**Acceso a datos**

**Tarea Unidad 5**

Contenido

[Estructura de la aplicación 2](#_Toc194752234)

[Localización de ficheros 6](#_Toc194752235)

[EJERCICIO 1. 9](#_Toc194752236)

[EJERCICIO 2. 13](#_Toc194752237)

[EJERCICIO 3. 15](#_Toc194752238)

[Aclaraciones y Explicaciones Generales de la tarea 22](#_Toc194752239)

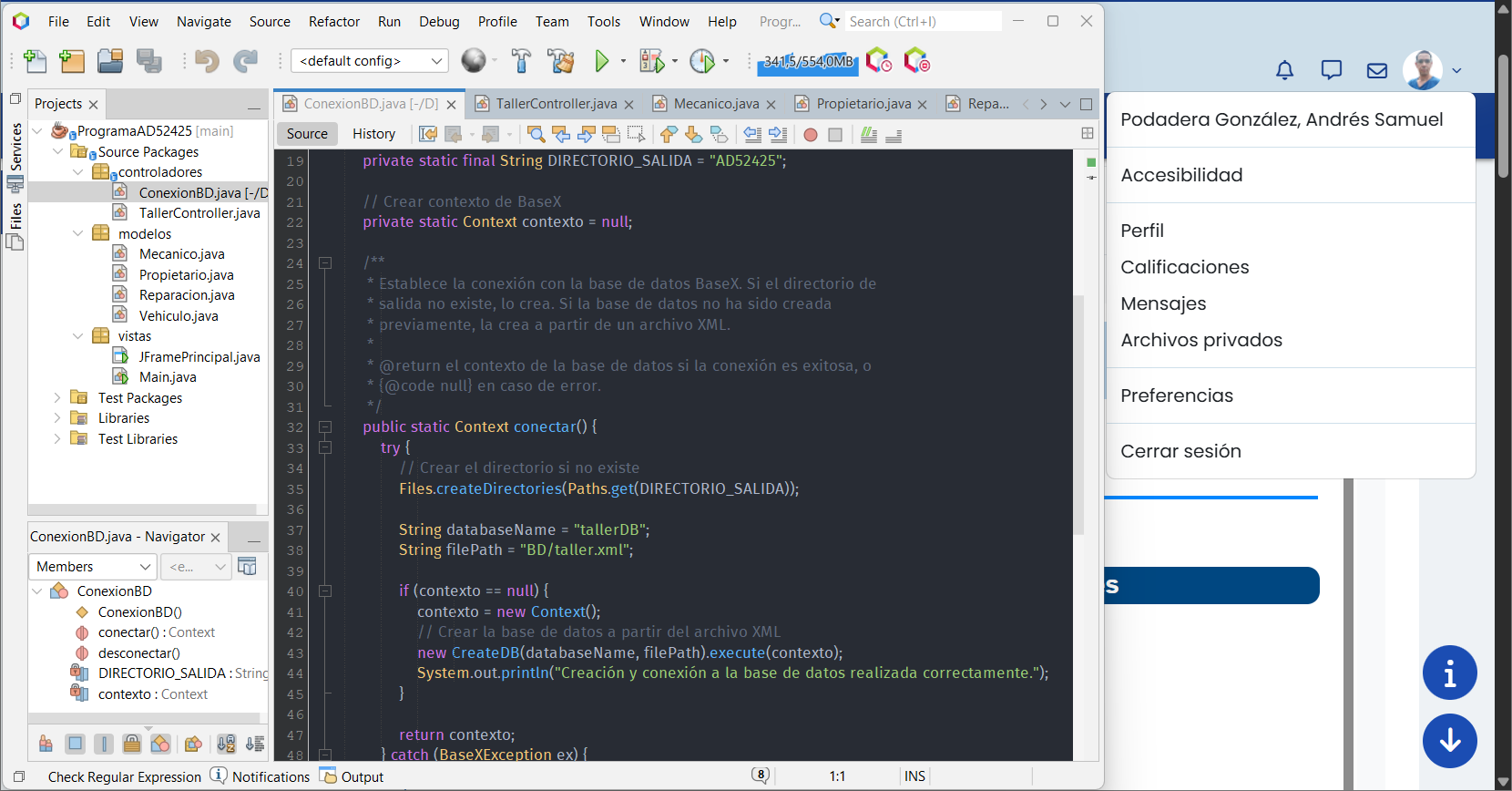
# Estructura de la aplicación

**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:**

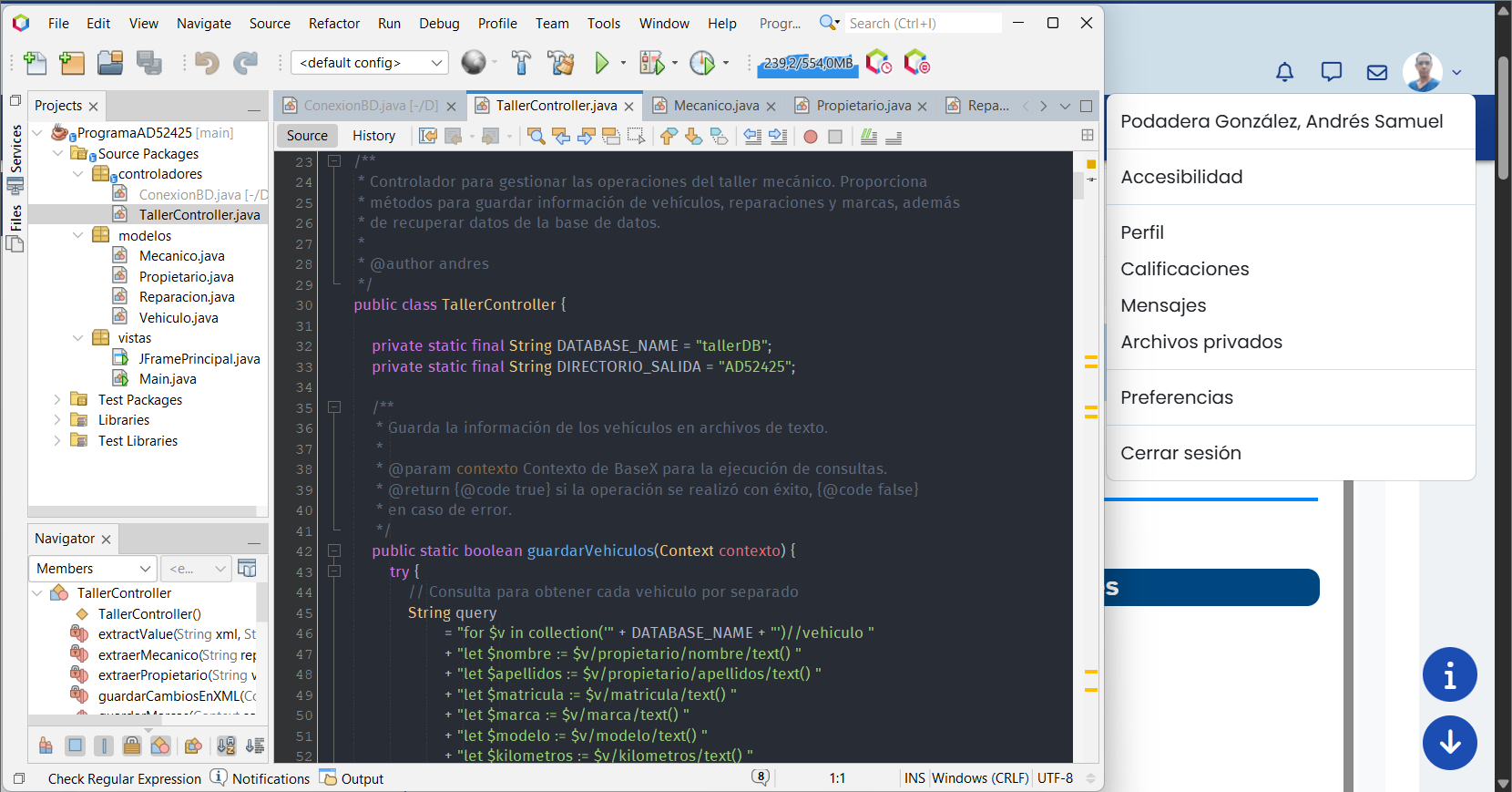
**ConexionBD.java**

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 se cerrar la conexión cuando la aplicación termina.



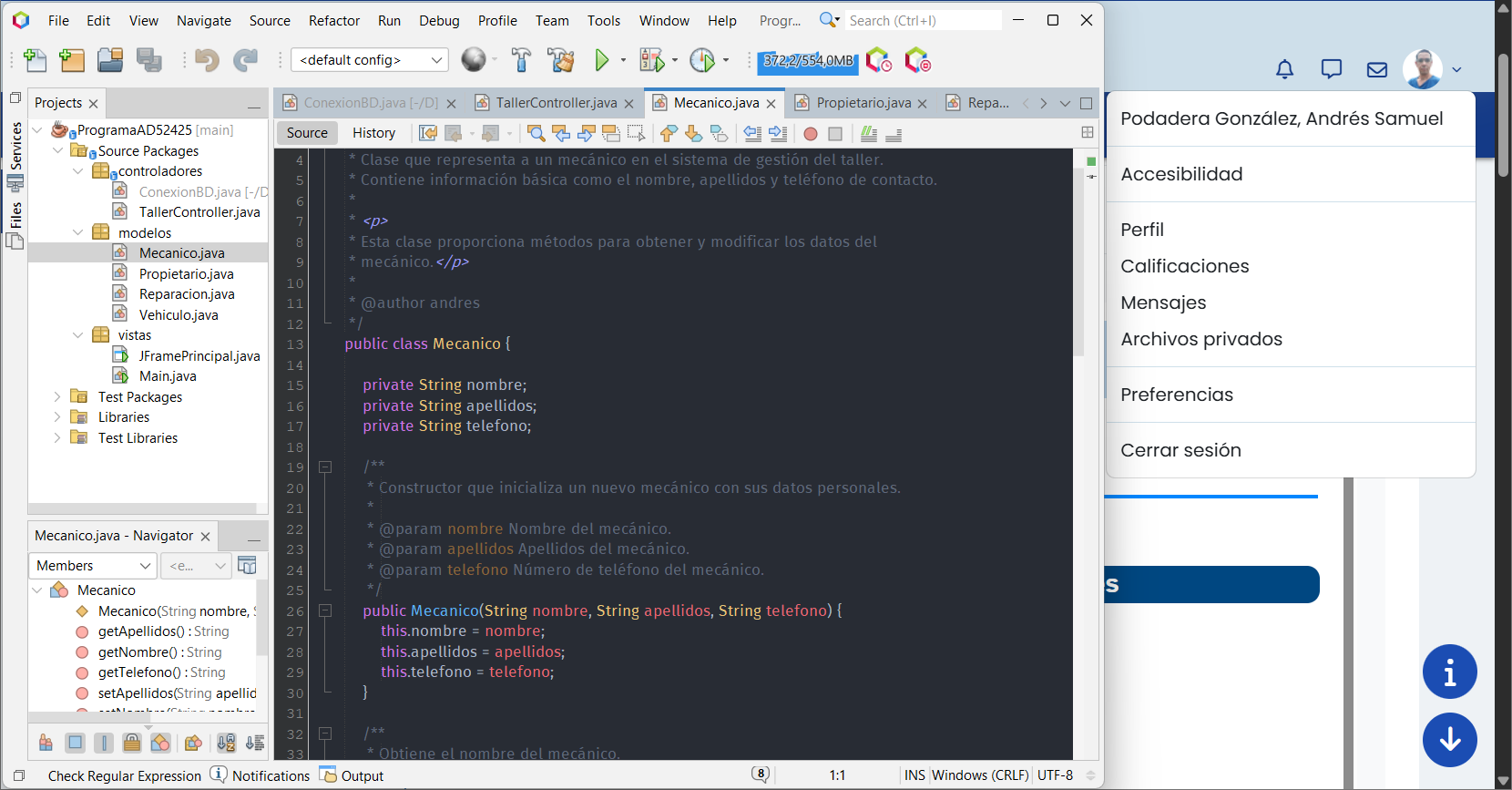
**TallerController.java**

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.



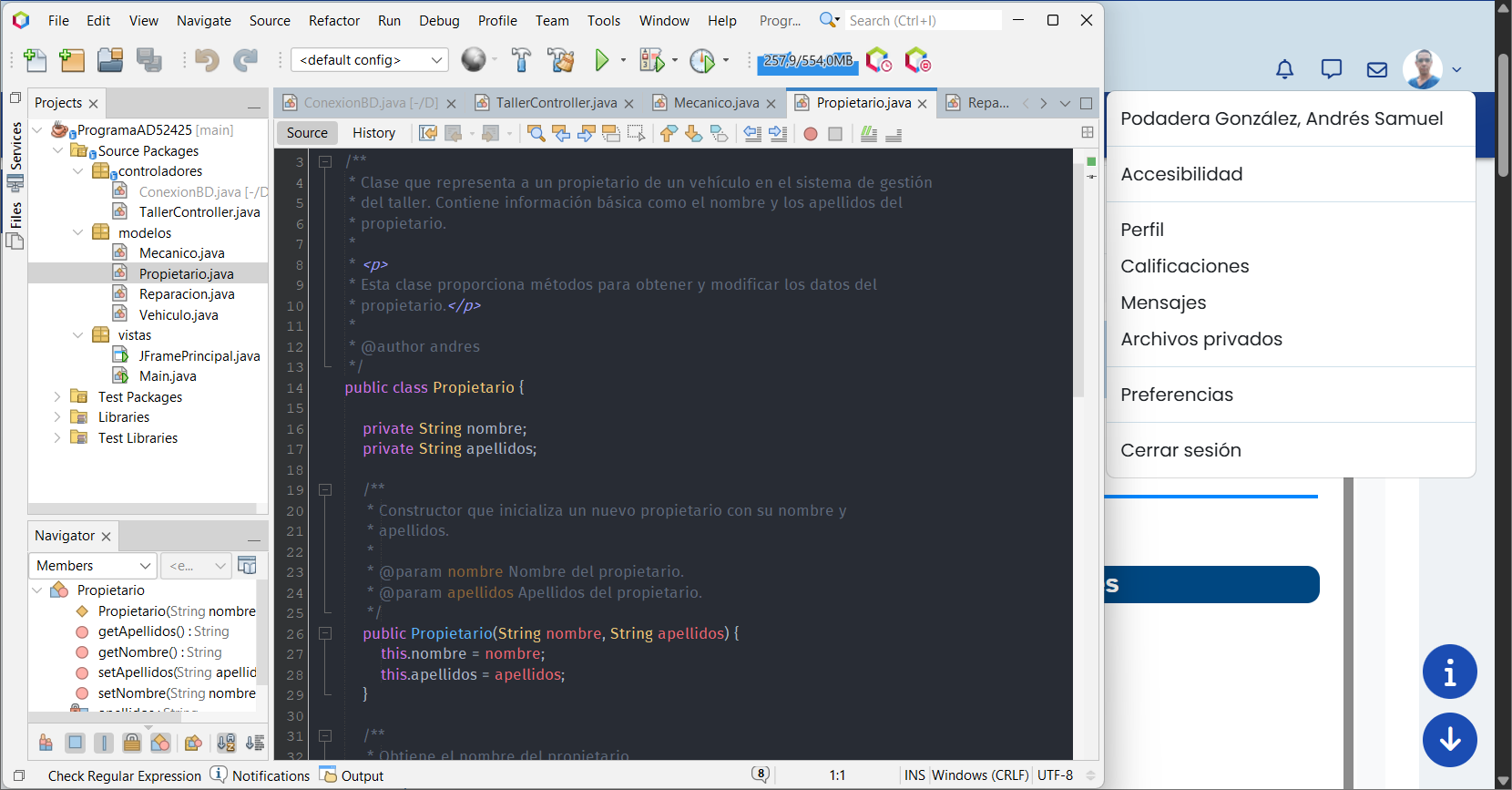
**Mecánico.java**

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.



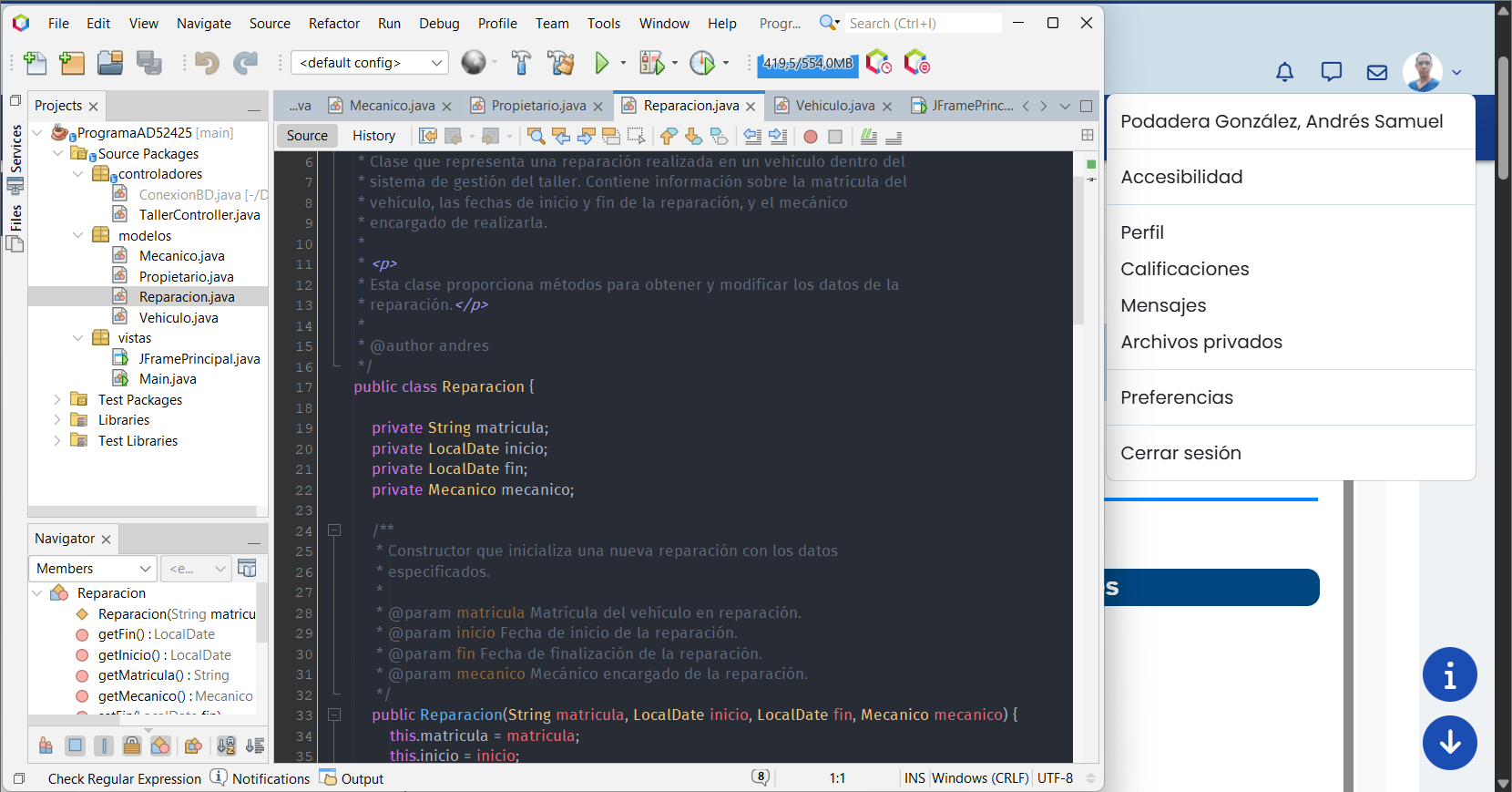
**Propietario.java**

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.



**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.



**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.



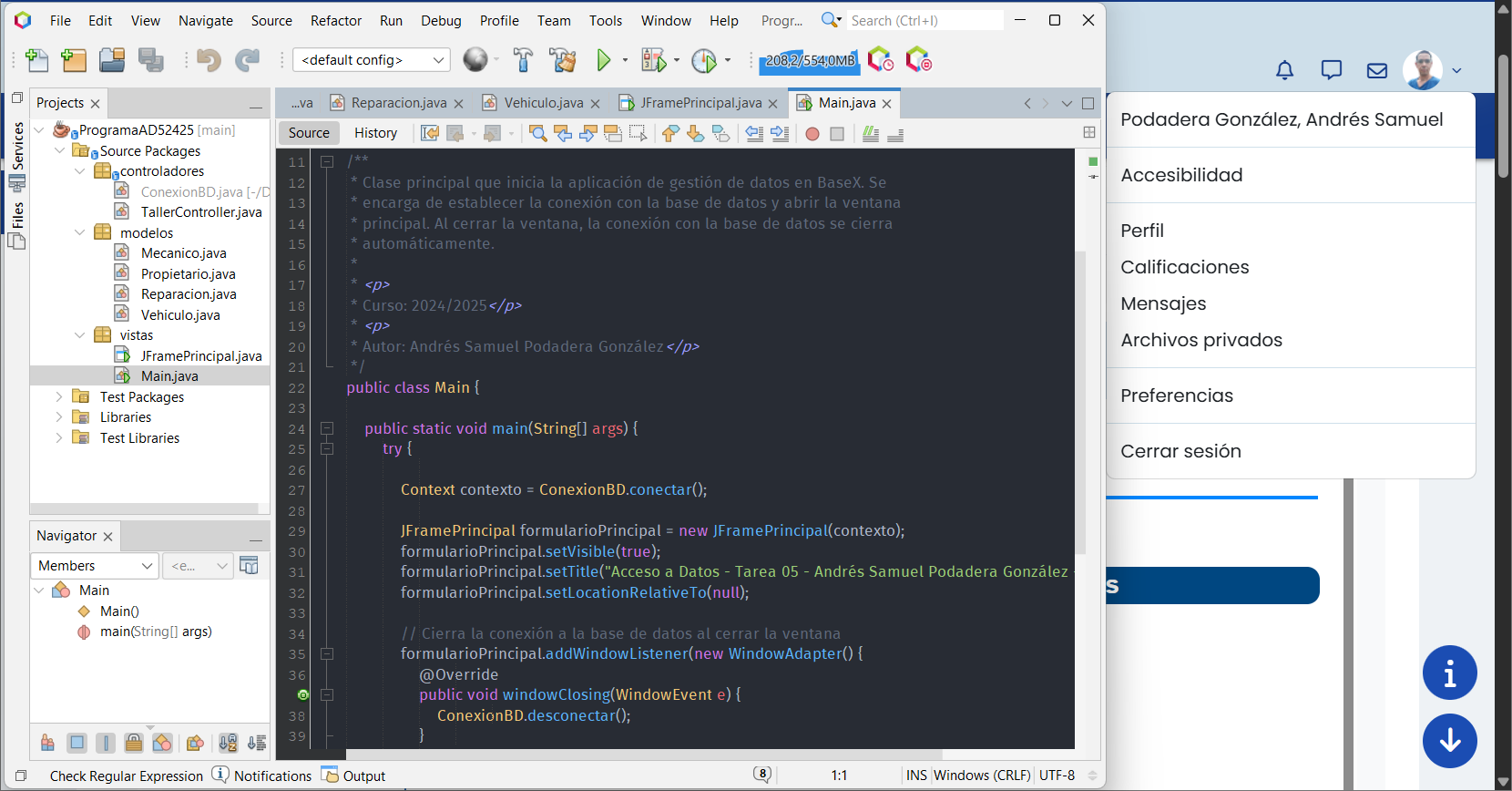
**JFramePrincipal.java**

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.



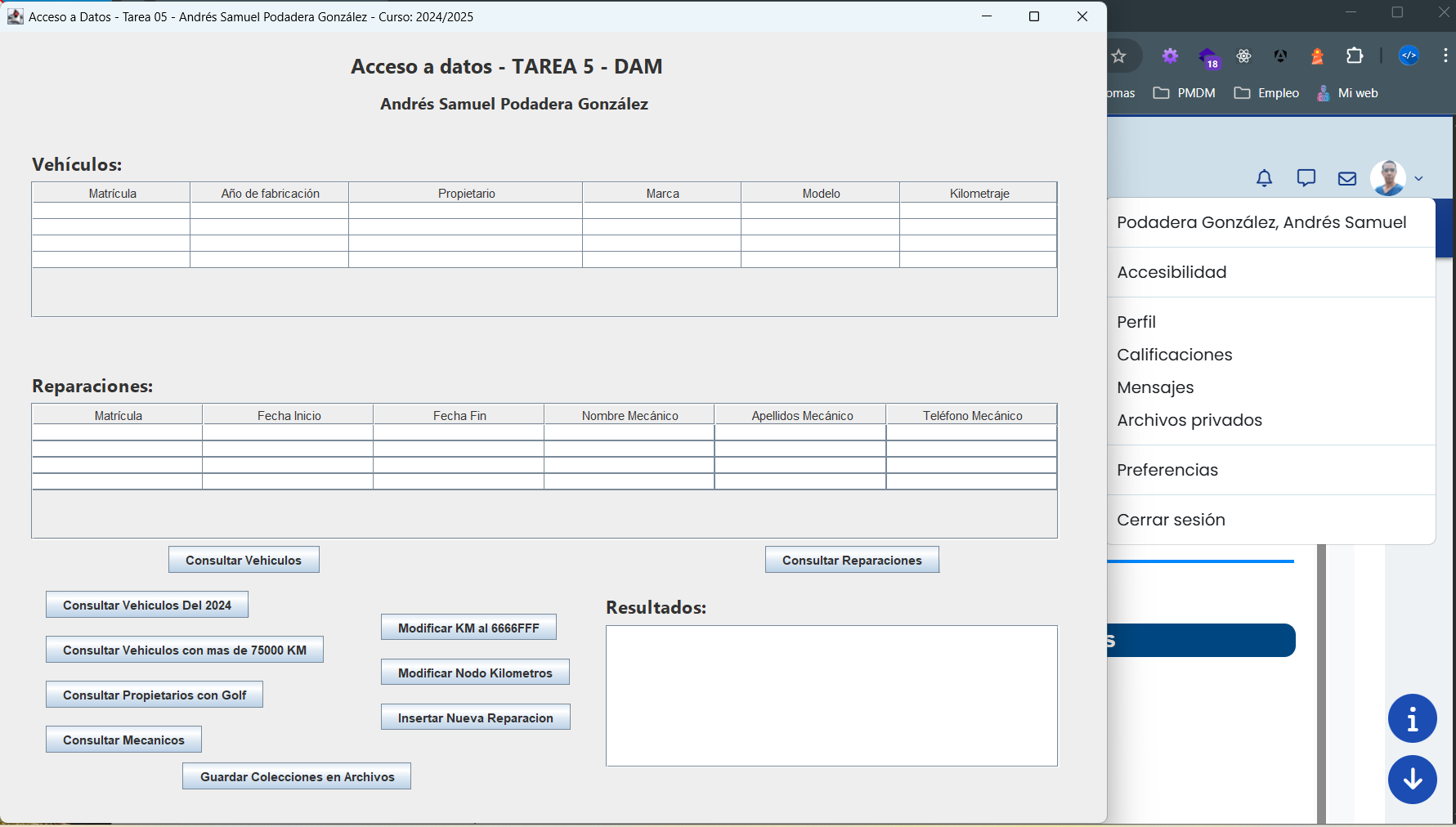
**Main.java**

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



**Aplicación en funcionamiento**

Nuestra aplicación una vez arrancada se presenta de esta forma:

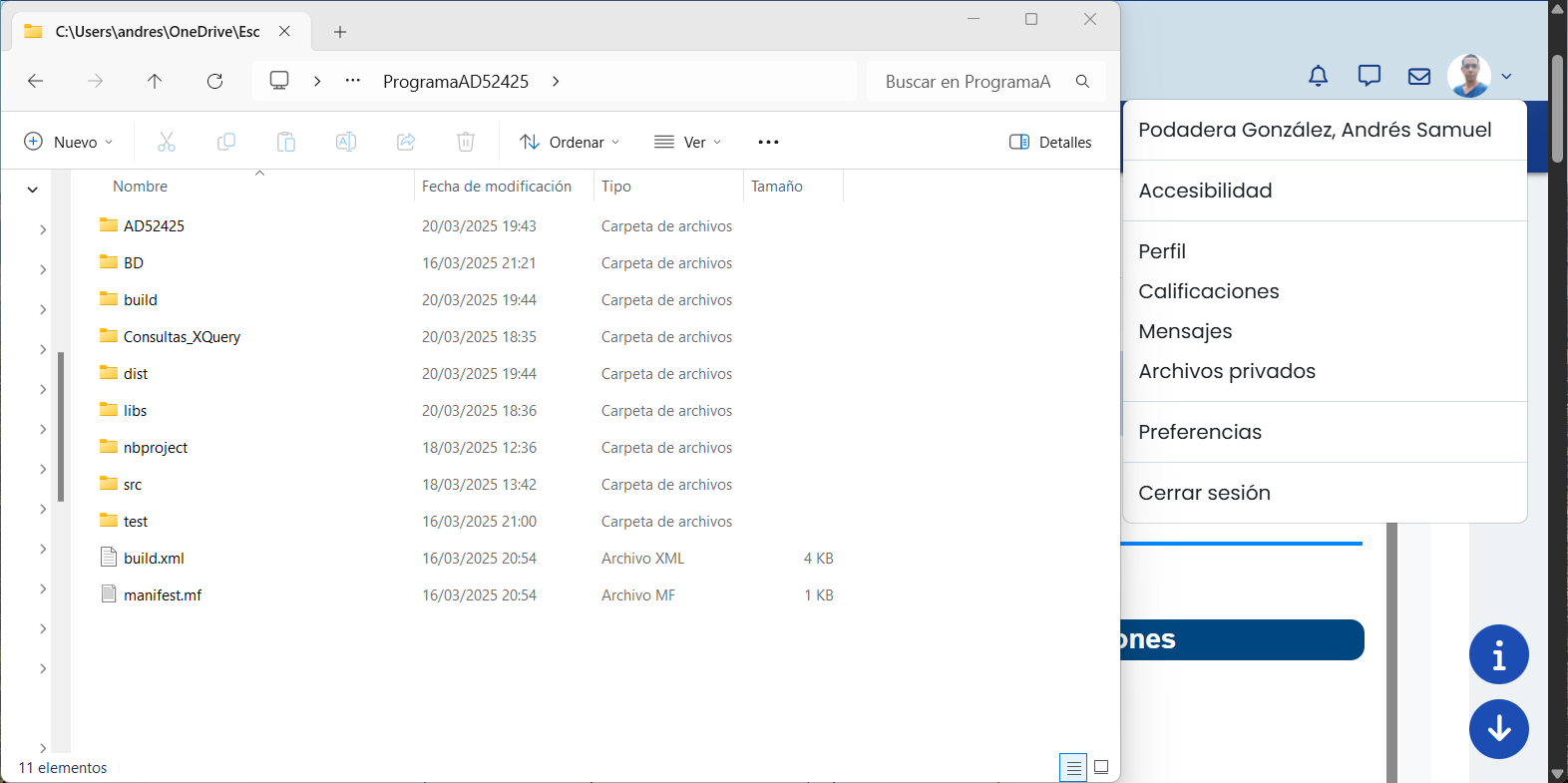


En la tabla Vehiculos, listamos los vehículos, en la tabla Reparaciones listamos las reparaciones registradas, el conjunto de botones nos permiten realizar las consultas de vehículos y reparaciones, además de realizar las operaciones que nos solicitan en la tarea como insertar nuevos nodos, modificarlos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

# Localización de ficheros

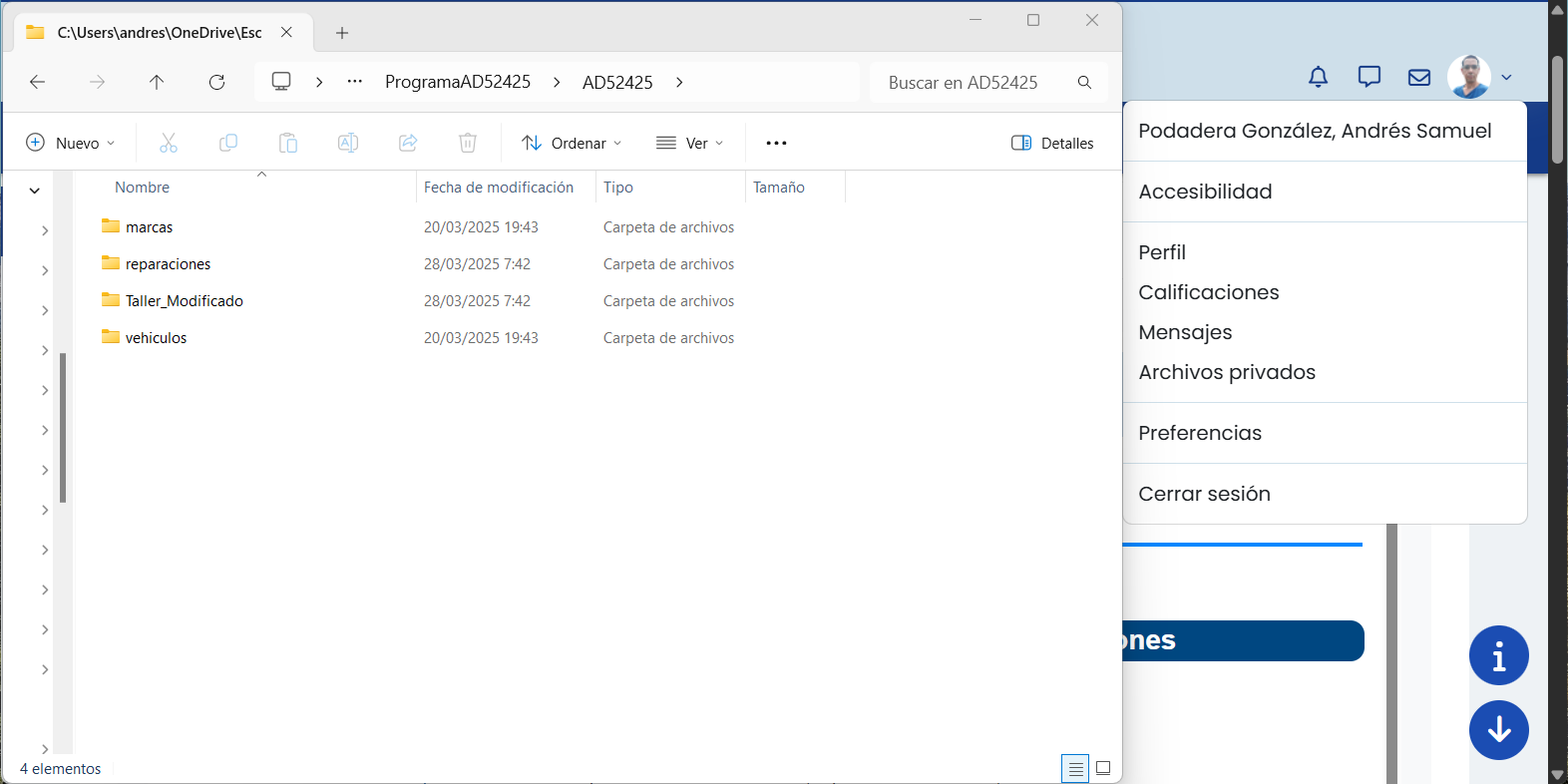
Dentro del nuestro proyecto los ficheros más importantes se encuentran dispuestos en:

**Estructura principal:**



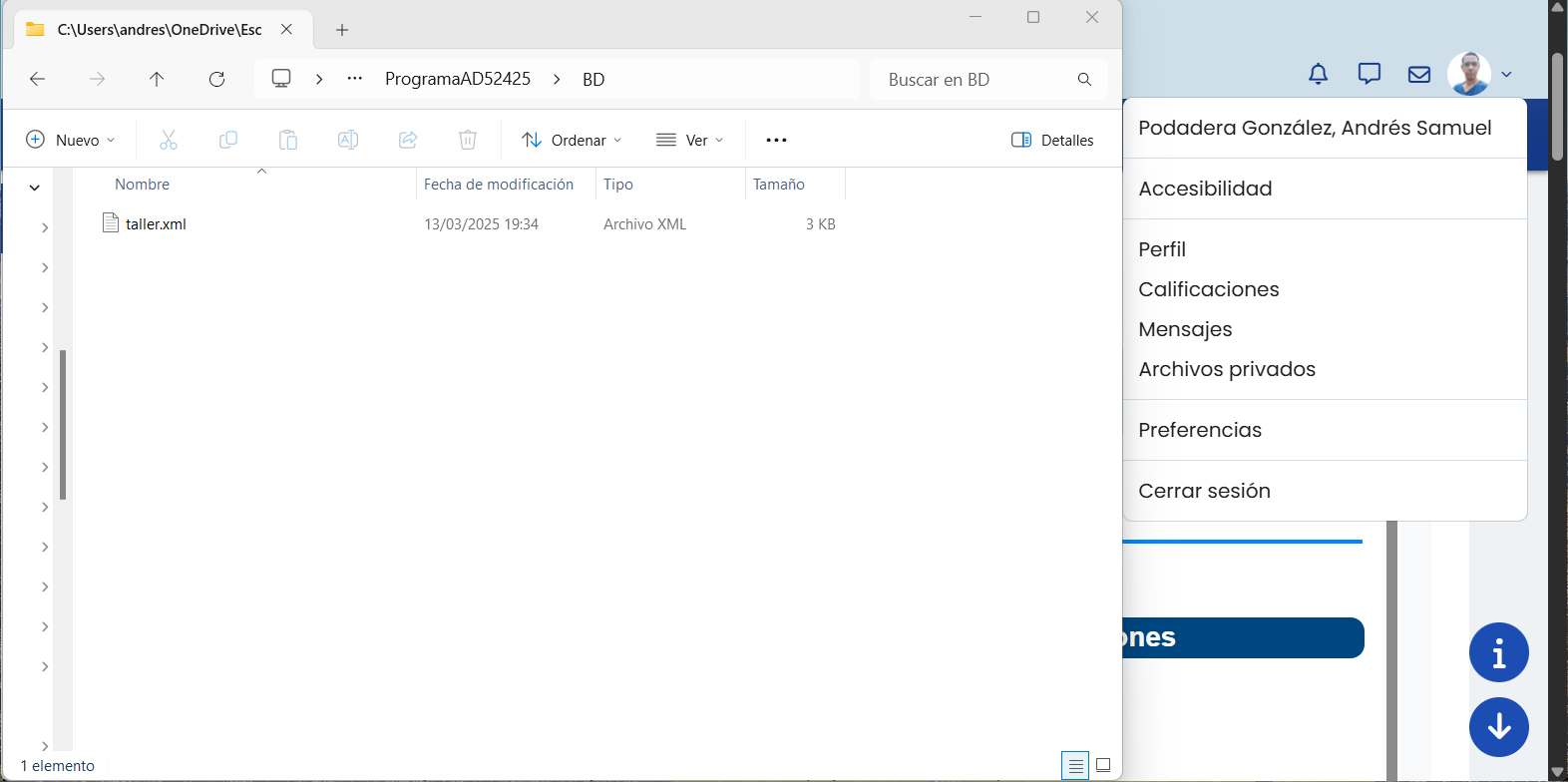
**Carpeta AD52425:**

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)



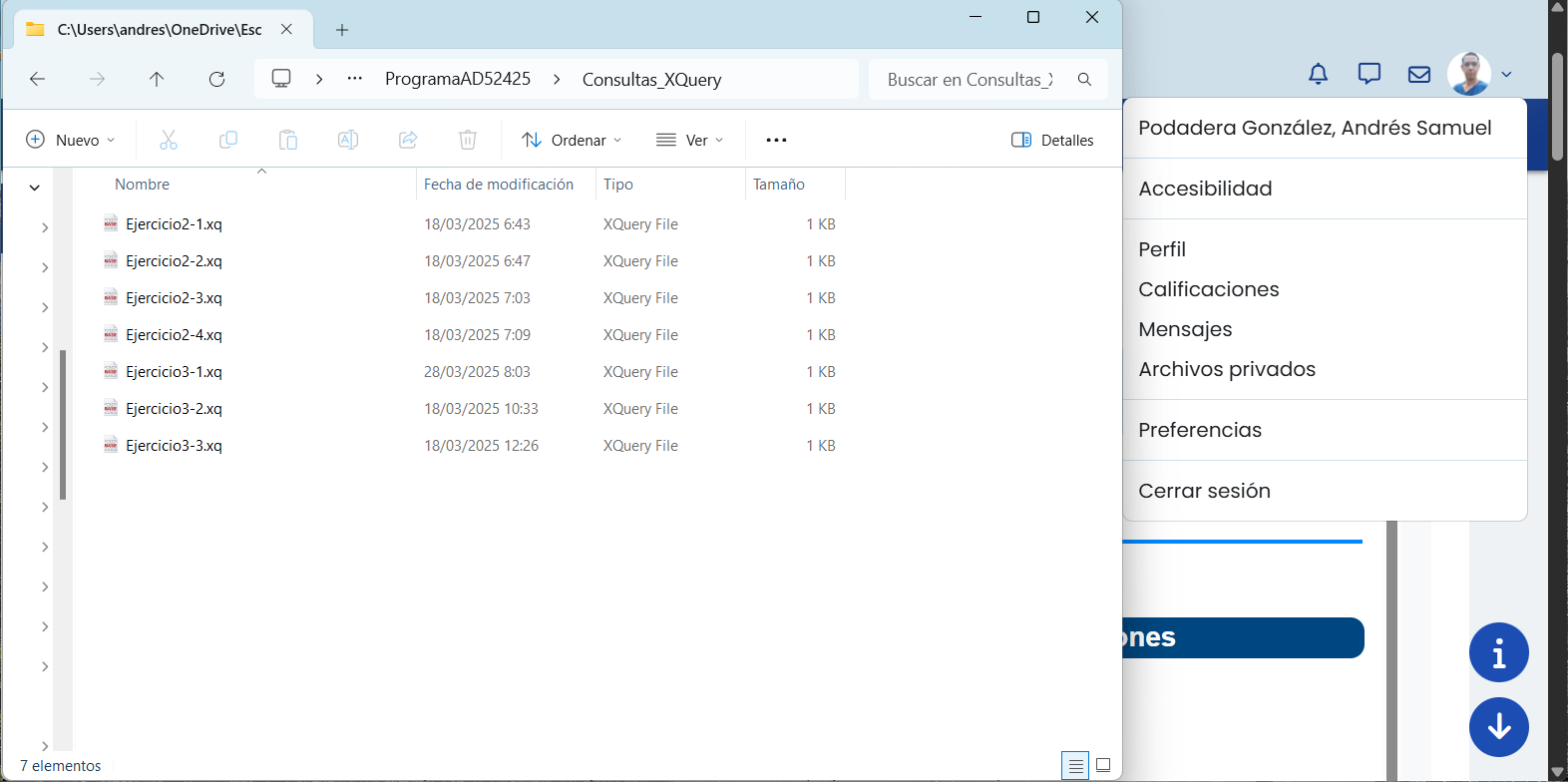
**Carpeta BD:**

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



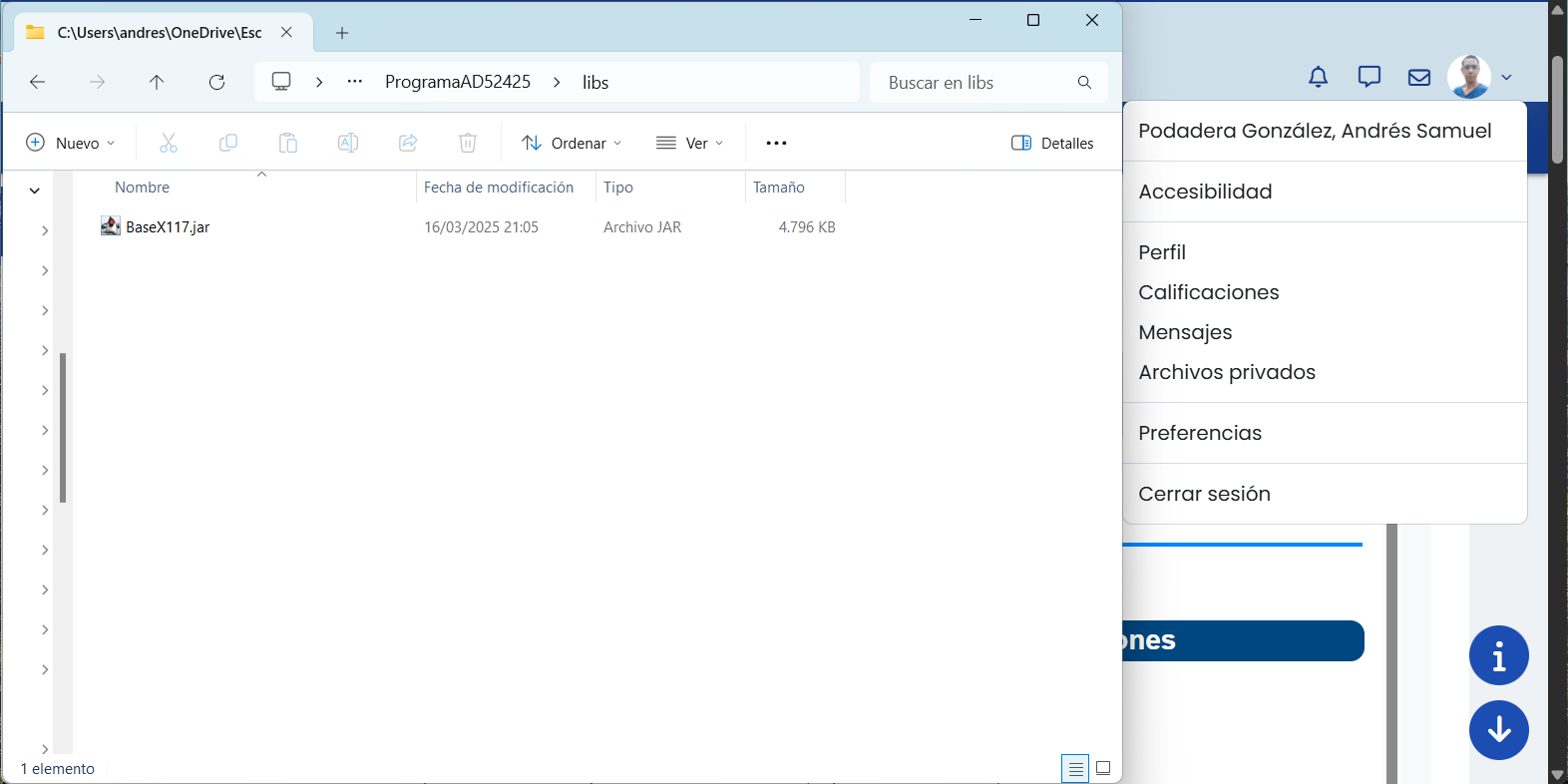
**Carpeta Consultas\_XQuery:**

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.



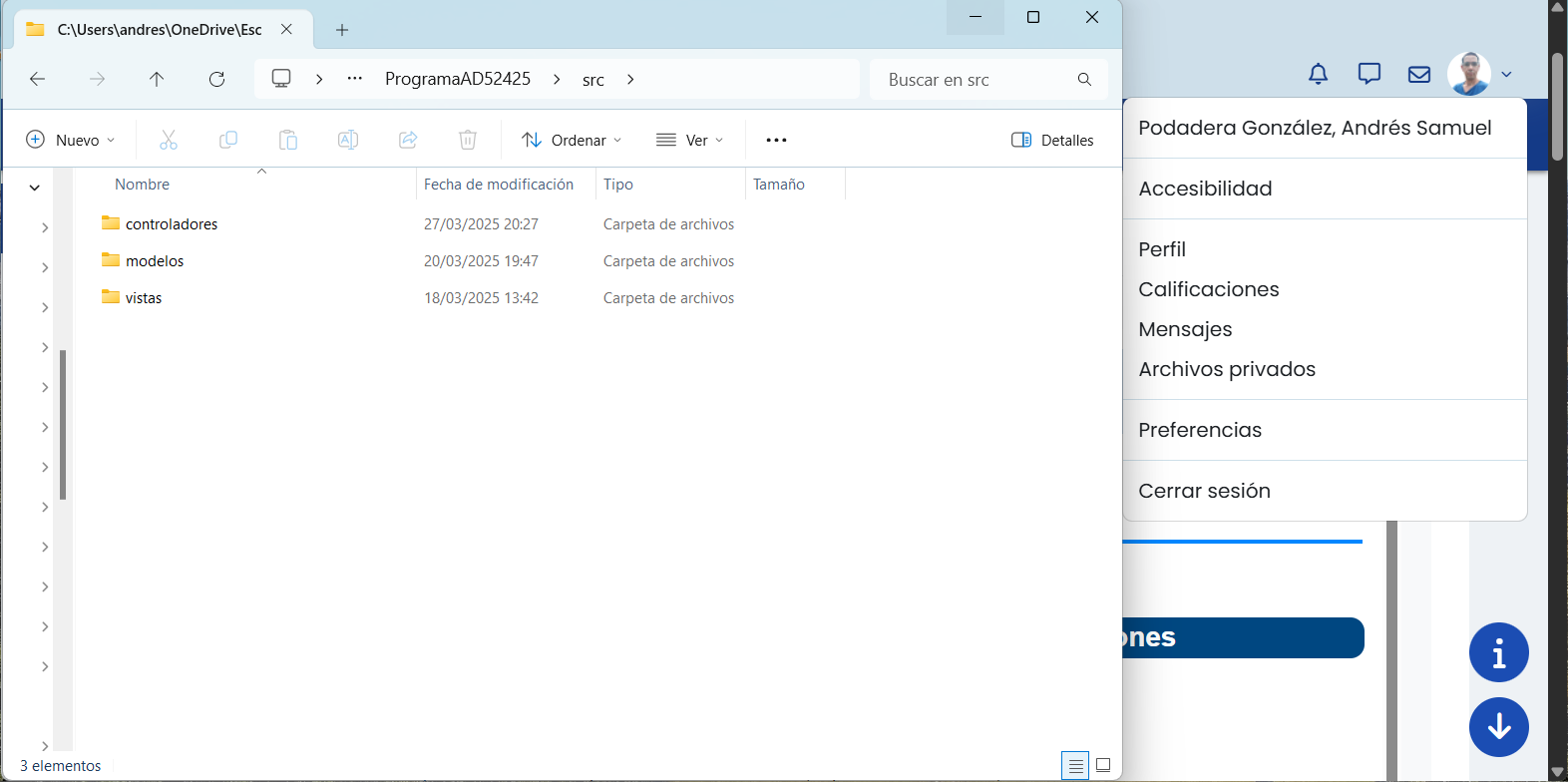
**Carpeta libs:**

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



**Carpeta src:**

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



# EJERCICIO 1.

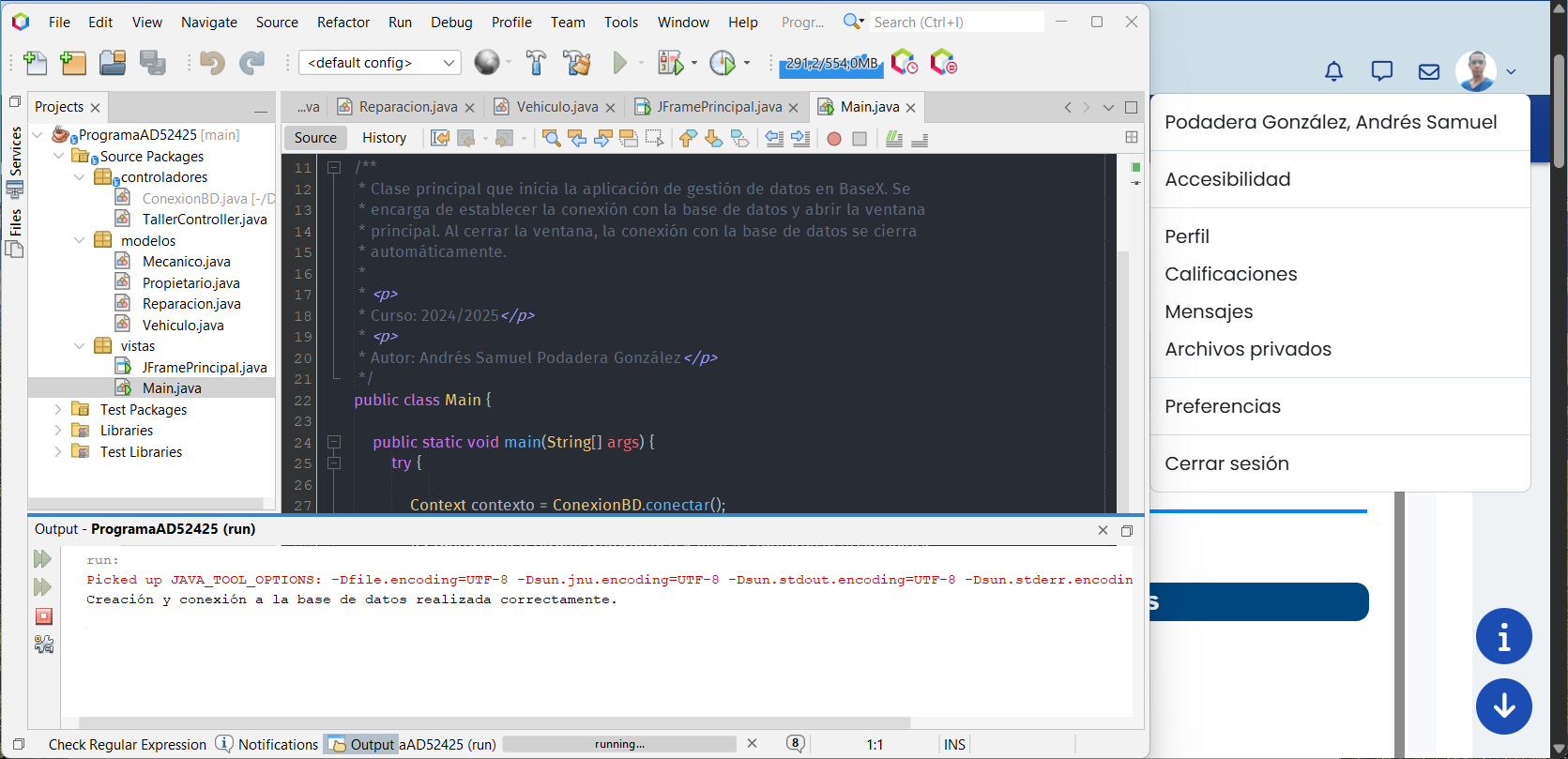
*Es necesario crear una aplicación con interfaz gráfica (usando Swing) en base al driver Basex.jar donde se realicen operaciones de lectura, inserción, modificación y borrado.*

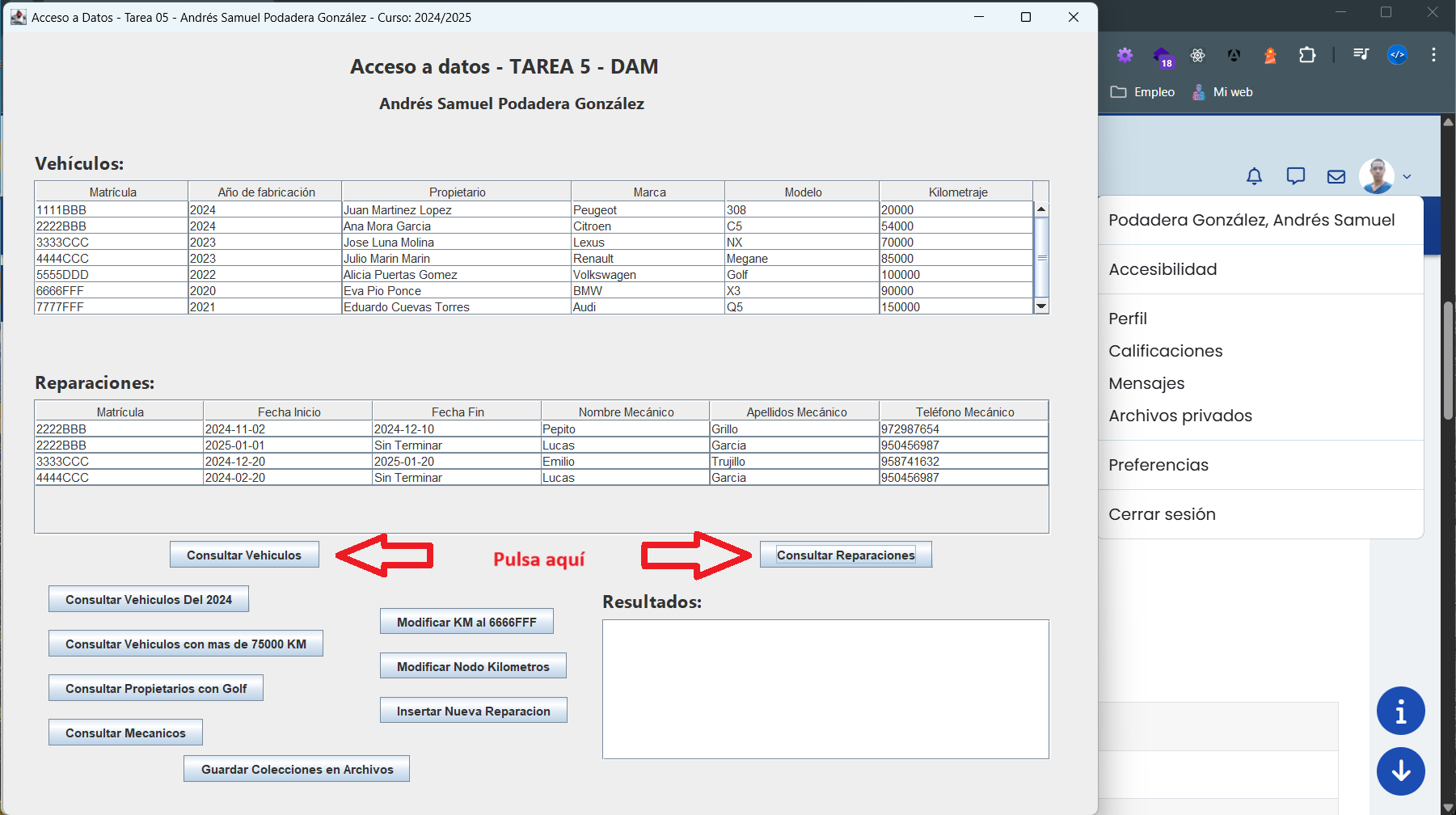
*Utilizando la base de datos*[*taller.xml*](https://drive.google.com/file/d/1KW7vKhikIh02rejXrL8vfD_SIJrVQ_T_/view?usp=sharing)*, extrae las colecciones 'Vehículos', 'Reparaciones' y 'Marcas' para realizar una copia local en el directorio de tu proyecto (AD52425) de manera que:*

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

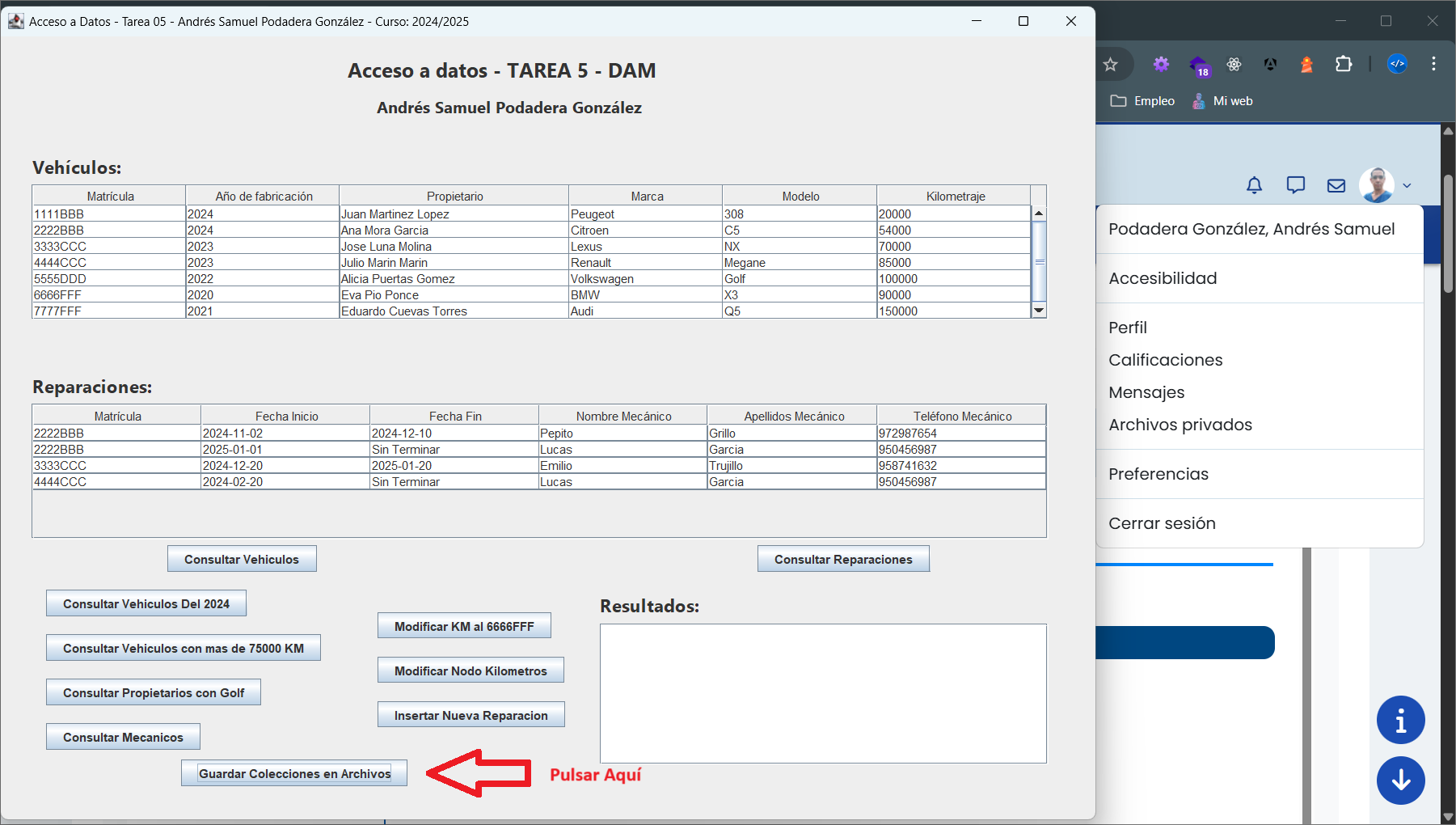
**Para comprobar que la aplicación realiza la operación que se pide comenzamos arrancando la aplicación y comprobando que dentro del proyecto no existe la carpeta AD22425 ya que es lo que se pide en este apartado, que la aplicación sea capaz de crear carpetas y ficheros con la información del archivo XML.**

**Empezaremos arrancando la aplicación y haciendo clic en los botones de “Consultar Vehiculos” 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:**

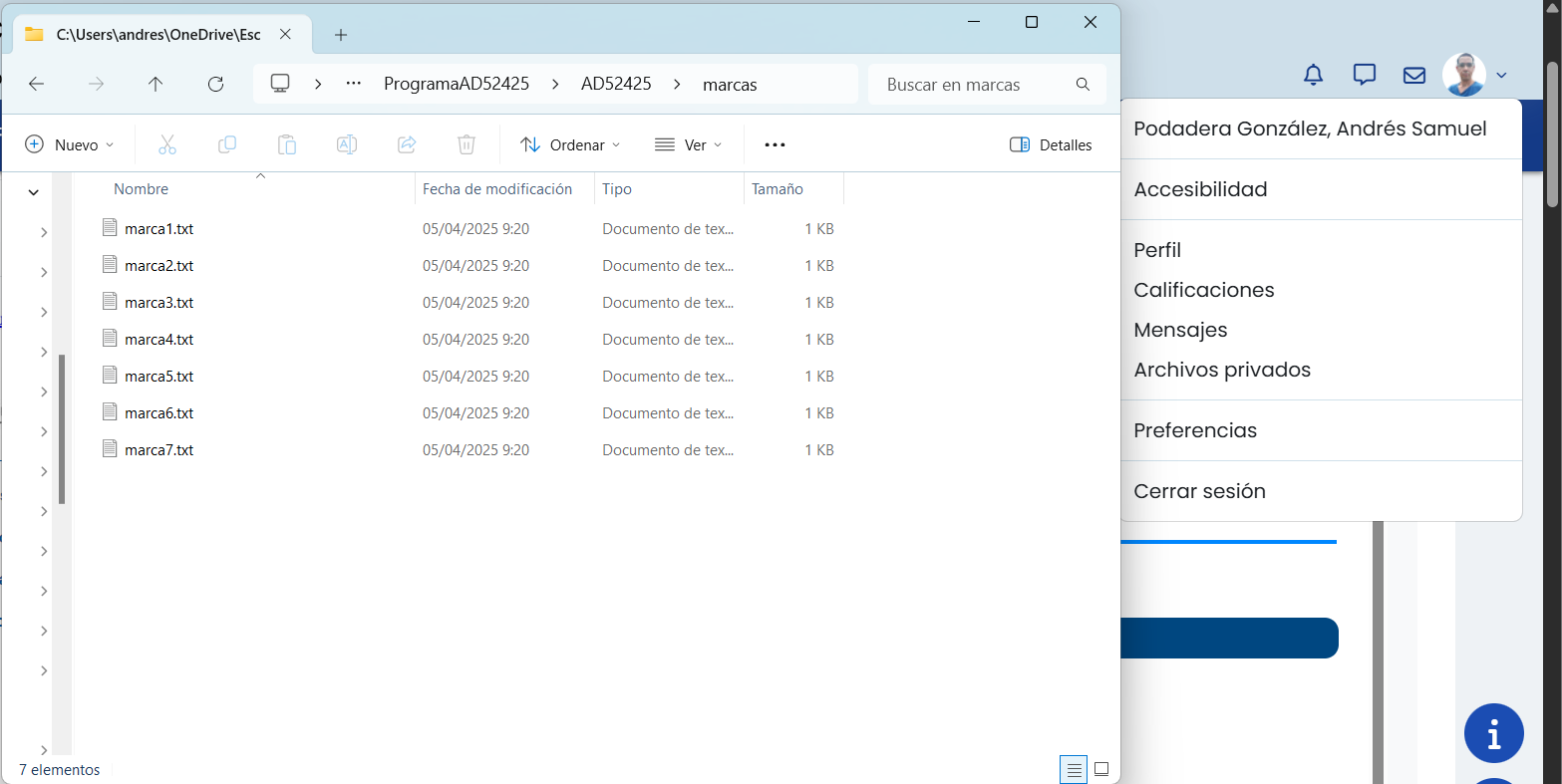


**

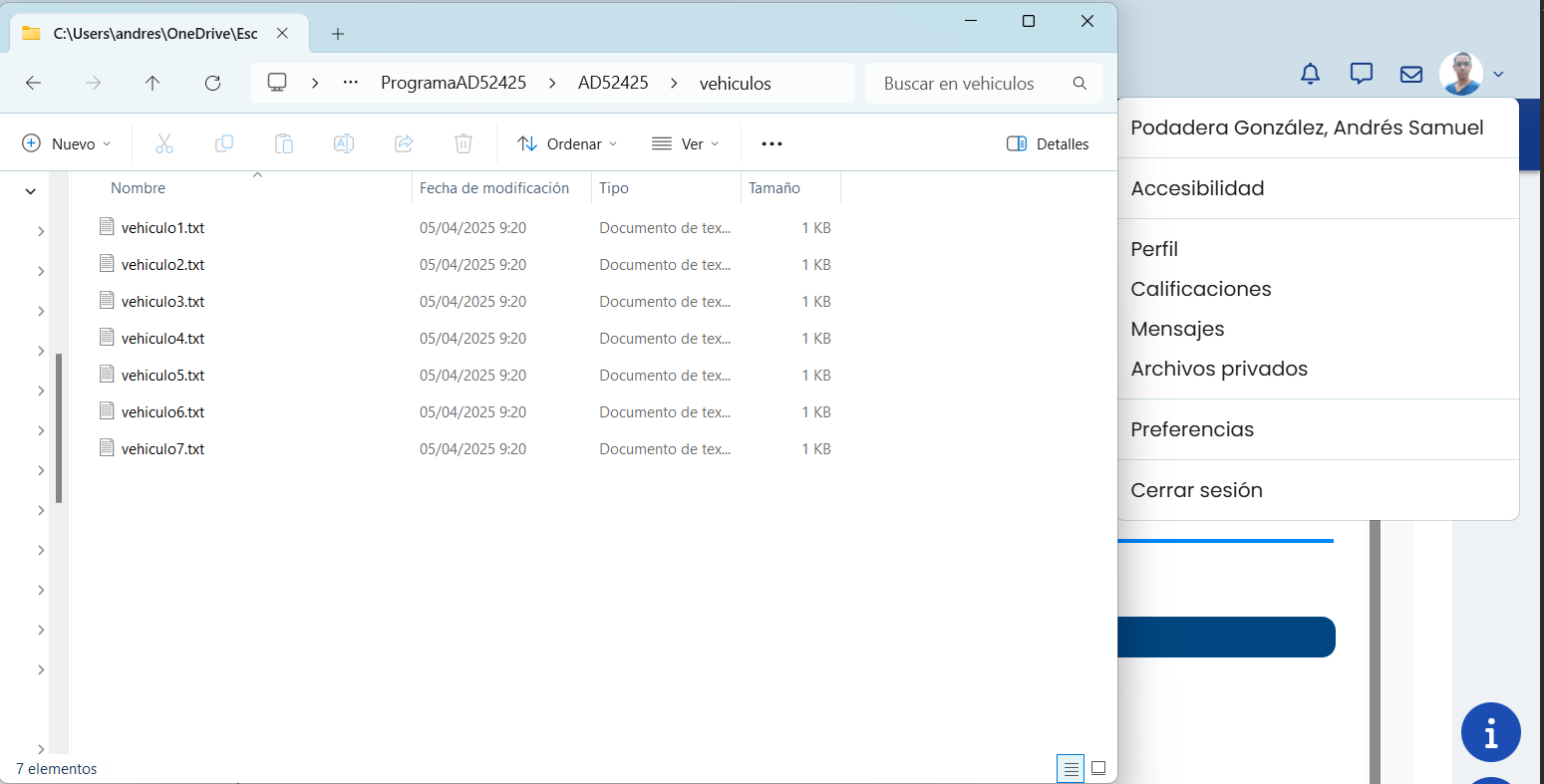
**En este punto nuestra aplicación está conectada y vemos que es capaz de acceder y recuperar la información. Para realizar este ejercicio tan sólo tendremos que pulsar sobre el botón con el texto “Guardar Colecciones en Archivos”, aquí el programa creará el directorio AD52425 sino no existe, las carpetas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



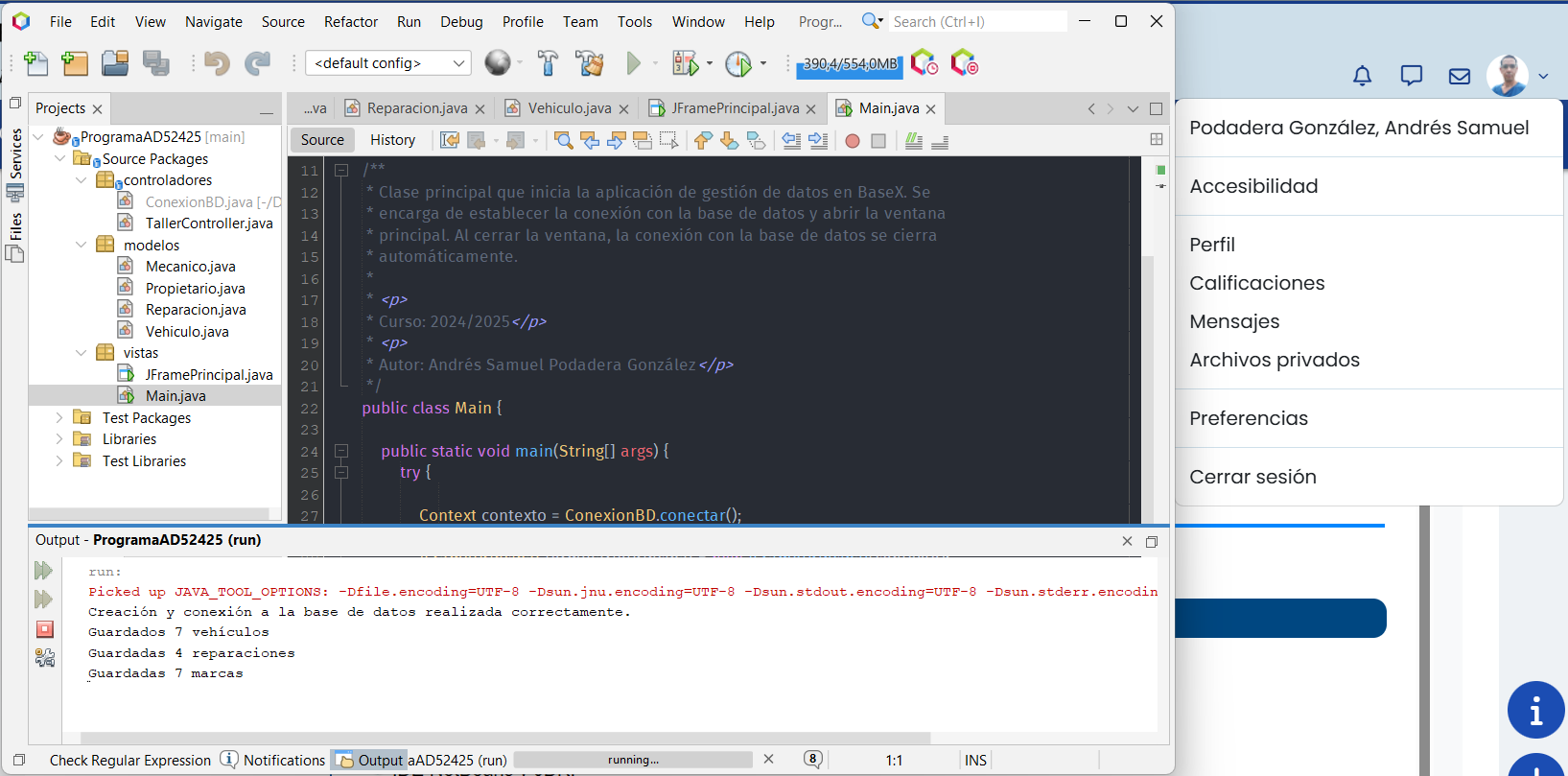
**Y se generan los archivos:**



**

**

**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.**

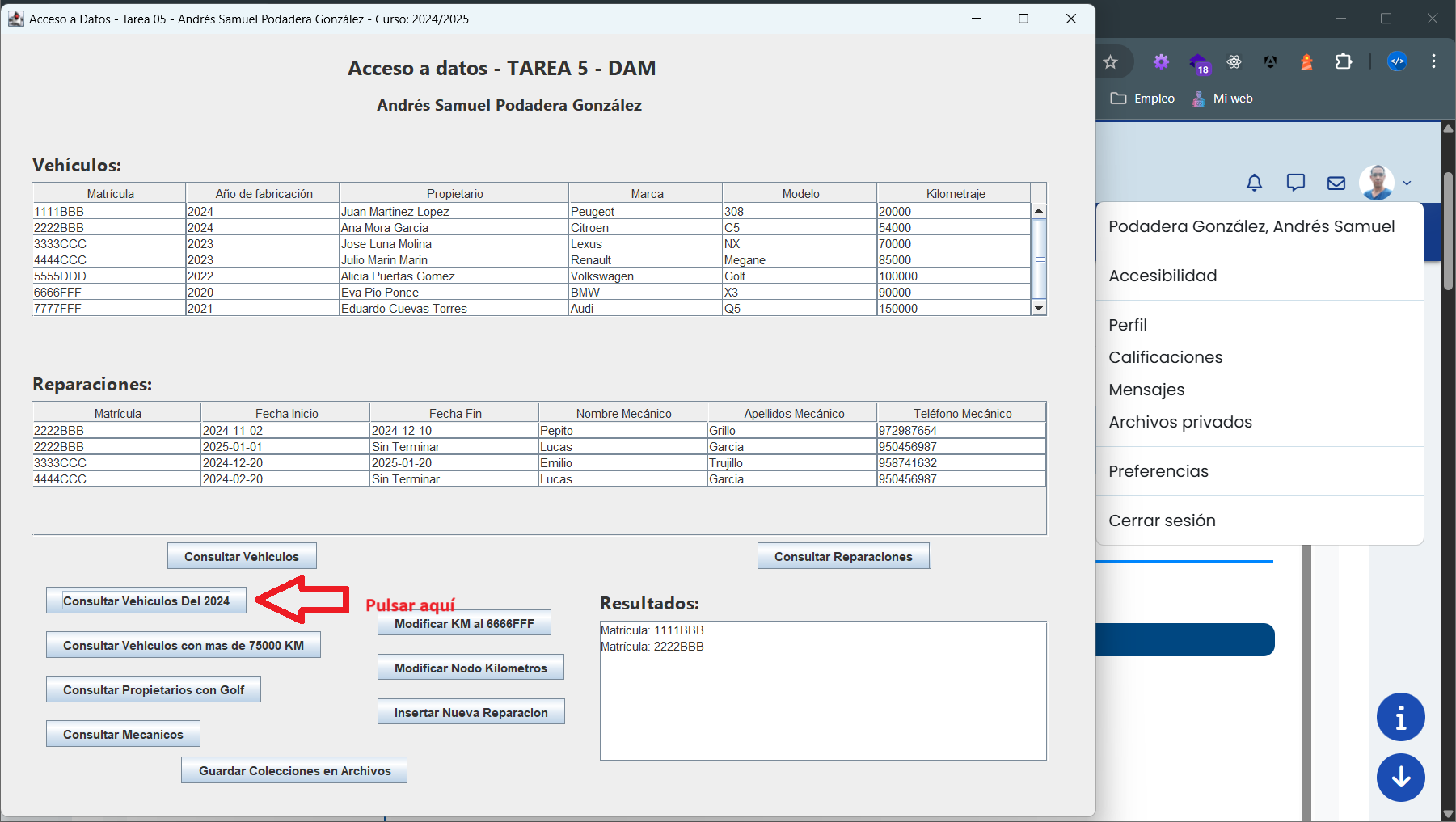
**

# EJERCICIO 2.

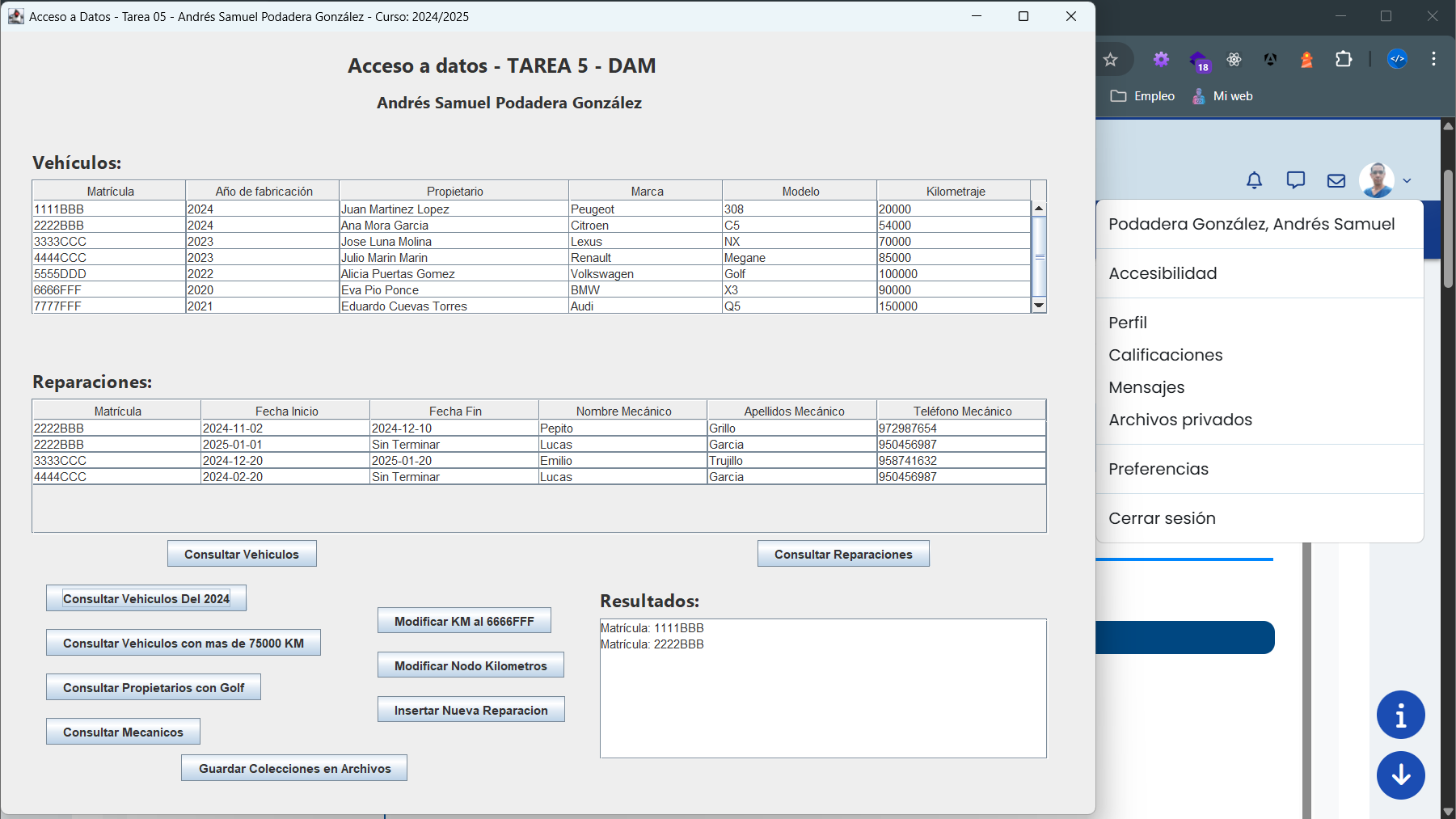
*Realiza estas consultas a la BD taller.xml y ejecútalas desde la aplicación Java. Guardar cada consulta en un script .xq, que llamarás desde tu programa:*

1. *Matrícula de los vehículos cuyo año de fabricación (afabricación) sea el 2024*

**Para realizar esta actividad pulsaremos en el botón:**

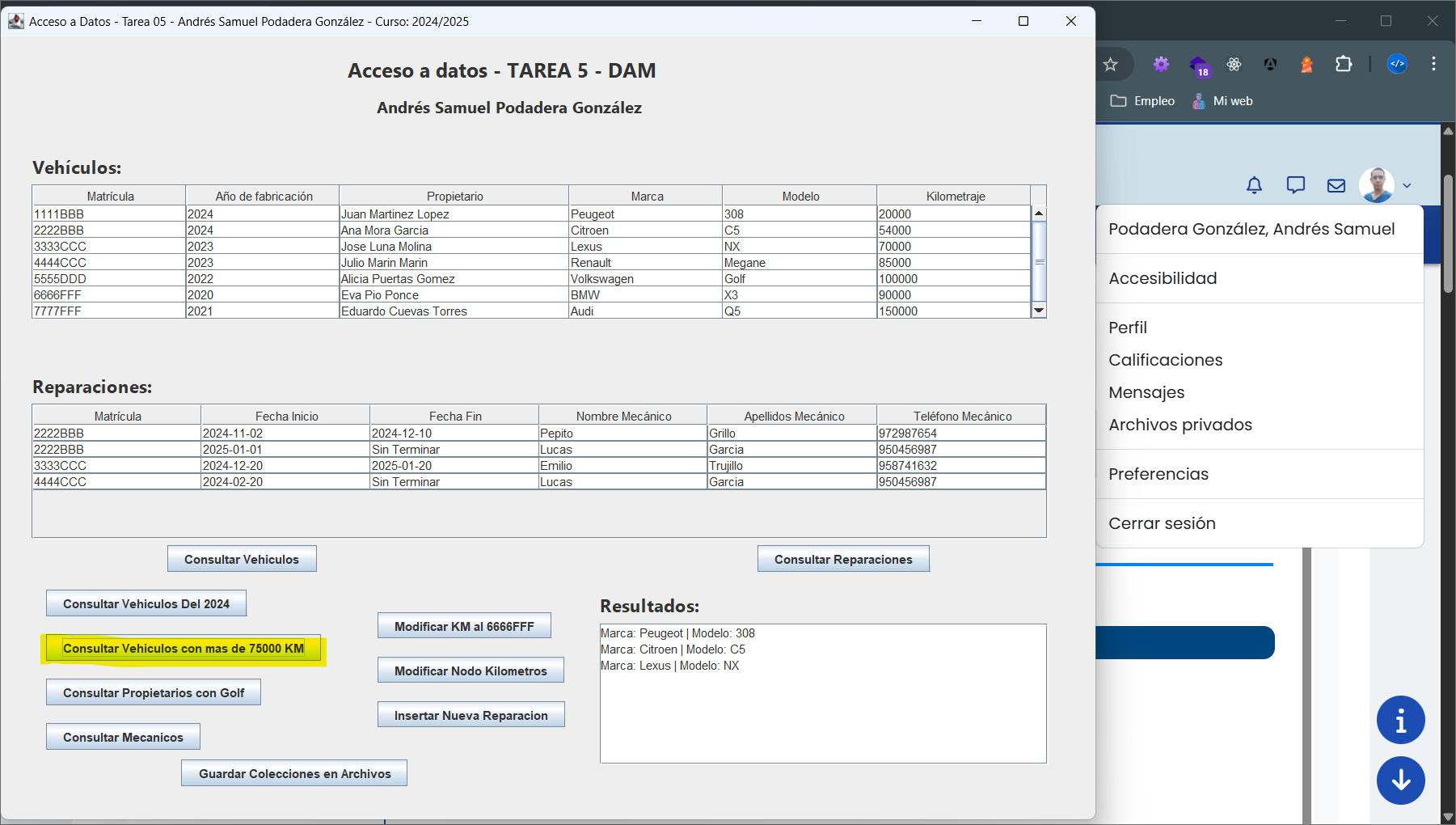


**Y en el cuadro de texto de la derecha, con título “Resultados”, nos aparecen las matrículas de los vehículos:**



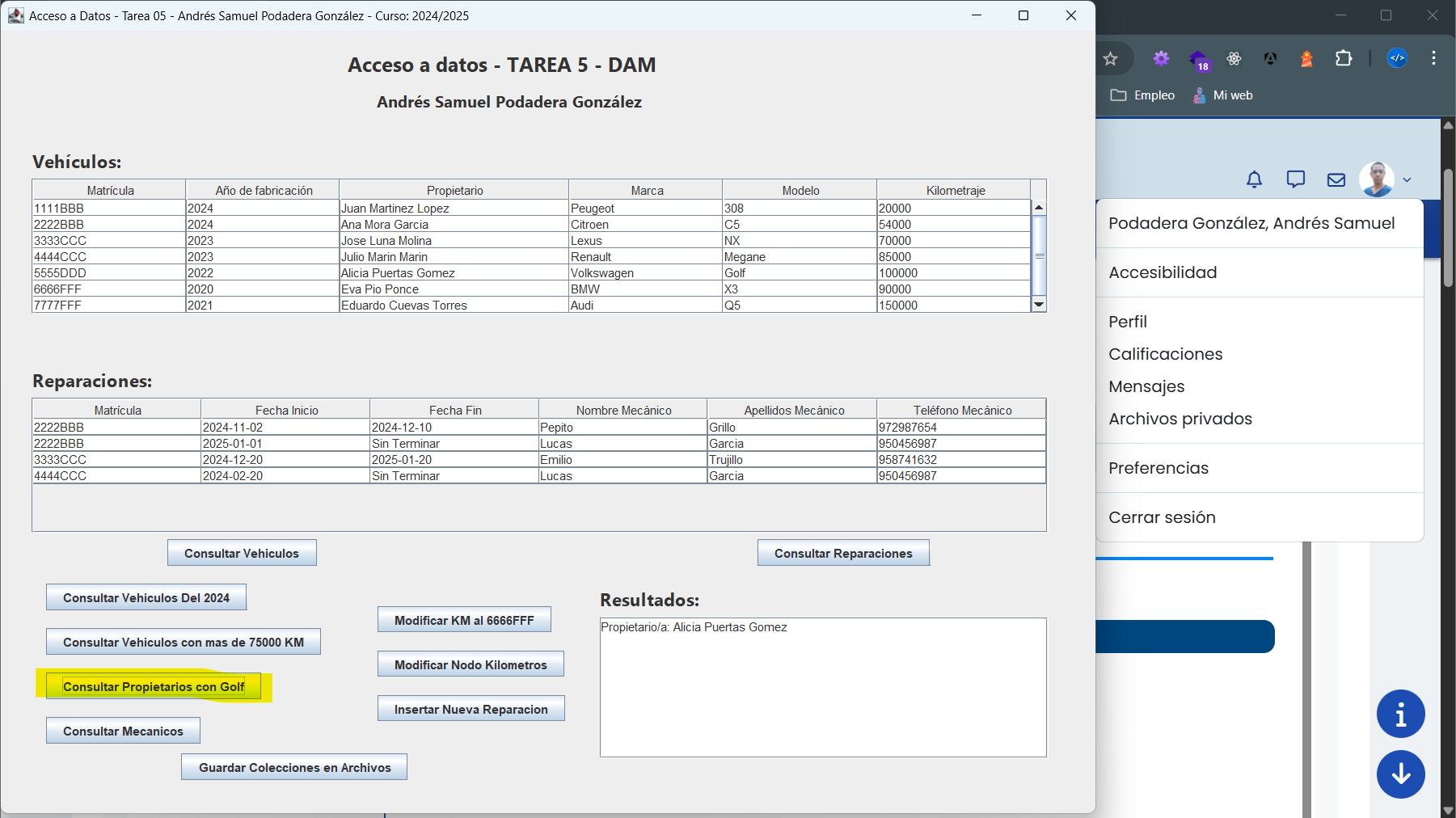
1. *Marca y modelo de los vehículos que tengan menos de 75000 kilómetros.*

**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:**

**

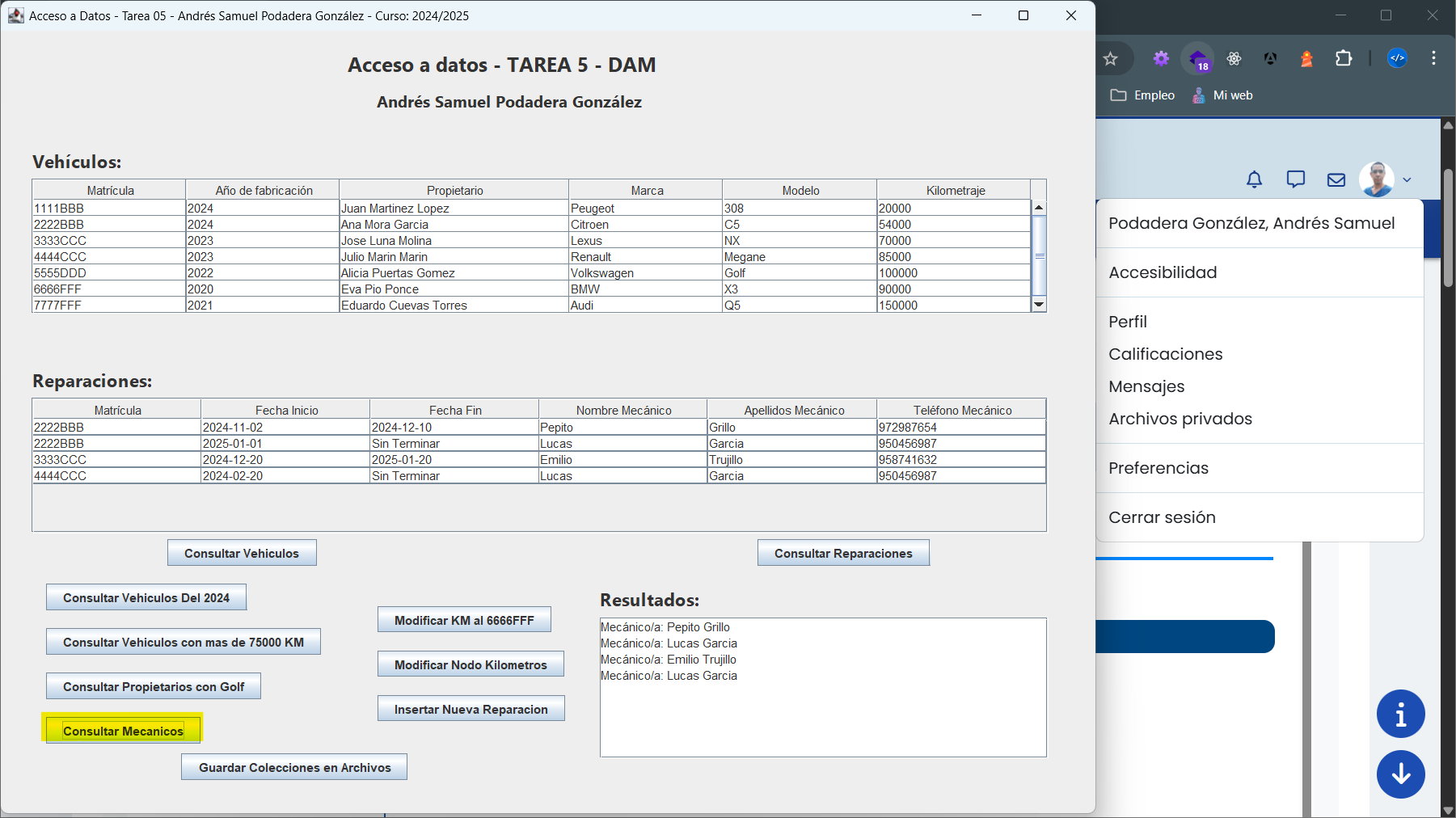
1. *Nombre y apellidos de los propietarios que tengan un Golf.*

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

**

1. *Nombre y apellidos de todos los mecánicos de vehículos.*

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

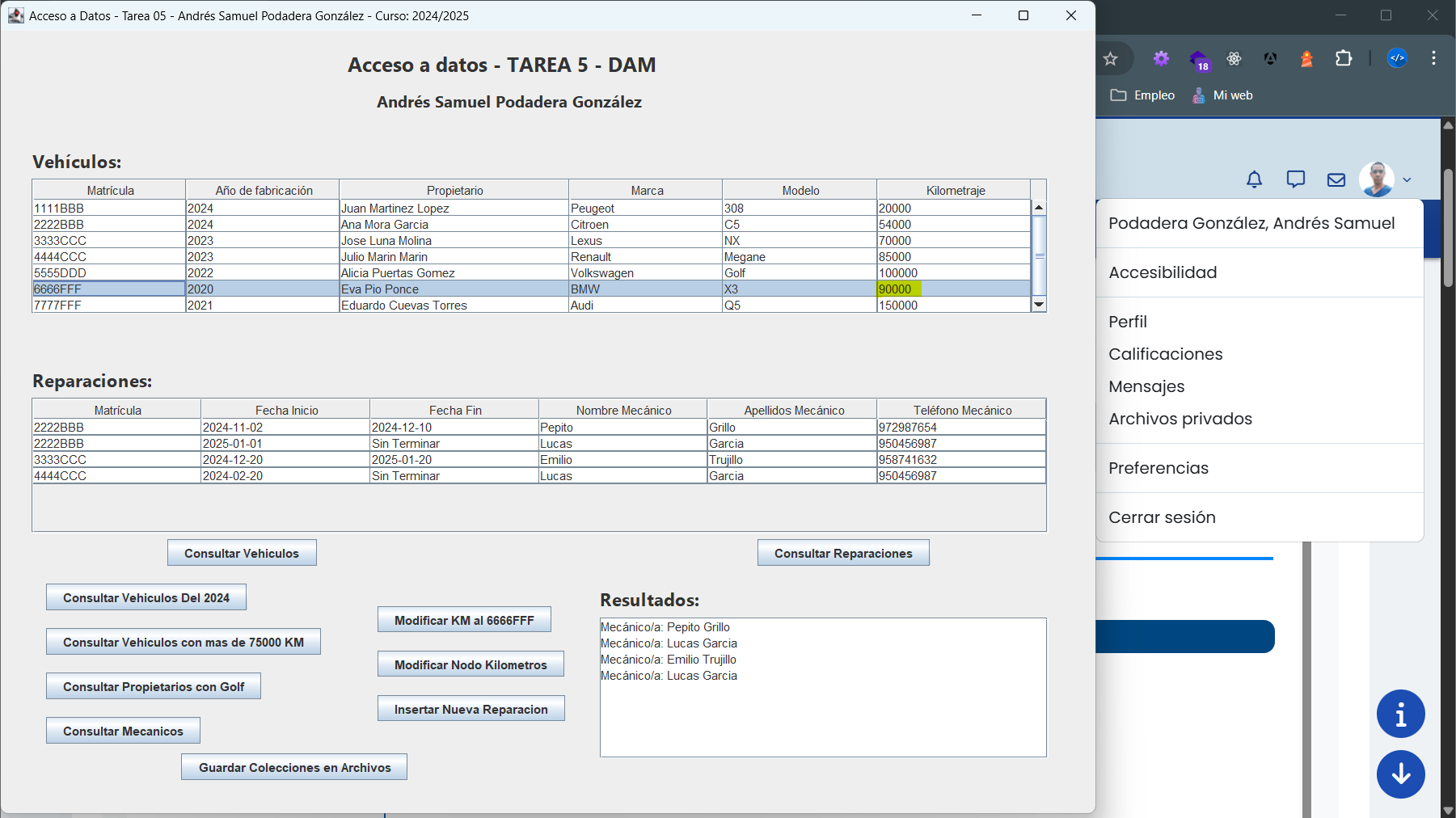
**

# EJERCICIO 3.

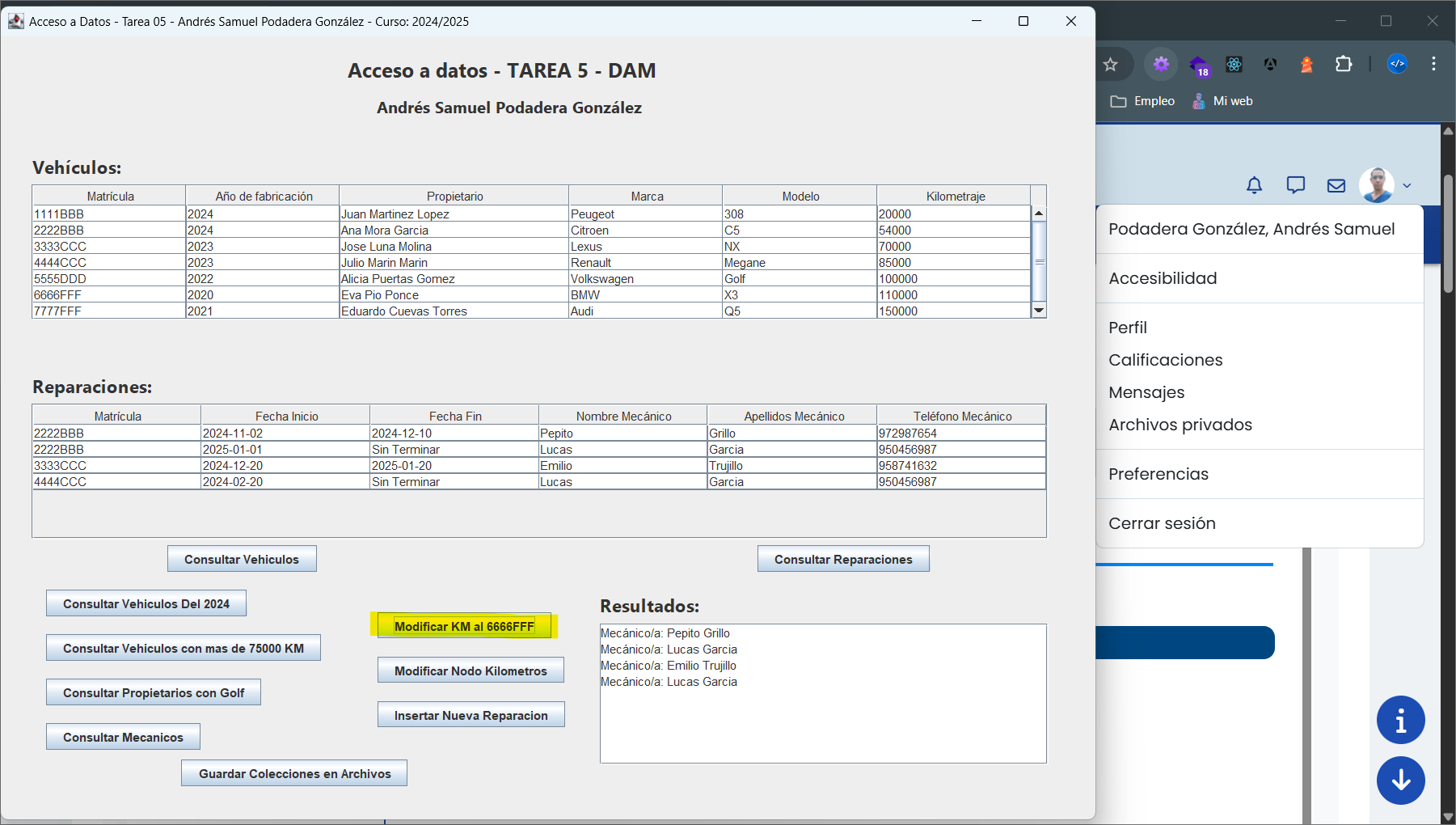
*Crea estas consultas de actualización .xq para la BD taller.xml y comprueba su resultado desde la aplicación Java:*

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

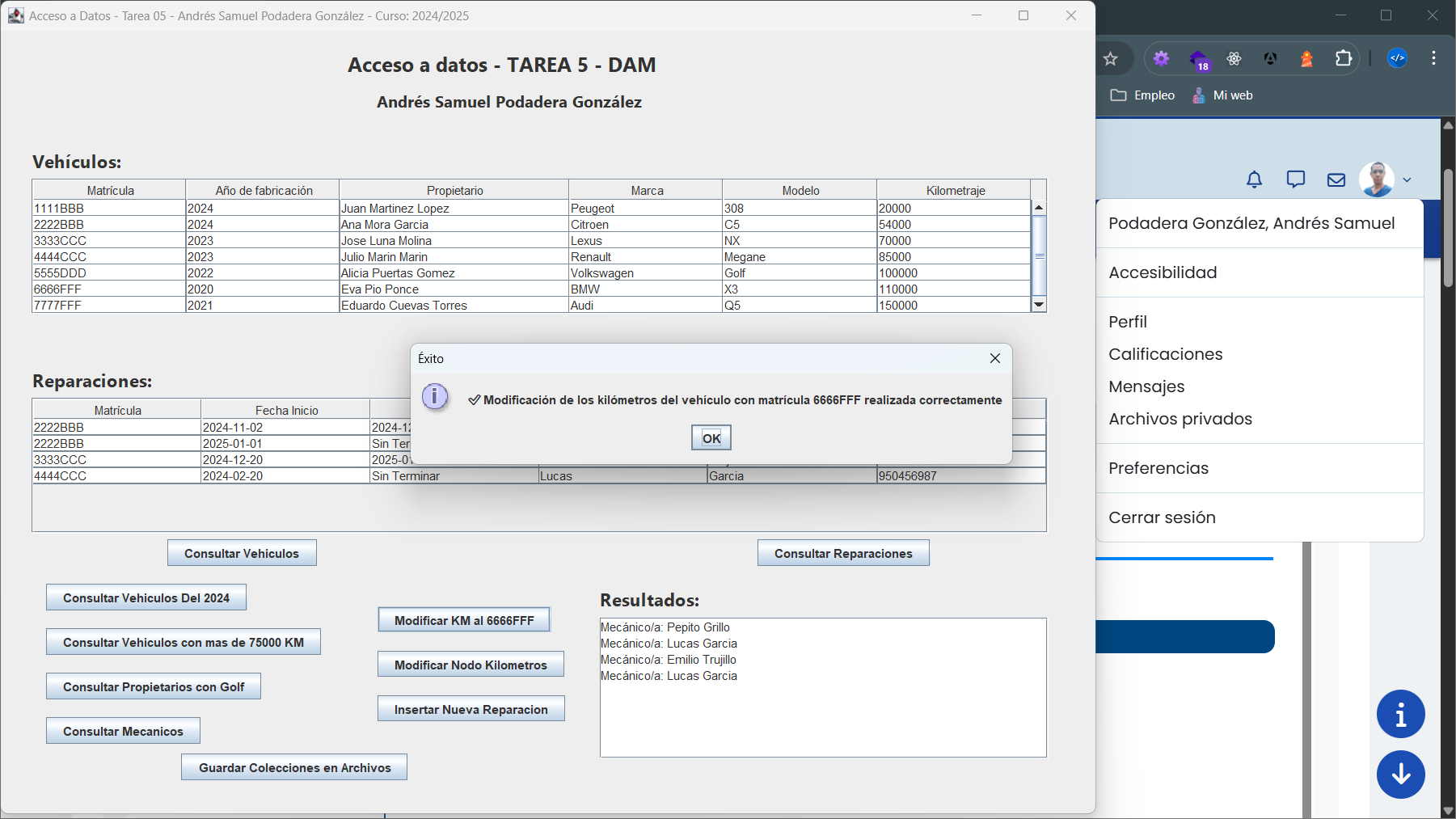
**Para este apartado primero observamos en la tabla que el registro contiene una cifra de kilómetros de 90000.**

****

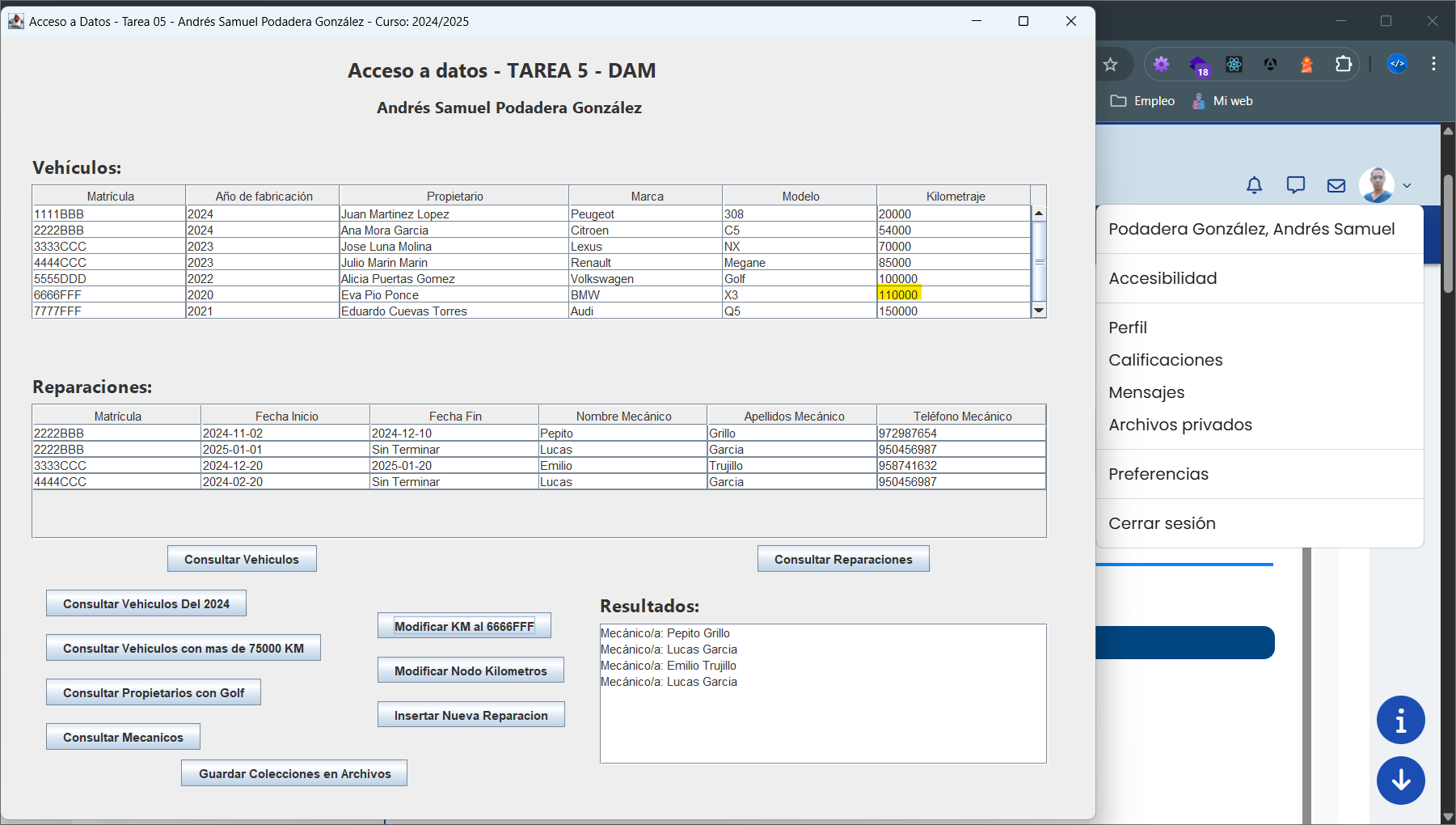
**Al pulsar en el botón con texto “Modificar KM al 6666FFF”:**

****

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

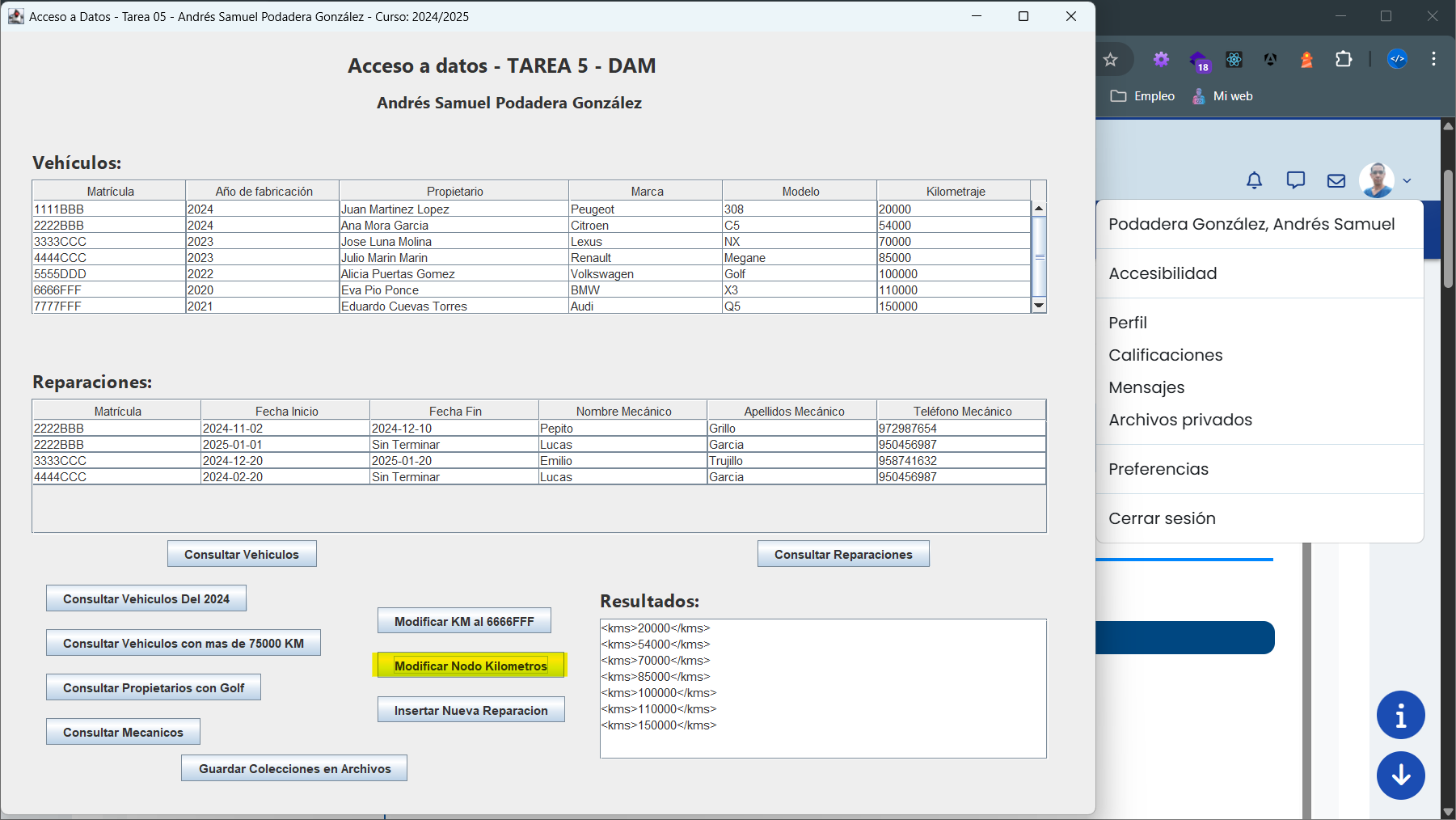
****

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

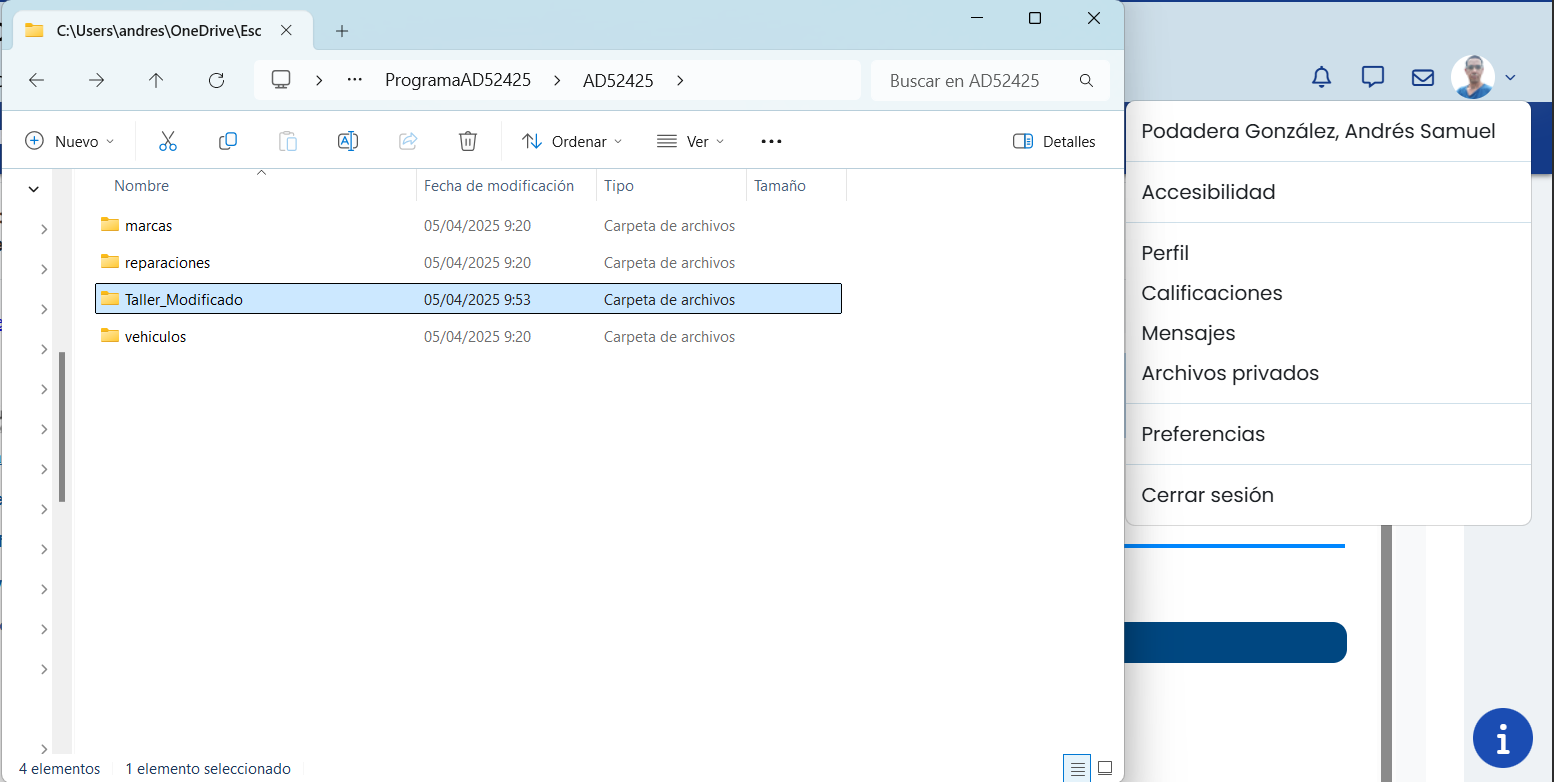
****

1. *Modificar el nombre del nodo kilómetros de cada vehículo de la colección vehículos por kms.*

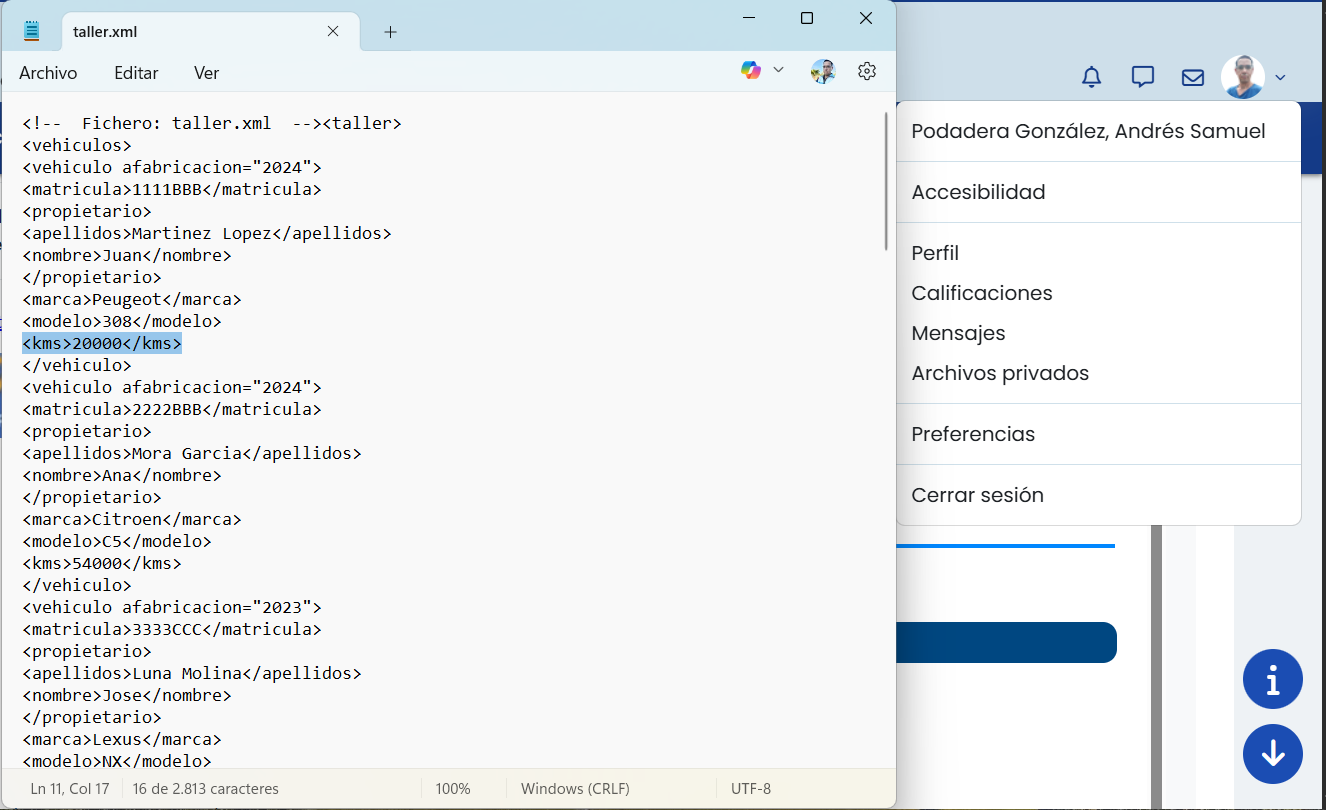
**Para este apartado, pulsaremos sobre el botón con texto “Modificar Nodo Kilómetros” y en los resultados nos aparecerán los nodos modificados.**

**

**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.**

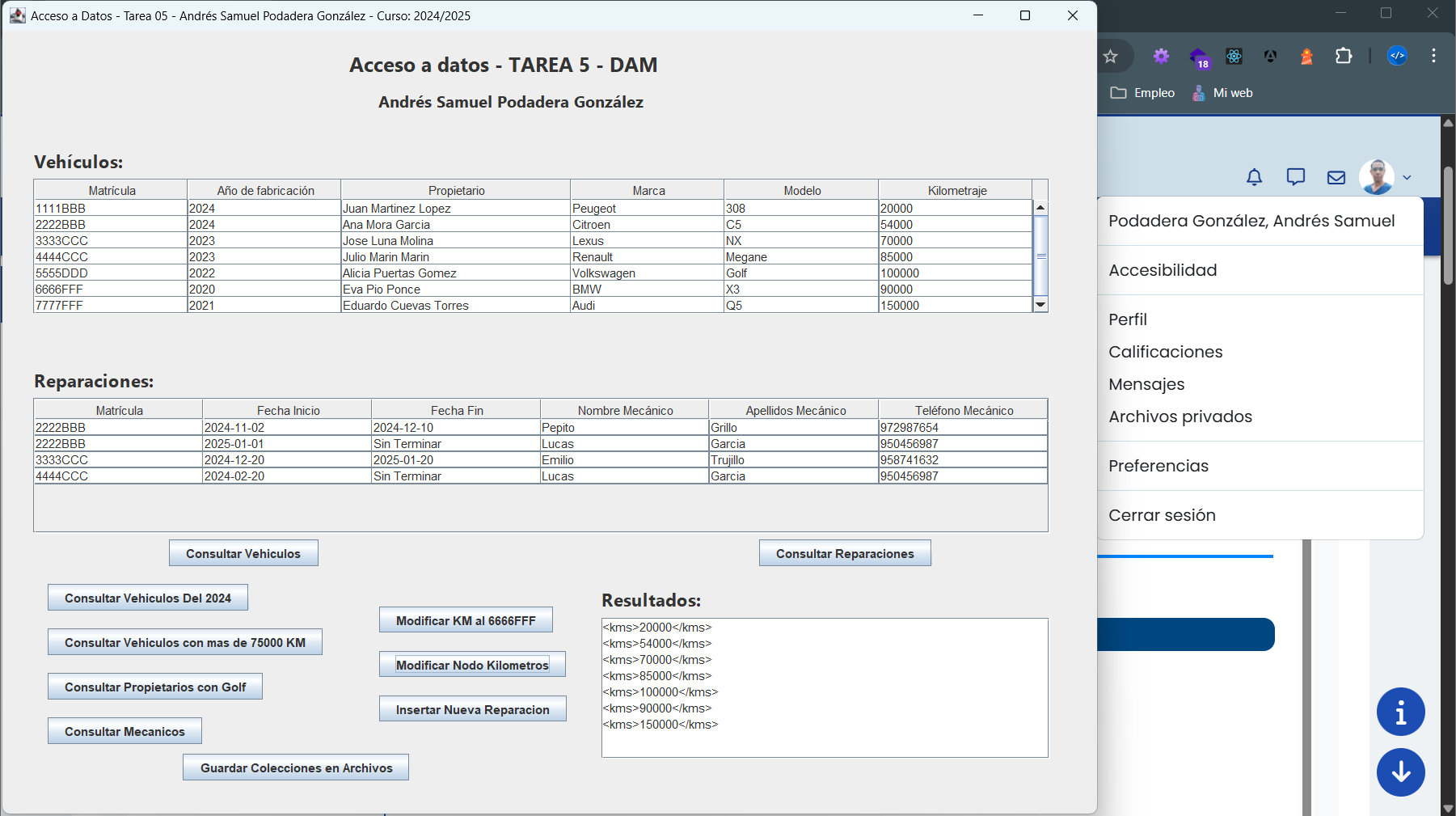
****

**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>.**

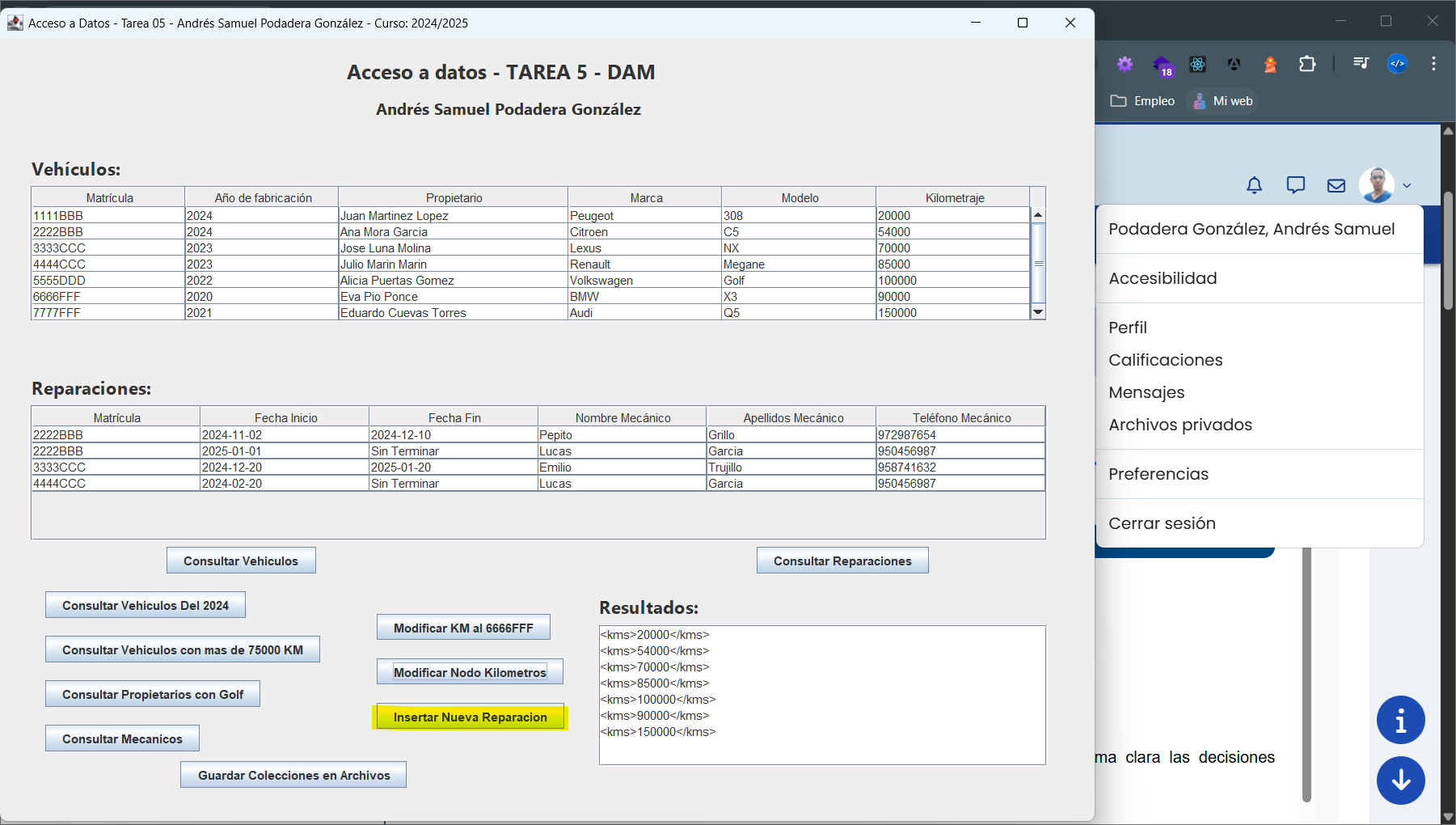
**

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

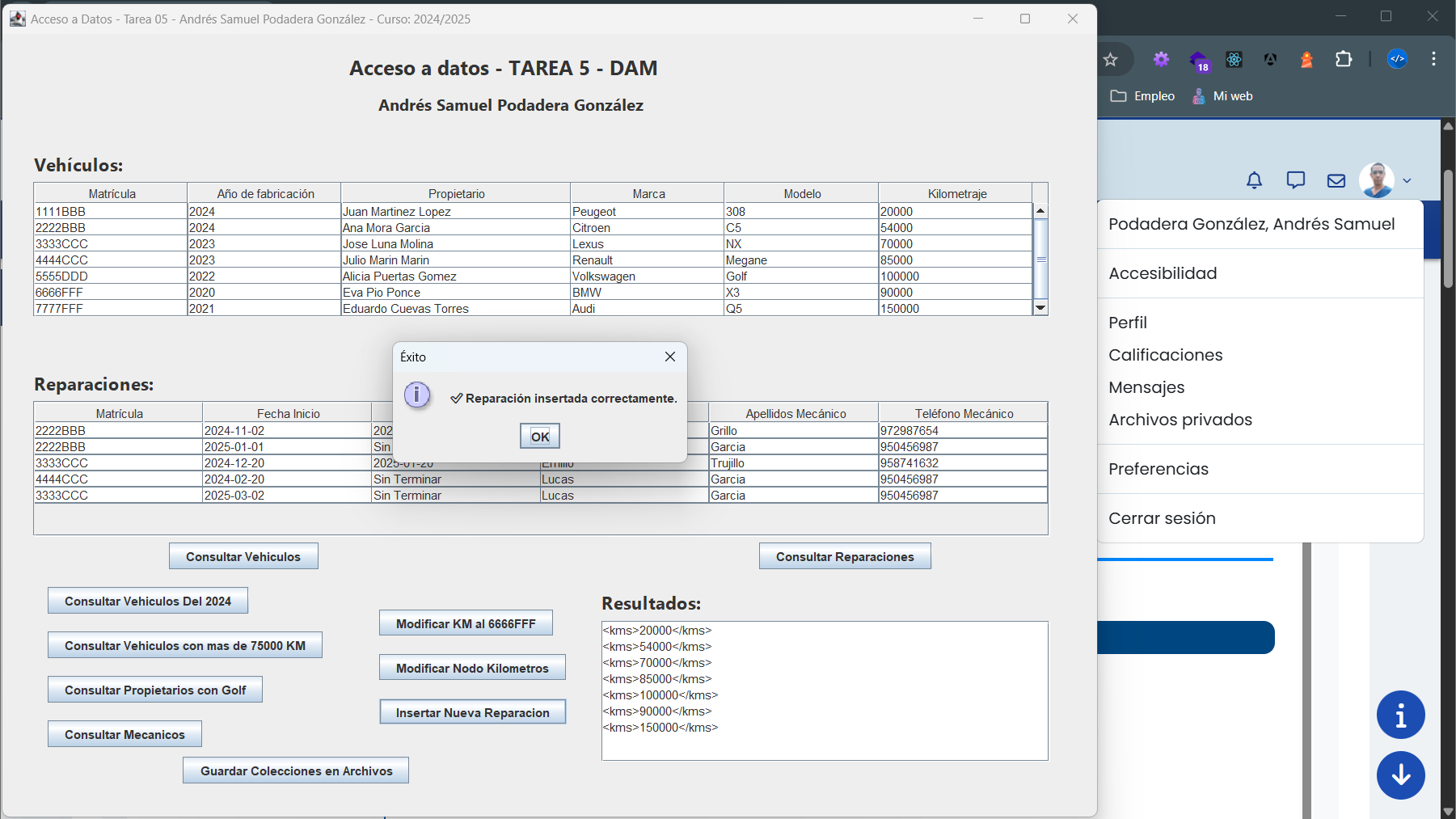
**Comprobamos que la tabla de reparaciones contiene 4 registros:**



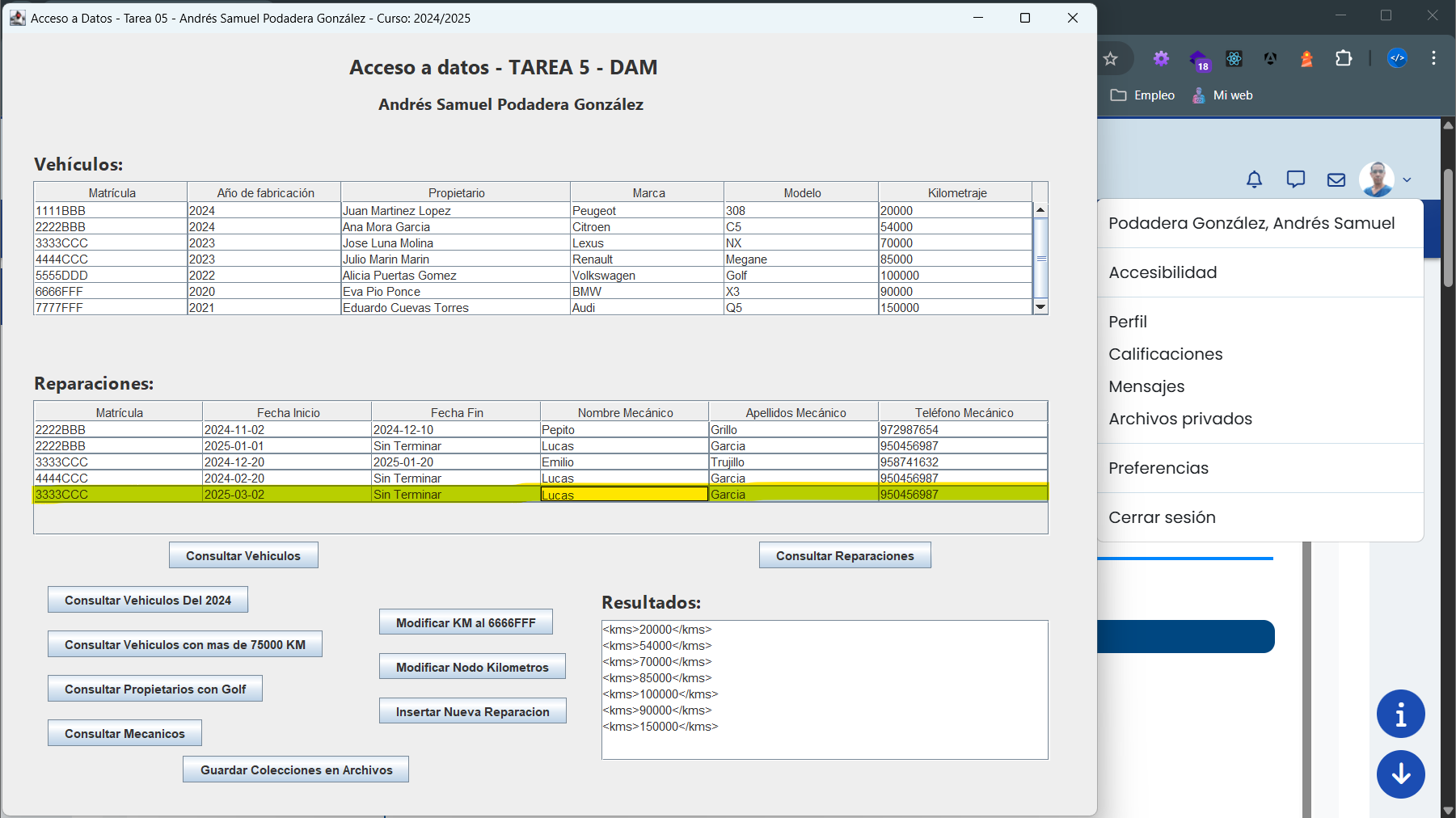
**Pulsamos en el botón con texto “Insertar Nueva Reparacion”:**



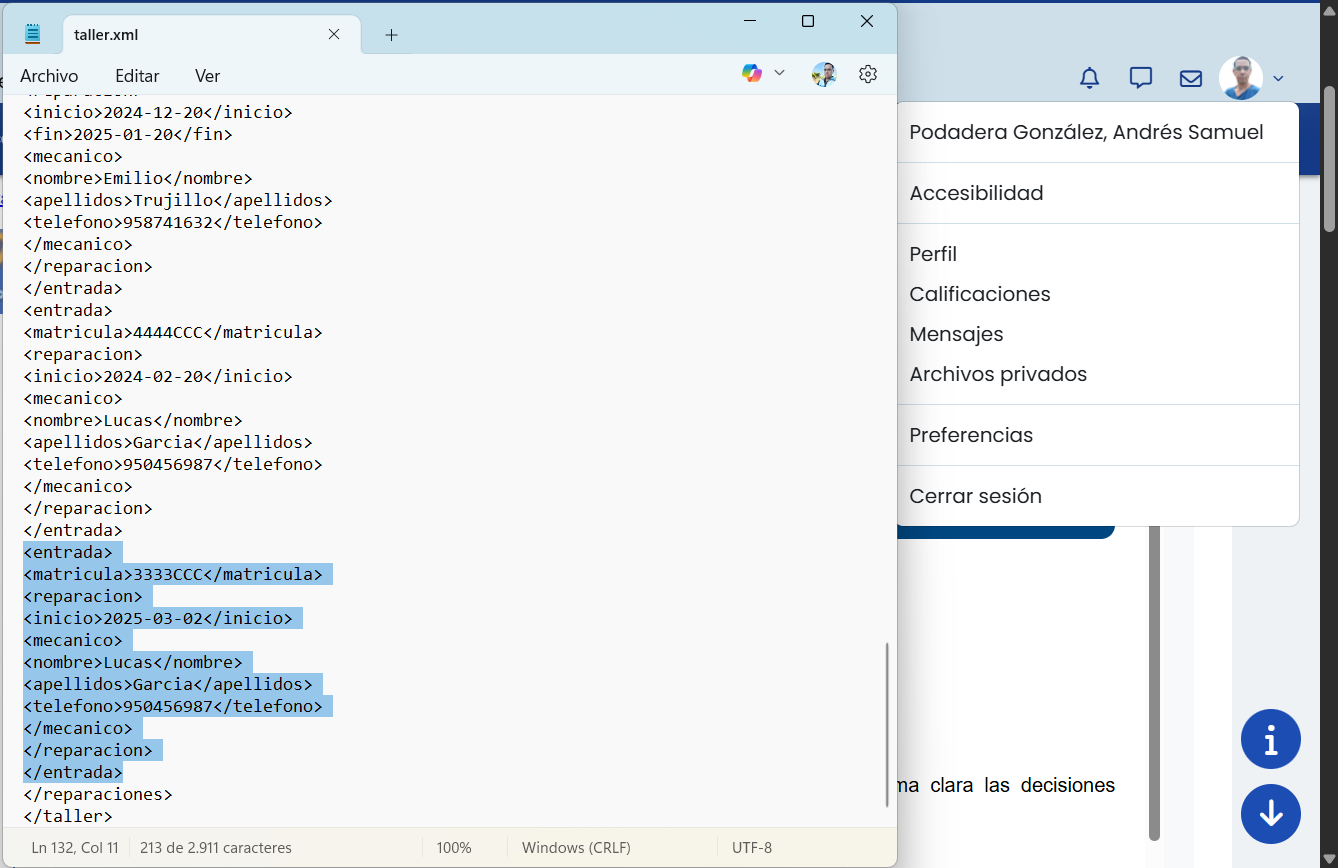
**Y la interfaz nos mostrará un mensaje de aviso de operación realizada correctamente:**



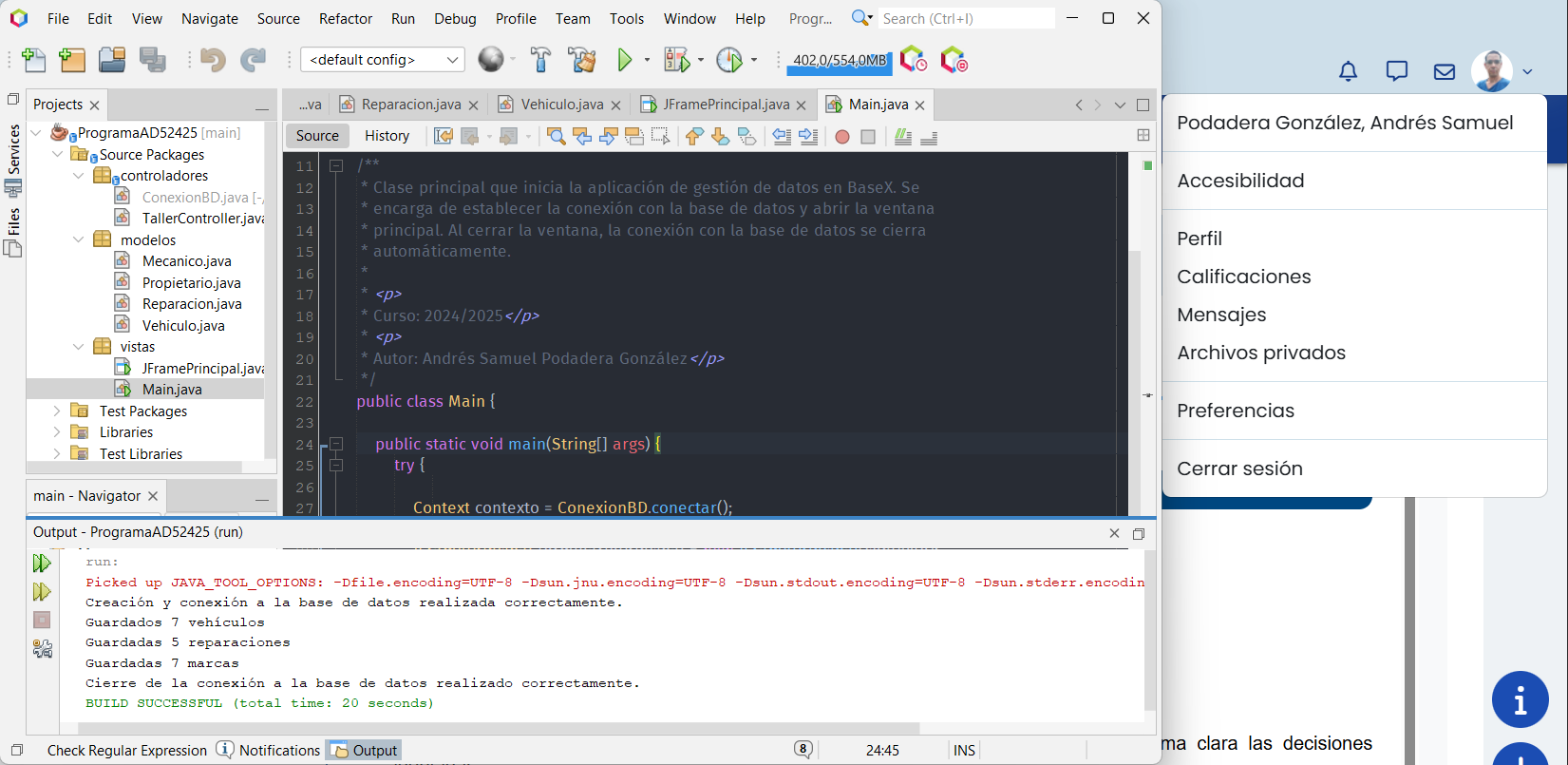
**Y si observamos en la tabla de reparaciones veremos que se ha insertado una nueva fila con los datos solicitados:**



**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:**

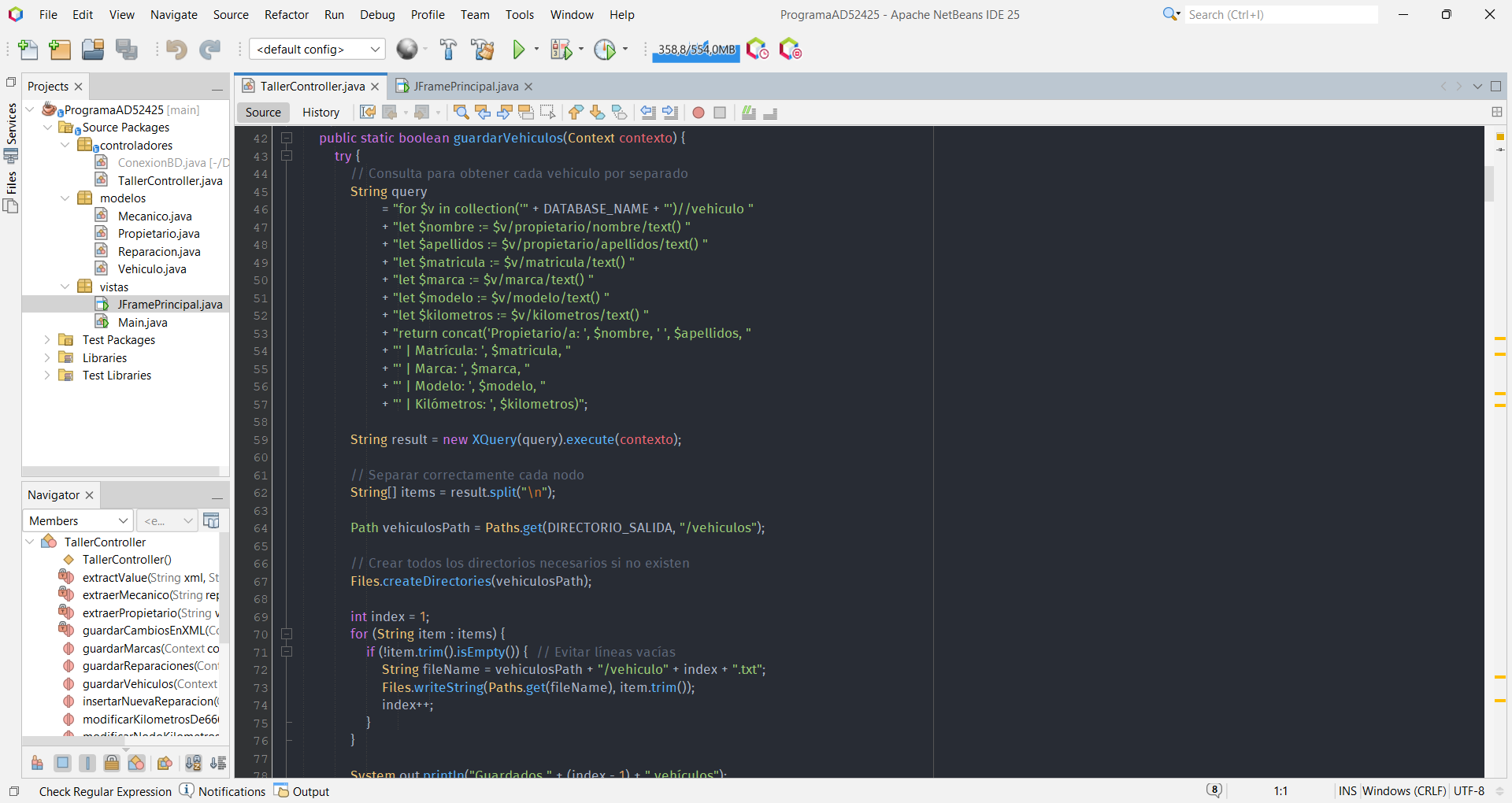


**Cuando cerramos la aplicación, se nos muestra un mensaje de “Conexión cerrada” en la consola de NetBeans:**



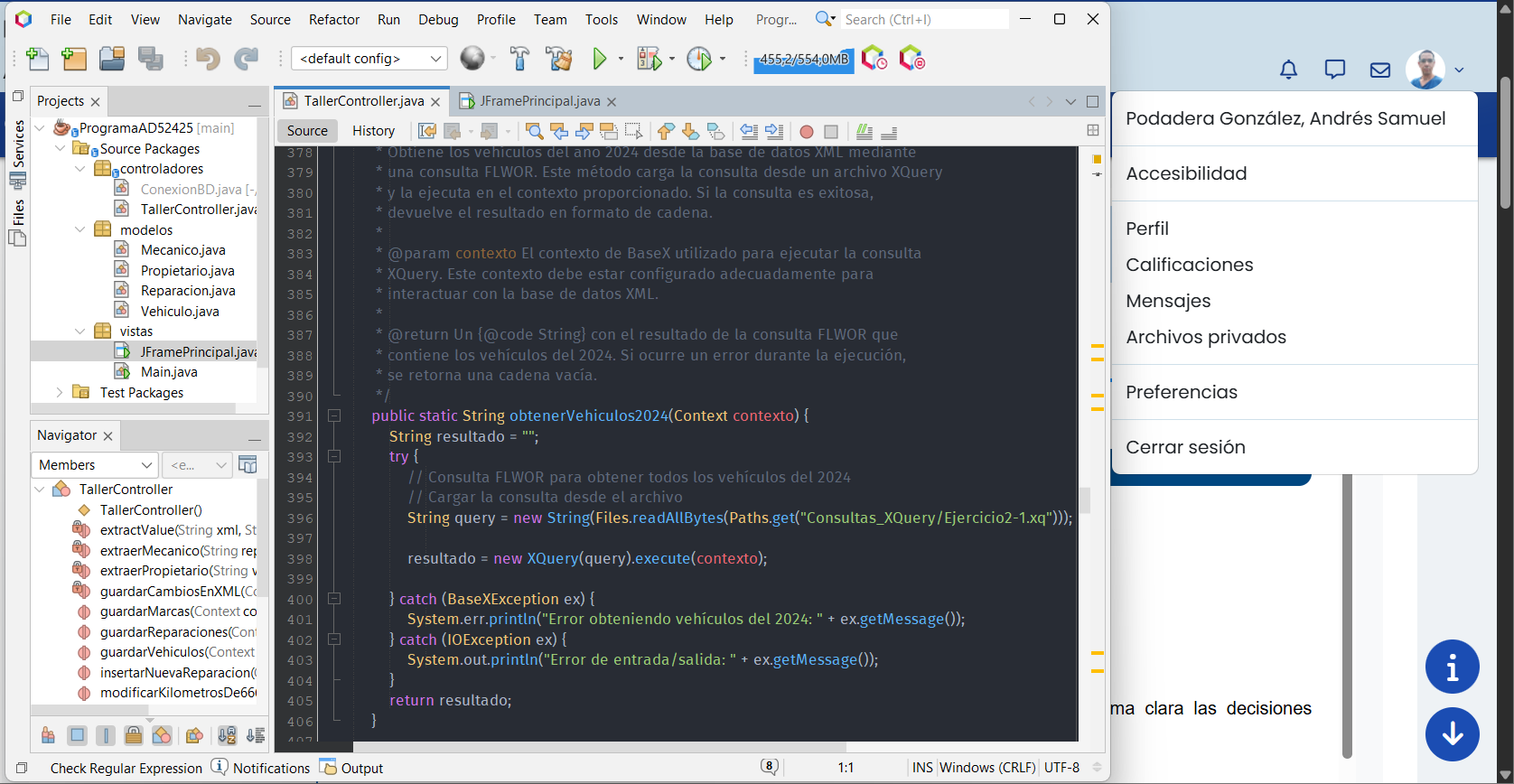
# Aclaraciones y Explicaciones Generales de la tarea

**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:**

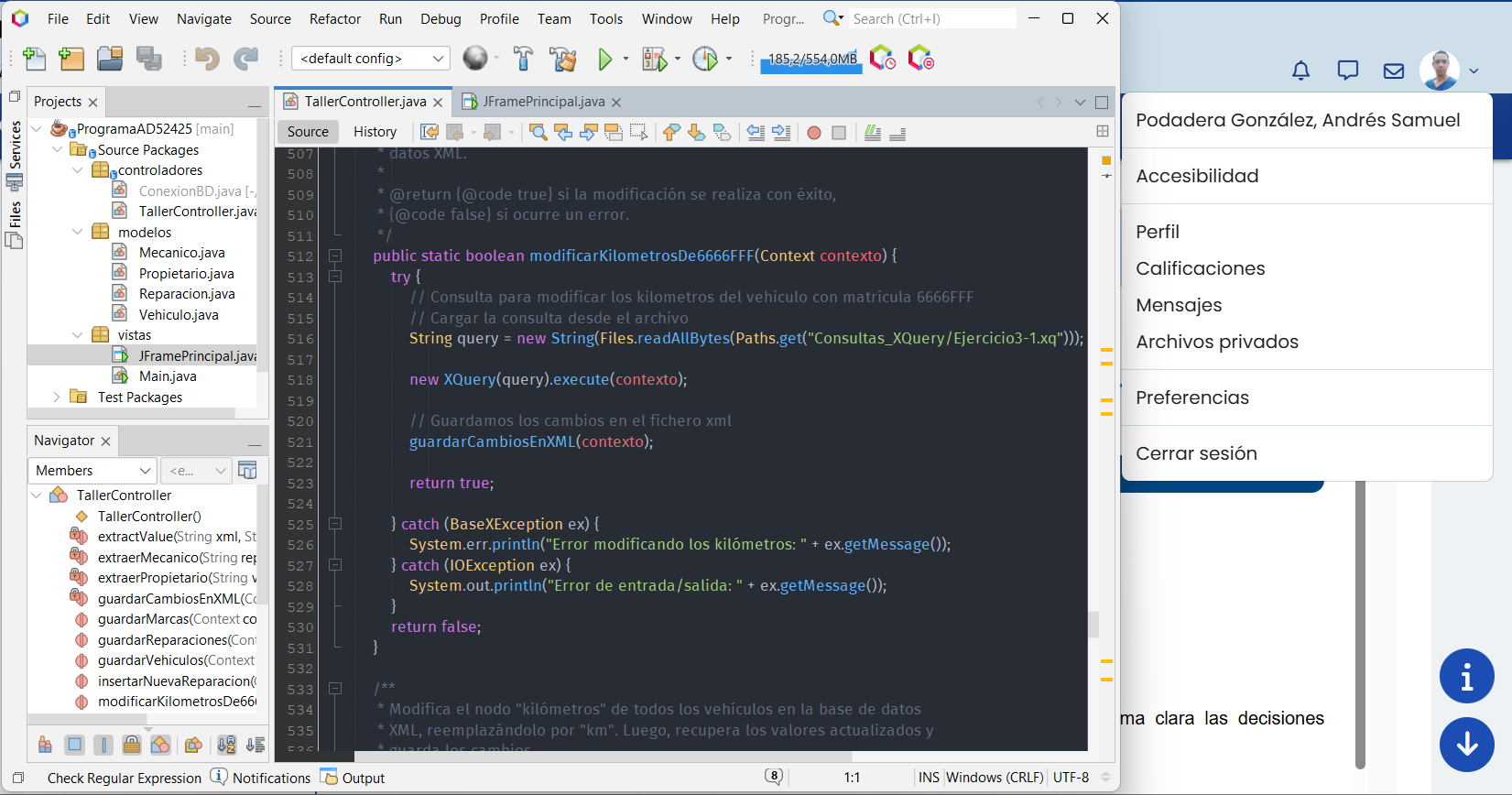


**Luego esas cadenas serán insertadas en los archivos txt con el nombre que se piden.**

**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.**



**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.**

****

**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.**