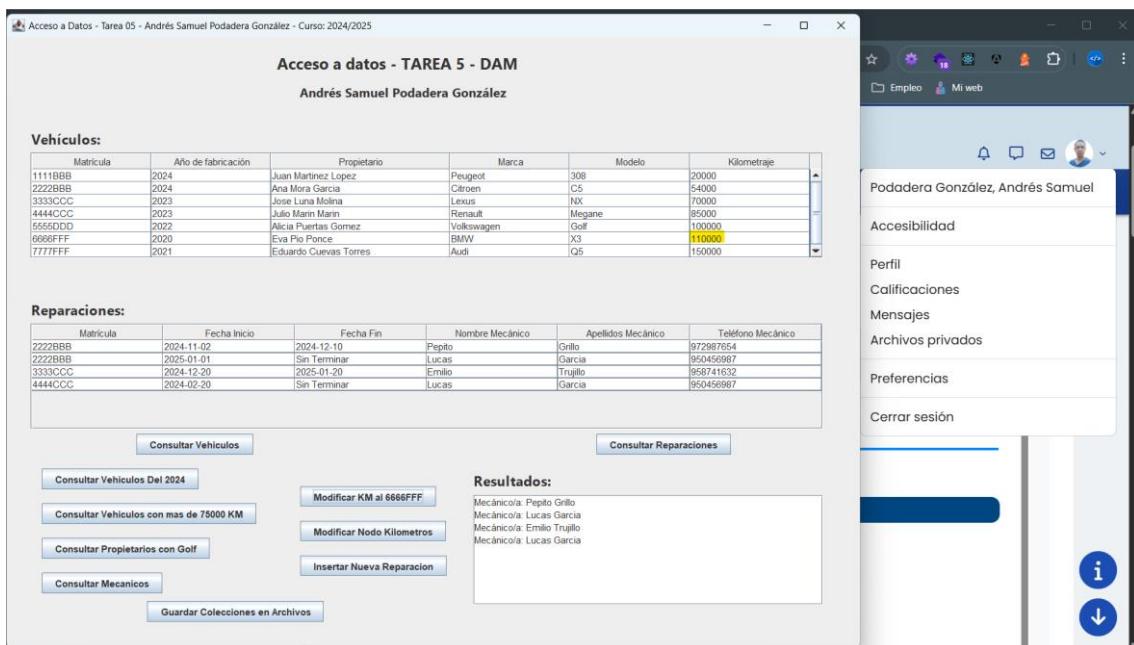
This screenshot shows the same application window after a modification. The 'Kilometraje' value for the row with Matrícula '6666FFF' has been changed from 90000 to 110000, as indicated by the updated value in the table. The rest of the interface remains the same, with the sidebar and other buttons visible.

La interfaz nos mostrará un mensaje de que el registro se ha modificado:

Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025



Y podremos observar en la tabla de resultados que efectivamente la cifra de kilómetros ha sido modificada a 110000.



- b. Modificar el nombre del nodo kilómetros de cada vehículo de la colección vehículos por kms.

Para este apartado, pulsaremos sobre el botón con texto “Modificar Nodo Kilómetros” y en los resultados nos aparecerán los nodos modificados.

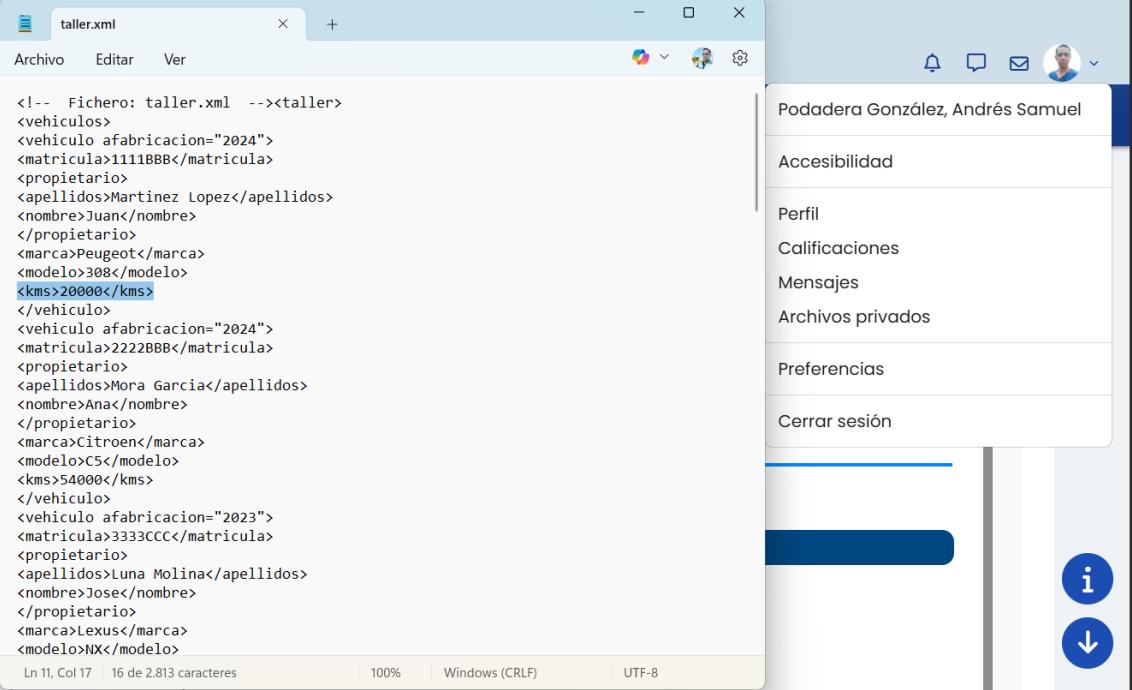
Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025

The screenshot shows a Windows desktop environment. On the left, a file explorer window is open, displaying a folder structure under 'C:\Users\andres\OneDrive\Escritorio\ProgramaAD52425\AD52425'. The folder 'Taller_Modificado' is selected. On the right, a pinned application window titled 'Acceso a datos - TAREA 5 - Andrés Samuel Podadera González' is visible. This window contains two tables: 'Vehículos' and 'Reparaciones', and several buttons for vehicle and repair operations. A sidebar on the right of the application window shows user profile information and session options.

Hay que aclarar que los cambios que se producen sobre los nodos de la base de datos se van almacenando en un fichero diferente situado en la carpeta AD52425/Taller_modificado/taller.xml para evitar problemas con la ejecución de cualquier consulta si modificamos en archivo XML original.

The screenshot shows a Windows desktop environment. On the left, a file explorer window is open, displaying a folder structure under 'C:\Users\andres\OneDrive\Escritorio\ProgramaAD52425\AD52425'. The folder 'Taller_Modificado' is selected. On the right, a pinned application window titled 'Acceso a datos - TAREA 5 - Andrés Samuel Podadera González' is visible. A message box in the application window displays a list of kilometer values: <kms>20000</kms>, <kms>40000</kms>, <kms>60000</kms>, <kms>80000</kms>, <kms>100000</kms>, <kms>110000</kms>, and <kms>150000</kms>. A sidebar on the right of the application window shows user profile information and session options.

Si en este punto, accedemos al archivo taller.xml que se encuentra dentro del directorio podremos observar que los nodos <kilómetros> han sido sustituidos por <kms>.



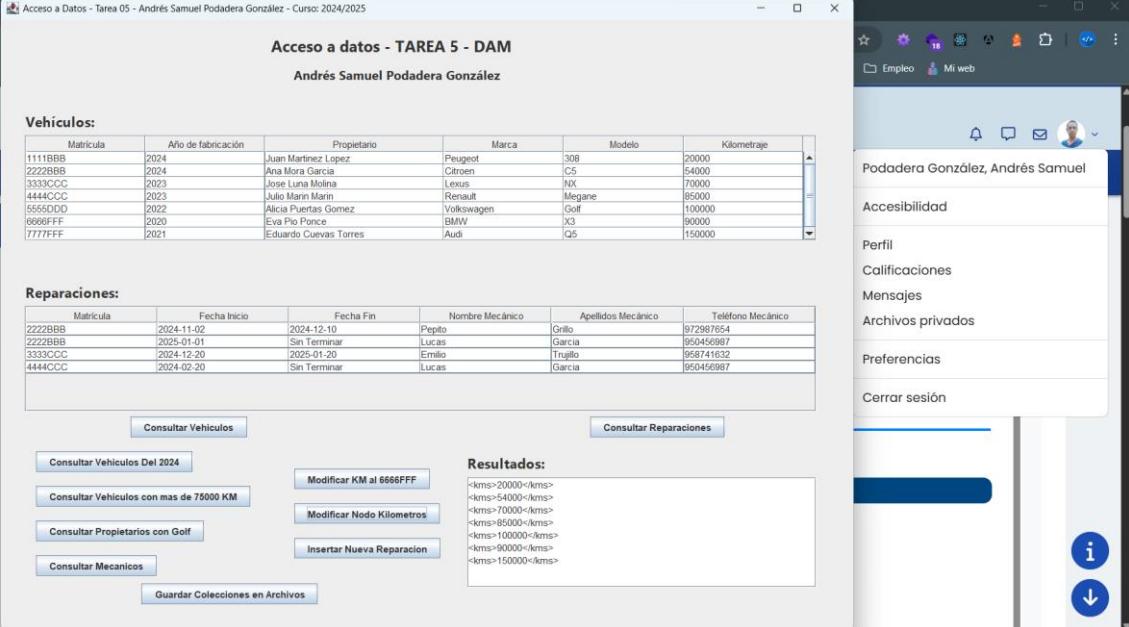
```

<!-- Fichero: taller.xml --><taller>
<vehiculos>
<vehiculo afabricacion="2024">
<matricula>1111BBB</matricula>
<propietario>
<apellidos>Martinez Lopez</apellidos>
<nombre>Juan</nombre>
</propietario>
<marca>Peugeot</marca>
<modelo>308</modelo>
<kms>20000</kms>
</vehiculo>
<vehiculo afabricacion="2024">
<matricula>2222BBB</matricula>
<propietario>
<apellidos>Mora Garcia</apellidos>
<nombre>Ana</nombre>
</propietario>
<marca>Citroen</marca>
<modelo>C5</modelo>
<kms>54000</kms>
</vehiculo>
<vehiculo afabricacion="2023">
<matricula>3333CCC</matricula>
<propietario>
<apellidos>Luna Molina</apellidos>
<nombre>Jose</nombre>
</propietario>
<marca>Lexus</marca>
<modelo>NX</modelo>
</vehiculo>
<vehiculo afabricacion="2023">
<matricula>4444CCC</matricula>
<propietario>
<apellidos>Alicia Puentes Gomez</apellidos>
<nombre>Eva Pio Ponce</nombre>
</propietario>
<marca>Volkswagen</marca>
<modelo>Golf</modelo>
<kms>100000</kms>
</vehiculo>
<vehiculo afabricacion="2022">
<matricula>5555DDO</matricula>
<propietario>
<apellidos>Julio Marin Martin</apellidos>
<nombre>Emilio</nombre>
</propietario>
<marca>Renault</marca>
<modelo>Megane</modelo>
<kms>85000</kms>
</vehiculo>
<vehiculo afabricacion="2020">
<matricula>6666FFF</matricula>
<propietario>
<apellidos>Alicia Puertas Gomez</apellidos>
<nombre>Eva Pio Ponce</nombre>
</propietario>
<marca>BMW</marca>
<modelo>X3</modelo>
<kms>90000</kms>
</vehiculo>
<vehiculo afabricacion="2021">
<matricula>7777FFF</matricula>
<propietario>
<apellidos>Eduardo Cuevas Torres</apellidos>
<nombre>Eduardo</nombre>
</propietario>
<marca>Audi</marca>
<modelo>Q5</modelo>
<kms>150000</kms>
</vehiculo>
</vehiculos>

```

- c. Inserta una nueva reparación con los siguientes datos:
 matrícula:3333CCC, inicio:02/03/2025, mecánico: Lucas García -950456987.

Comprobamos que la tabla de reparaciones contiene 4 registros:



Vehículos:						
Matrícula	Año de fabricación	Propietario	Marca	Modelo	Kilometraje	
1111BBB	2024	Juan Martinez Lopez	Peugeot	308	20000	
2222BBB	2024	Ana Mora Garcia	Citroen	C5	54000	
3333CCC	2023	Laura Luna Molina	Volkswagen	Golf	70000	
4444CCC	2023	Julio Marin Martin	Renault	Megane	85000	
5555DDO	2022	Alicia Puertas Gomez	Volkswagen	Golf	100000	
6666FFF	2020	Eva Pio Ponce	BMW	X3	90000	
7777FFF	2021	Eduardo Cuevas Torres	Audi	Q5	150000	

Reparaciones:						
Matrícula	Fecha Inicio	Fecha Fin	Nombre Mecánico	Apellidos Mecánico	Teléfono Mecánico	
2222BBB	2024-11-02	2024-12-10	Pepito	Grillo	972987654	
2222BBB	2025-01-01	Sin Terminar	Lucas	García	950456987	
3333CCC	2024-12-20	2025-01-20	Emilio	Trujillo	958741632	
4444CCC	2024-02-20	Sin Terminar	Lucas	García	950456987	

Consultar Vehículos
Consultar Reparaciones

Consultar Vehículos Del 2024
Modificar KM al 6666FFF

Consultar Vehículos con mas de 75000 KM
Modificar Nodo Kilometros

Consultar Propietarios con Golf
Insertar Nueva Reparación

Consultar Mecánicos
Guardar Colecciones en Archivos

Resultados:
 <kms>20000</kms>
 <kms>54000</kms>
 <kms>70000</kms>
 <kms>85000</kms>
 <kms>100000</kms>
 <kms>90000</kms>
 <kms>150000</kms>

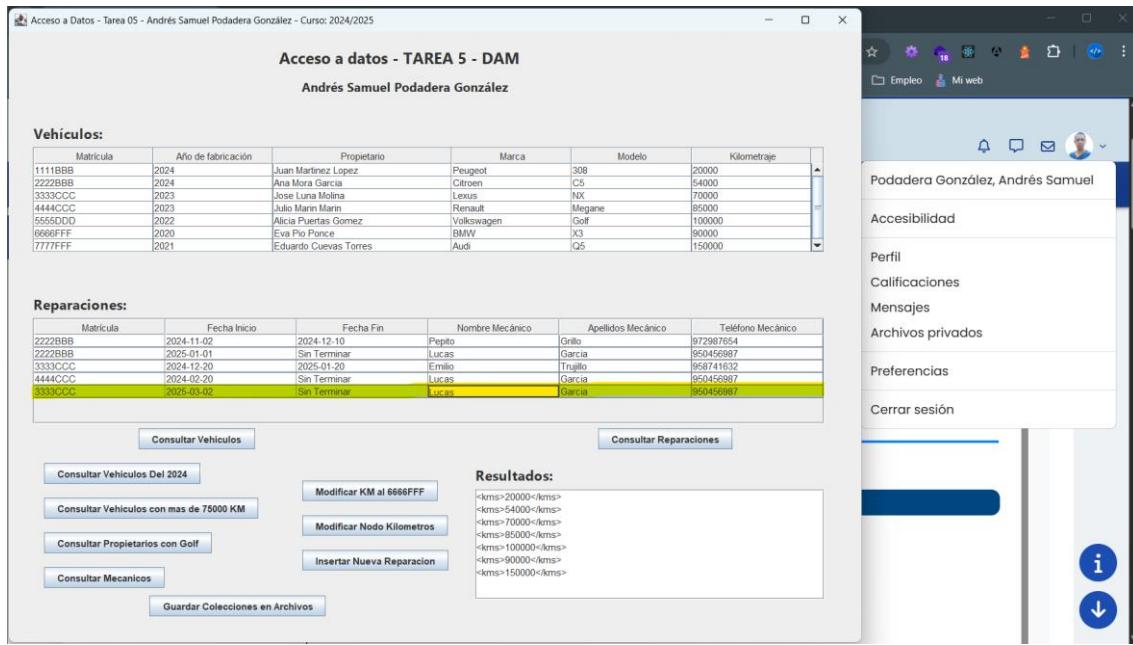
Pulsamos en el botón con texto “Insertar Nueva Reparacion”:

Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025

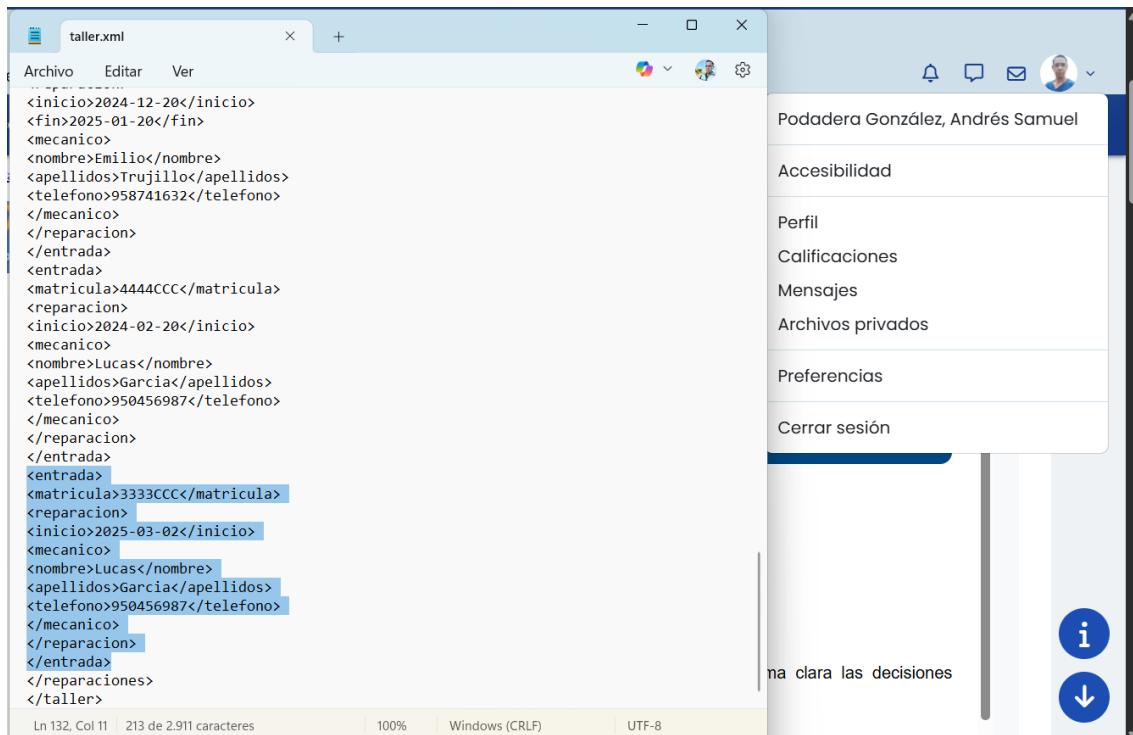
Y la interfaz nos mostrará un mensaje de aviso de operación realizada correctamente:

Y si observamos en la tabla de reparaciones veremos que se ha insertado una nueva fila con los datos solicitados:

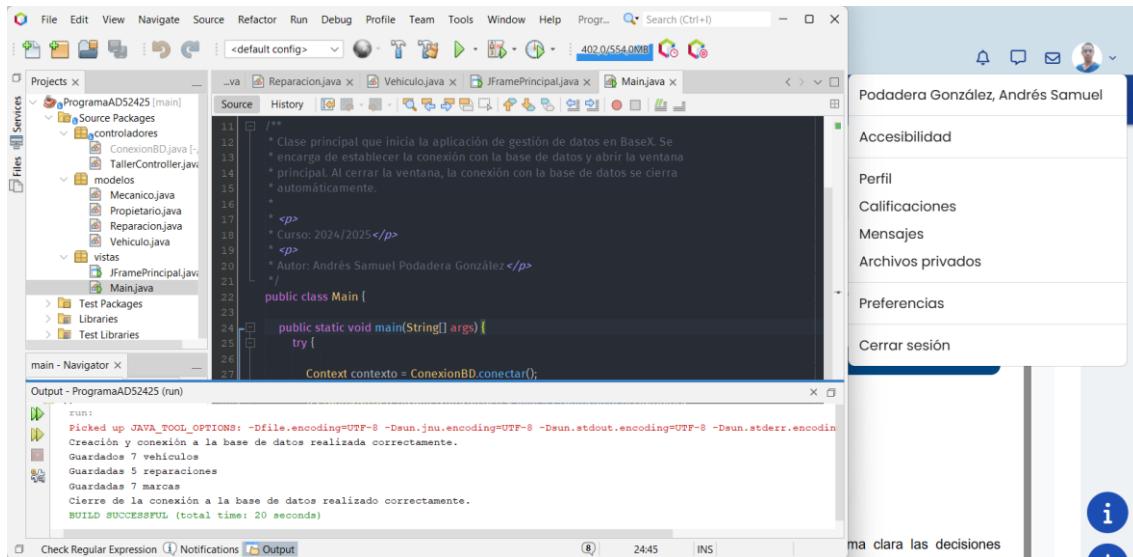
Nombre: Andrés Samuel Podadera González
 Ciclo: CFGS Desarrollo de aplicaciones multiplataforma
 Curso: 2024/2025



Y si nos dirigimos al archivo XML con el taller modificado(AD52425/Taller_Modificado/taller.xml) veremos que también se ha guardado la nueva reparación:

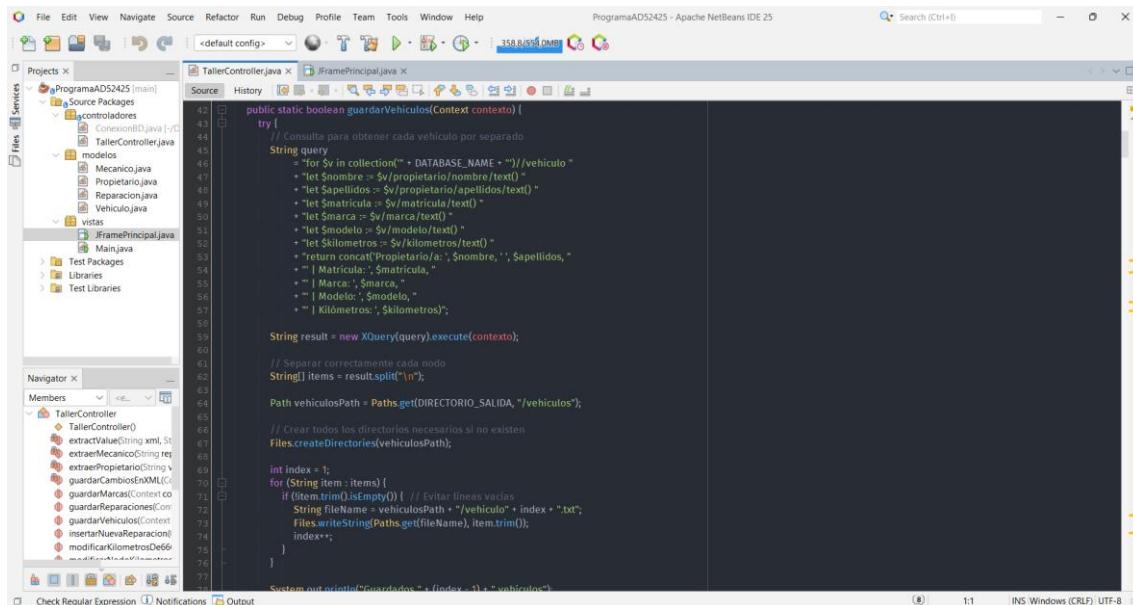


Cuando cerramos la aplicación, se nos muestra un mensaje de “Conexión cerrada” en la consola de NetBeans:



Aclaraciones y Explicaciones Generales de la tarea

En el ejercicio 1 mi propuesta de solución utiliza consultas XQuery para obtener la información del archivo xml, luego realiza un Split para dividir esa cadena por filas, obteniendo un array con los diferentes elementos ya sean vehículos, marcas o reparaciones:



Luego esas cadenas serán insertadas en los archivos txt con el nombre que se piden.

En el segundo ejercicio he creado las consultas en BaseX, las almacené cada una en archivos con extensión XQ para luego ejecutarlas desde el programa. El código lo que hace es acceder al archivo de la consulta obtener el texto, que en este caso será una consulta XQuery, y con el método adecuado la ejecutamos sobre nuestra base de datos en memoria para obtener los resultados esperados. Estos resultados serán devueltos a la vista para que sean mostrados al usuario a través de la interfaz gráfica.

```

    * Obtiene los vehículos del año 2024 desde la base de datos XML mediante
    * una consulta FLWOR. Este método carga la consulta desde un archivo XQuery
    * y la ejecuta en el contexto proporcionado. Si la consulta es exitosa,
    * devuelve el resultado en formato de cadena.
    *
    * @param contexto El contexto de BaseX utilizado para ejecutar la consulta
    * XQuery. Este contexto debe estar configurado adecuadamente para
    * interactuar con la base de datos XML.
    *
    * @return Un {@code String} con el resultado de la consulta FLWOR que
    * contiene los vehículos del 2024. Si ocurre un error durante la ejecución,
    * se retorna una cadena vacía.
    */
public static String obtenerVehiculos2024(Context contexto) {
    String resultado = "";
    try {
        // Consulta FLWOR para obtener todos los vehículos del 2024
        // Cargar la consulta desde el archivo
        String query = new String(Files.readAllBytes(Paths.get("Consultas_XQuery/Ejercicio2-1.xq")));
        resultado = new XQuery(query).execute(contexto);
    } catch (BaseXException ex) {
        System.err.println("Error obteniendo vehículos del 2024: " + ex.getMessage());
    } catch (IOException ex) {
        System.out.println("Error de entrada/salida: " + ex.getMessage());
    }
    return resultado;
}

```

El ejercicio 3, sigue el mismo “modus operandi” del ejercicio anterior con la salvedad que tras la ejecución de sentencias que modifican el XML, es decir, los datos cargados en memoria, seguidamente llamo a una función que me permite almacenarlos en un archivo llamado taller.xml dentro de la carpeta

Taller_Modificado. De esta forma verifico que efectivamente los cambios se realizan, pero sin alterar el archivo taller.xml original.

```

    * datos XML.
    * @return {@code true} si la modificación se realiza con éxito,
    * {@code false} si ocurre un error.
    */
public static boolean modificarKilometrosDe6666FFF(Context contexto) {
    try {
        // Consulta para modificar los kilómetros del vehículo con matrícula 6666FFF
        // Cargar la consulta desde el archivo
        String query = new String(Files.readAllBytes(Paths.get("Consultas_XQuery/Ejercicio3-1.xq")));
        new XQuery(query).execute(contexto);

        // Guardamos los cambios en el fichero XML
        guardarCambiosEnXML(contexto);

        return true;
    } catch (BaseXException ex) {
        System.err.println("Error modificando los kilómetros: " + ex.getMessage());
    } catch (IOException ex) {
        System.out.println("Error de entrada/salida: " + ex.getMessage());
    }
    return false;
}

/**
 * Modifica el nodo "kilómetros" de todos los vehículos en la base de datos
 * XML, reemplazándolo por "km". Luego, recupera los valores actualizados
 * para usarlos en la consulta.
 */

```

El método “guardarCambiosEnXML(contexto)” me guarda el contexto o el estado de la base de datos en memoria en el punto en que es llamado, permitiendo comprobar que los cambios efectivamente se han producido.

Aclaraciones: Todas las ejecuciones de sentencias XQuery y creación o acceso a ficheros son manejadas con bloques try/catch para el tratamiento de excepciones.