



---

# PROYECTO MULTIMODULAR PROGRAMACIÓN

---

Marco Antonio Morenilla Alonso.



31 DE MARZO DE 2024

UAX

CEO: Raúl Albiol Salguero

# Índice

Introducción.....	2
Conexión a la BBDD.....	2
Escritura de documentos.....	5
Interacción con el usuario y funcionamiento del programa.....	7
Demo.....	9
Resultados.....	10

## Introducción

Se pide realizar una aplicación que se conecte a una BBDD y haga tres consultas sobre esta exportando los resultados en diferentes formatos.

La BBDD y las consultas elegidas son las propuestas en el enunciado del proyecto.

Para la inserción de datos en la BBDD se utiliza ChatGPT pero solamente para esto, el resto es cosecha propia.

Se adjunta en el mismo proyecto un archivo Leeme.txt con instrucciones relativas a su utilización.

Por ello se decide realizar de la siguiente manera:

Se crea una clase Main con Scanner para poder interactuar con el usuario e ir creando dichas consultas a través de menús de opciones que tienen que resolverse obligatoriamente.

Se crea una clase Tienda donde se almacenan datos de una tienda con un nombre ficticio “Tu tienda del comic” esta tienda es la que almacena los datos de comics en la BBDD por lo que tiene un atributo que es un ArrayList de Comics para agrupar los resultados de la consulta.

Se crea una clase Comic que representa los comics de la BBDD.

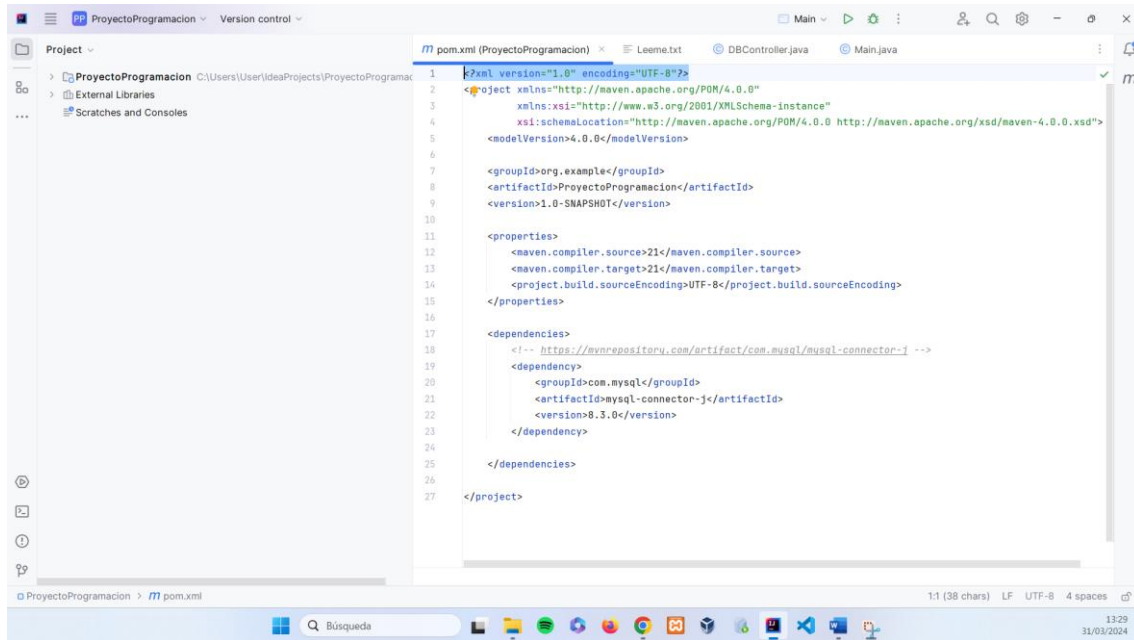
Se crea una clase DBController donde se irán definiendo los métodos para abrir conexión con la BBDD, cerrar conexión y realizar búsquedas.

Estos resultados se agrupan en un objeto resulset y se guardan en un ArrayList de Comic que será el que utilizemos para crear la Tienda con el resultado de la búsqueda con un método getter del ArrayList.

## Conexión a la BBDD

Para establecer la conexión:

1. Necesitamos tener las dependencias bien configuradas en el archivo pom.xml

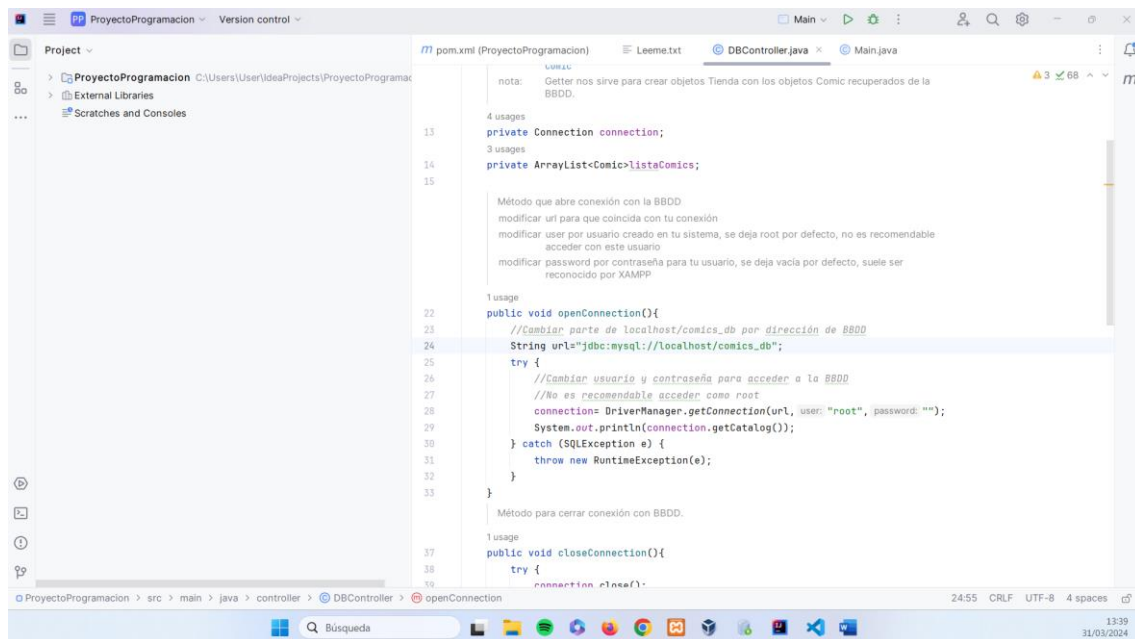


## 2. Necesitamos una dirección compuesta por:

- jdbc:mysql:// : hace referencia al controlador que se utiliza para conectarse a la BBDD Java Data Base Controller y al tipo de artefacto que es necesario mysql en este caso.
- Localhost: en este caso como tengo configurado los puertos 80 en apache y 3306 en phpMyadmin es el nombre equivalente a escribir 127.0.0.1:3306, si cambiamos el puerto 3306 habría que escribir 127.0.0.1:(número de puerto).
- Comics\_db: es el nombre que le he dado a la BBDD.

## 3. Un objeto connection que ejecute un método que se conecte a la BBDD con esa dirección seguido del nombre de usuario y contraseña.

- Para el pantallazo he dejado usuario root y password vacía que funciona en XAMPP pero para conectarme durante el programa he usado un usuario y password creados en phpMyAdmin.



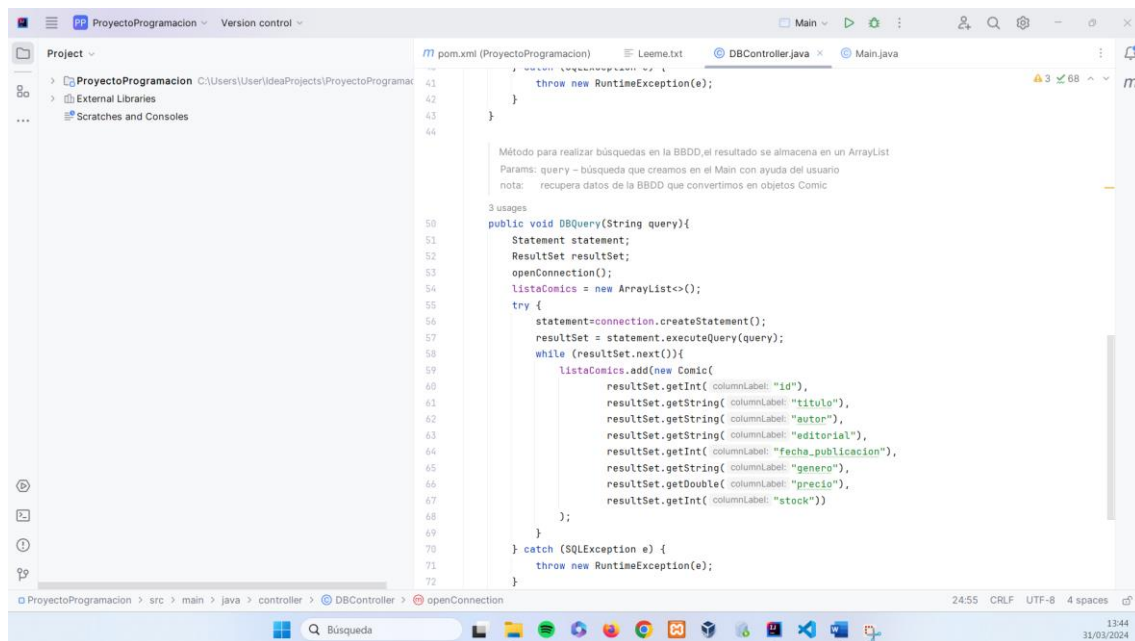
Una vez establecida la conexión, se crea método para cerrarla a través del mismo objeto connection con el método close.

El método para realizar las consultas necesita de objeto statement y resultSet:

- Se decide utilizar statement porque le paso siempre un string y el motor de BBDD no necesita interpretar datos, en caso de que lo necesitase para opciones de insertar, modificar o eliminar, utilizaría PreparedStatement, que delega en el programa la interpretación de los datos no llegando a conectar si se produce algún error con estos.

Una vez realizada la consulta a la BBDD el resultado se recoge en un objeto tipo ResultSet, que se recorre con un while y mientras tenga registros a su derecha se ejecuta.

Por cada ejecución, crea un objeto Comic y lo añade al ArrayList de la clase.



Podría ser más eficiente el programa accediendo a estos registros para escribir los archivos, pero al delegarlo a la clase Tienda se hace mucho más escalable, ya que podría reutilizar el objeto DBController para cada tienda e ir agrupando los resultados específicos en cada una de ellas, por eso se decide hacer así.

## Escritura de documentos

Los métodos para exportar los documentos en diferentes formatos se escriben como static para acceder a ellos sin necesidad de un objeto Main ni crear otra clase con estos métodos de escritura.

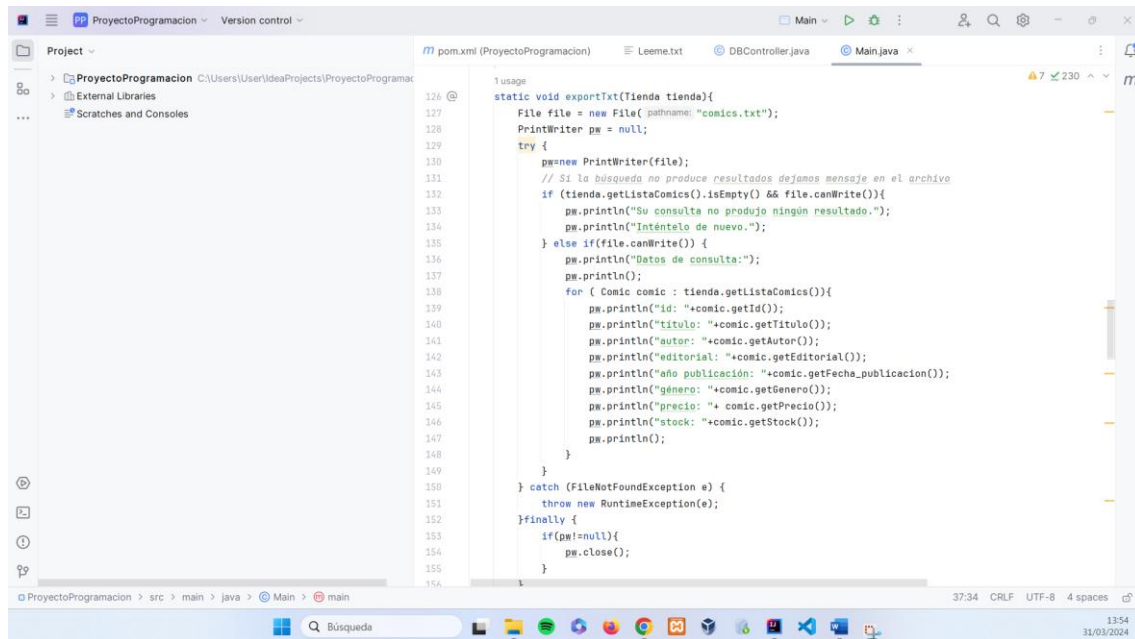
Se utiliza un método para cada tipo de documento para hacer el programa modular y más sencillo.

Para decidir el formato, se realiza a través de un menú de opciones también de obligada resolución, en definitiva, no se puede salir del programa si no realizas la consulta y si no escribes un documento, es mi toque personal y perverso.

Todos los métodos reciben un objeto de tipo tienda por parámetro del que sacan la información para escribir, se crean sin append, para que cada vez que se ejecute el programa se borre el contenido anterior y se escriba el nuevo.

Además, se les añade una comprobación par ver si el programa ha resuelto la consulta y ha obtenido resultados, en caso de no tenerlos, se escribe en el documento un aviso de la consulta no ha generado resultados comprobando que el ArrayList está vacío.

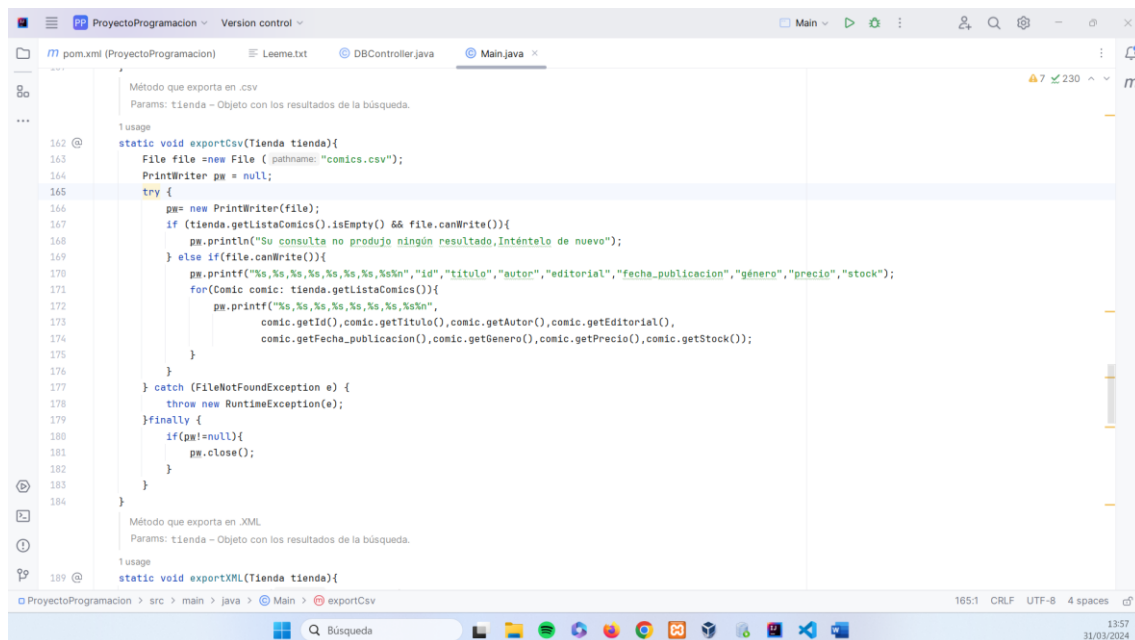
1. Exportar a txt:



The screenshot shows the IntelliJ IDEA IDE with the 'Main.java' file open. The code defines a static method `exportTxt` that takes a `Tienda` object as a parameter. It attempts to write the list of comics from the store to a file named 'comics.txt'. The method includes a try-catch block to handle `FileNotFoundException` and a finally block to close the `PrintWriter` if it was successfully created. Comments in Spanish provide context for the logic, such as checking if the list is empty and handling the case where no results are found.

```
126 @
127 static void exportTxt(Tienda tienda){
128     File file = new File( pathname: "comics.txt");
129     PrintWriter pw = null;
130     try {
131         pw=new PrintWriter(file);
132         // Si la búsqueda no produce resultados dejamos mensaje en el archivo
133         if (tienda.getListaComics().isEmpty() && file.canWrite()){
134             pw.println("Su consulta no produjo ningún resultado.");
135             pw.println("Inténtelo de nuevo.");
136         } else if(file.canWrite()) {
137             pw.println("Datos de consulta:");
138             pw.println();
139             for ( Comic comic : tienda.getListaComics()){
140                 pw.println("id: "+comic.getId());
141                 pw.println("titulo: "+comic.getTitulo());
142                 pw.println("autor: "+comic.getAutor());
143                 pw.println("editorial: "+comic.getEditorial());
144                 pw.println("año publicación: "+comic.getFecha_publicacion());
145                 pw.println("género: "+comic.getGenero());
146                 pw.println("precio: "+ comic.getPrecio());
147                 pw.println("stock: "+comic.getStock());
148                 pw.println();
149             }
150         } catch (FileNotFoundException e) {
151             throw new RuntimeException(e);
152         } finally {
153             if(pw!=null){
154                 pw.close();
155             }
156         }
157     }
```

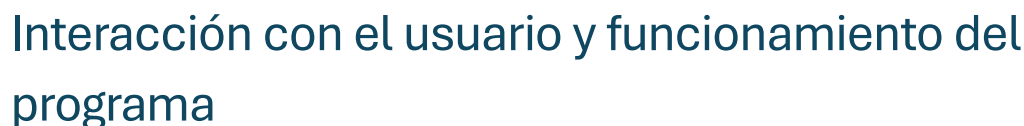
## 2. Exportar a csv:



The screenshot shows the IntelliJ IDEA IDE with the 'Main.java' file open. The code defines a static method `exportCsv` that takes a `Tienda` object as a parameter. It attempts to write the list of comics from the store to a file named 'comics.csv'. The method includes a try-catch block to handle `FileNotFoundException` and a finally block to close the `PrintWriter` if it was successfully created. Comments in Spanish provide context for the logic, such as checking if the list is empty and handling the case where no results are found. The CSV output is formatted with commas as separators.

```
162 @
163 static void exportCsv(Tienda tienda){
164     File file =new File ( pathname: "comics.csv");
165     PrintWriter pw = null;
166     try {
167         pw= new PrintWriter(file);
168         if (tienda.getListaComics().isEmpty() && file.canWrite()){
169             pw.println("Su consulta no produjo ningún resultado,Inténtelo de nuevo");
170         } else if(file.canWrite()){
171             pw.printf("%s,%s,%s,%s,%s,%s,%s,%s","id","titulo","autor","editorial","fecha_publicacion","género","precio","stock");
172             for(Comic comic: tienda.getListaComics()){
173                 pw.printf("%s,%s,%s,%s,%s,%s,%s,%s",
174                     comic.getId(),comic.getTitulo(),comic.getAutor(),comic.getEditorial(),
175                     comic.getFecha_publicacion(),comic.getGenero(),comic.getPrecio(),comic.getStock());
176             }
177         } catch (FileNotFoundException e) {
178             throw new RuntimeException(e);
179         } finally {
180             if(pw!=null){
181                 pw.close();
182             }
183         }
184     }
```

## 3. Exportar a XML:



Página 7 de 12

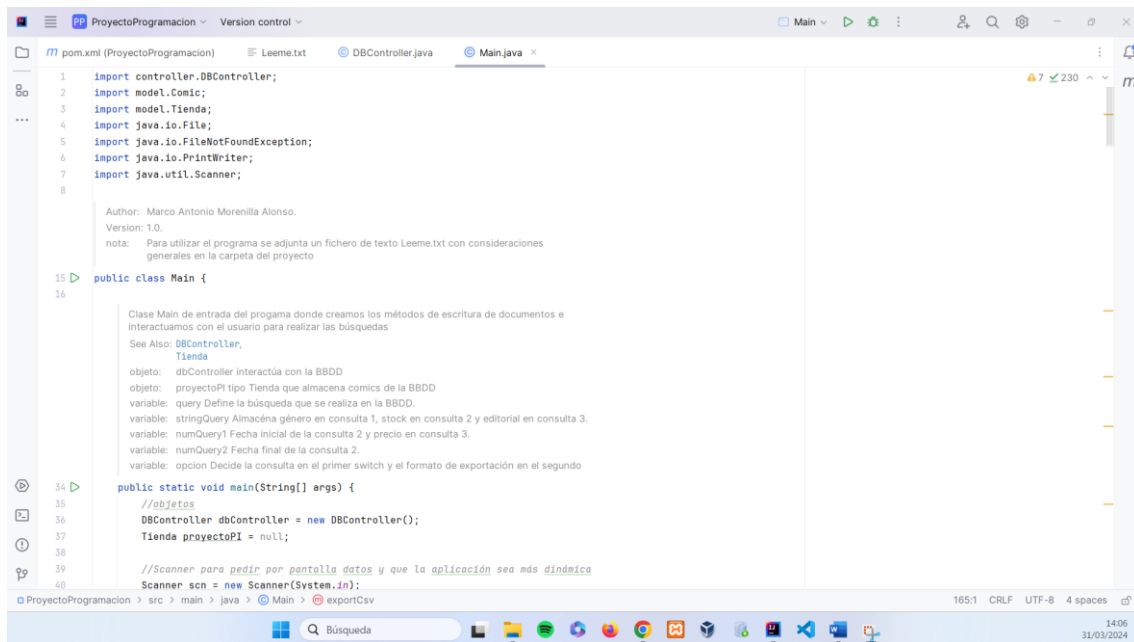


Al seleccionar la consulta se nos realizan preguntas más específicas sobre dicha consulta, con las respuestas que da el usuario, se ejecuta el método de búsqueda de la clase DBController.

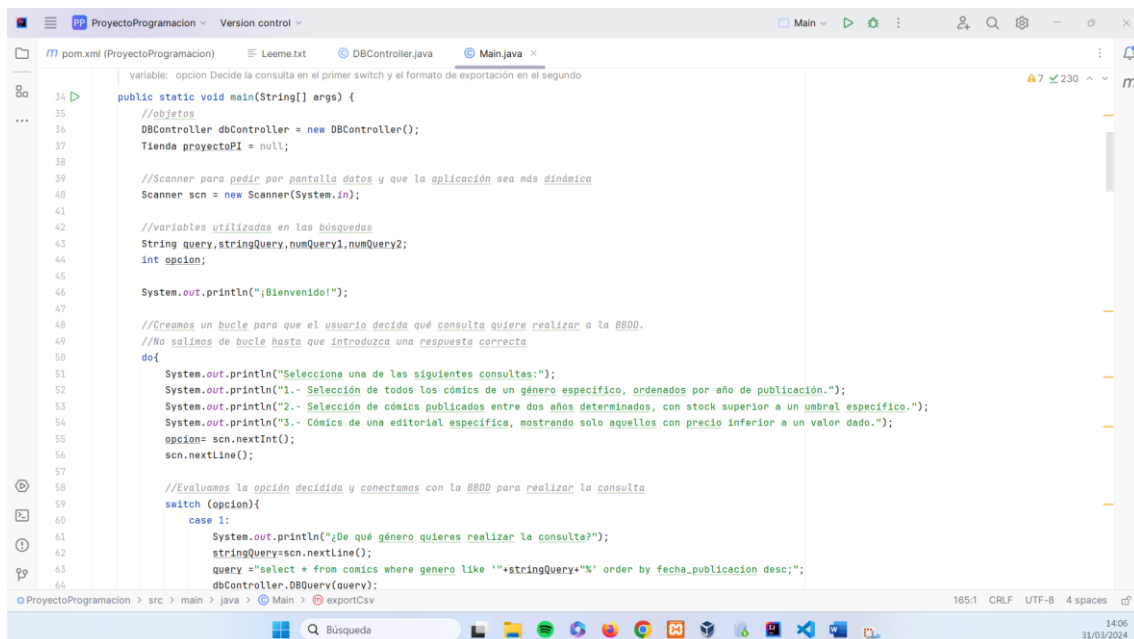
Una vez ejecutada la búsqueda se llena un objeto tienda creado en vacío al inicio del programa y se despliega un segundo menú de opciones para decidir el formato de exportación.

Una vez seleccionado, se llama al método correspondiente y se escribe el documento.

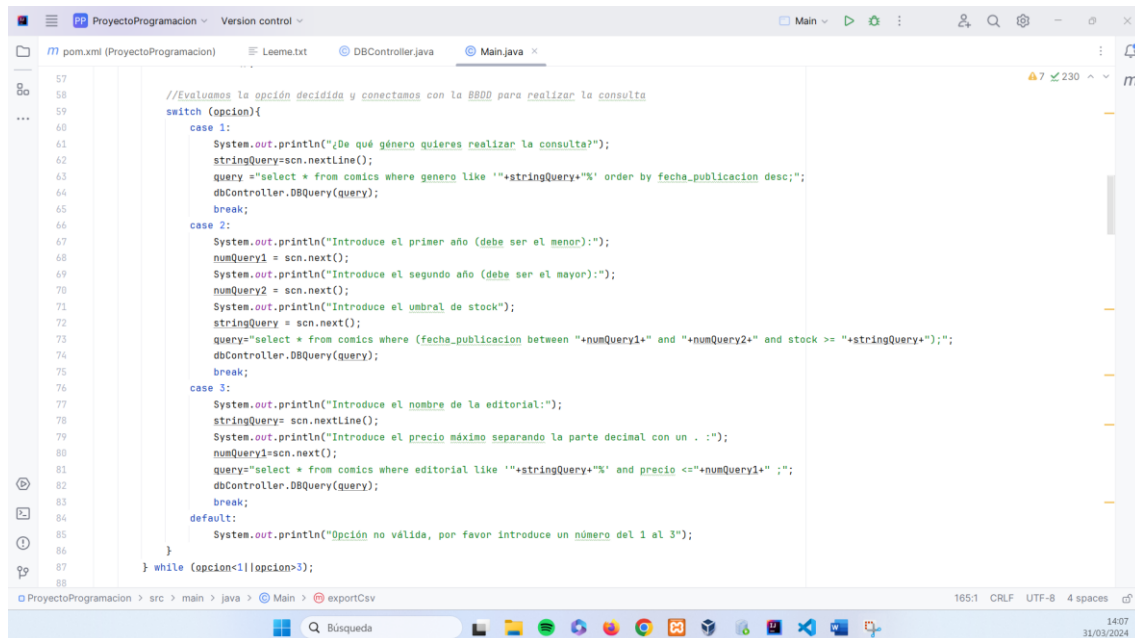
Al finalizar se escribe un mensaje por pantalla que señala el fin de la ejecución.



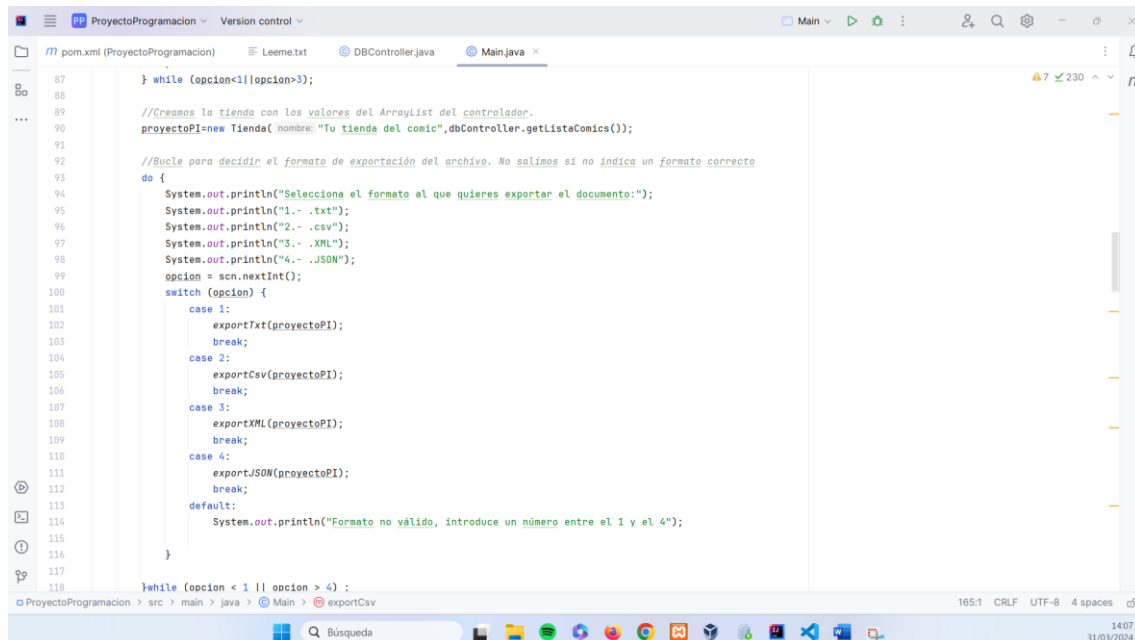
```
1 import controller.DBController;
2 import model.Comic;
3 import model.Tienda;
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.io.PrintWriter;
7 import java.util.Scanner;
8
9 Author: Marco Antonio Morenilla Alonso.
10 Version: 1.0.
11 nota: Para utilizar el programa se adjunta un fichero de texto Leeme.txt con consideraciones
12       generales en la carpeta del proyecto
13
14
15 public class Main {
16
17     Clase Main de entrada del programa donde creamos los métodos de escritura de documentos e
18     interactuamos con el usuario para realizar las búsquedas
19     See Also: DBController,
20              Tienda
21     objeto: dbController interactúa con la BBDD
22     objeto: proyectoPI tipo Tienda que almacena cómics de la BBDD
23     variable: query Define la búsqueda que se realiza en la BBDD.
24     variable: stringQuery Almacena género en consulta 1, stock en consulta 2 y editorial en consulta 3.
25     variable: numQuery1 Fecha inicial de la consulta 2 y precio en consulta 3.
26     variable: numQuery2 Fecha final de la consulta 2.
27     variable: opcion Decide la consulta en el primer switch y el formato de exportación en el segundo
28
29     public static void main(String[] args) {
30         //objetos
31         DBController dbController = new DBController();
32         Tienda proyectoPI = null;
33
34         //Scanner para pedir por pantalla datos y que la aplicación sea más dinámica
35         Scanner scn = new Scanner(System.in);
36
37     }
38 }
```



```
34 public static void main(String[] args) {
35     //objetos
36     DBController dbController = new DBController();
37     Tienda proyectoPI = null;
38
39     //Scanner para pedir por pantalla datos y que la aplicación sea más dinámica
40     Scanner scn = new Scanner(System.in);
41
42     //variables utilizadas en las búsquedas
43     String query, stringQuery, numQuery1, numQuery2;
44     int opcion;
45
46     System.out.println("¡Bienvenido!");
47
48     //Creamos un bucle para que el usuario decida qué consulta quiere realizar a la BBDD.
49     //No salimos de bucle hasta que introduzca una respuesta correcta
50     do{
51         System.out.println("Selecciona una de las siguientes consultas:");
52         System.out.println("1.- Selección de todos los cómics de un género específico, ordenados por año de publicación.");
53         System.out.println("2.- Selección de cómics publicados entre dos años determinados, con stock superior a un umbral específico.");
54         System.out.println("3.- Cómics de una editorial específica, mostrando solo aquellos con precio inferior a un valor dado.");
55         opcion= scn.nextInt();
56         scn.nextLine();
57
58         //Evaluamos la opción decidida y conectamos con la BBDD para realizar la consulta
59         switch (opcion){
60             case 1:
61                 System.out.println("¿De qué género quieres realizar la consulta?");
62                 stringQuery=scn.nextLine();
63                 query ="select * from comics where genero like '"+stringQuery+"%' order by fecha_publicacion desc;";
64                 dbController.DBQuery(query);
65
66     }
67 }
```



```
//Evaluamos la opción decidida y conectamos con la BBDD para realizar la consulta
switch (opcion){
    case 1:
        System.out.println("¿De qué género quieres realizar la consulta?");
        stringQuery=scn.nextLine();
        query ="select * from comics where genero like '"+stringQuery+"%' order by fecha_publicacion desc;";
        dbController.DBQuery(query);
        break;
    case 2:
        System.out.println("Introduce el primer año (debe ser el menor:");
        numQuery1 = scn.nextInt();
        System.out.println("Introduce el segundo año (debe ser el mayor:");
        numQuery2 = scn.nextInt();
        System.out.println("Introduce el umbral de stock");
        stringQuery = scn.nextLine();
        query="select * from comics where (fecha_publicacion between '"+numQuery1+"' and '"+numQuery2+"' and stock >= '"+stringQuery+"');";
        dbController.DBQuery(query);
        break;
    case 3:
        System.out.println("Introduce el nombre de la editorial:");
        stringQuery= scn.nextLine();
        System.out.println("Introduce el precio máximo separando la parte decimal con un . :");
        numQuery1=scn.nextInt();
        query="select * from comics where editorial like '"+stringQuery+"%' and precio <='"+numQuery1+"' ";
        dbController.DBQuery(query);
        break;
    default:
        System.out.println("Opción no válida, por favor introduce un número del 1 al 3");
}
} while (opcion<1||opcion>3);
```



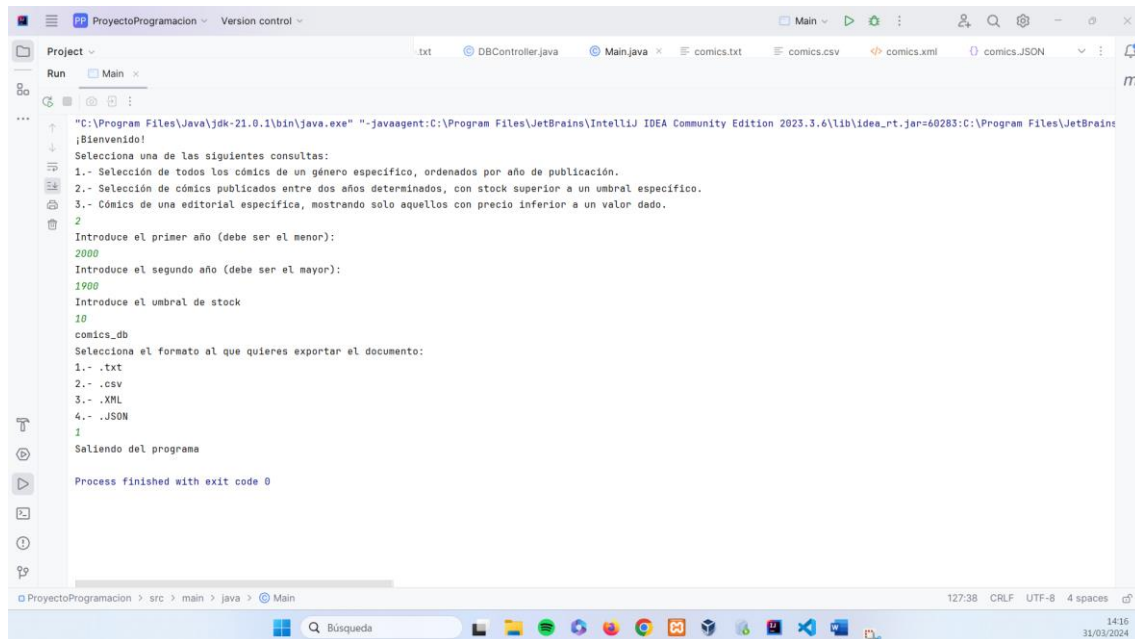
```
} while (opcion<1||opcion>3);

//Creamos la tienda con los valores del ArrayList del controlador.
proyectoPI=new Tienda( nombre:"Tu tienda del comic",dbController.getListComics());

//Bucle para decidir el formato de exportación del archivo. No salimos si no indica un formato correcto
do {
    System.out.println("Selecciona el formato al que quieres exportar el documento:");
    System.out.println("1.- .txt");
    System.out.println("2.- .csv");
    System.out.println("3.- .XML");
    System.out.println("4.- .JSON");
    opcion = scn.nextInt();
    switch (opcion) {
        case 1:
            exportTxt(proyectoPI);
            break;
        case 2:
            exportCsv(proyectoPI);
            break;
        case 3:
            exportXML(proyectoPI);
            break;
        case 4:
            exportJSON(proyectoPI);
            break;
        default:
            System.out.println("Formato no válido, introduce un número entre el 1 y el 4");
    }
} while (opcion < 1 || opcion > 4);
```

## Demo

Se realiza para que la consulta no produzca resultados y dejar un fichero txt como ejemplo.txt en resultados para mostrar funcionamiento si la búsqueda no tiene resultados.



```
"C:\Program Files\Java\jdk-21.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.6\lib\idea_rt.jar=60283:C:\Program Files\JetBrains\
;Bienvenido!
Selecciona una de las siguientes consultas:
1.- Selección de todos los cómics de un género específico, ordenados por año de publicación.
2.- Selección de cómics publicados entre dos años determinados, con stock superior a un umbral específico.
3.- Cómics de una editorial específica, mostrando solo aquellos con precio inferior a un valor dado.
2
Introduce el primer año (debe ser el menor):
2000
Introduce el segundo año (debe ser el mayor):
1900
Introduce el umbral de stock
10
comics_db
Selecciona el formato al que quieres exportar el documento:
1.- .txt
2.- .csv
3.- .XML
4.- .JSON
1
Saliendo del programa

Process finished with exit code 0
```

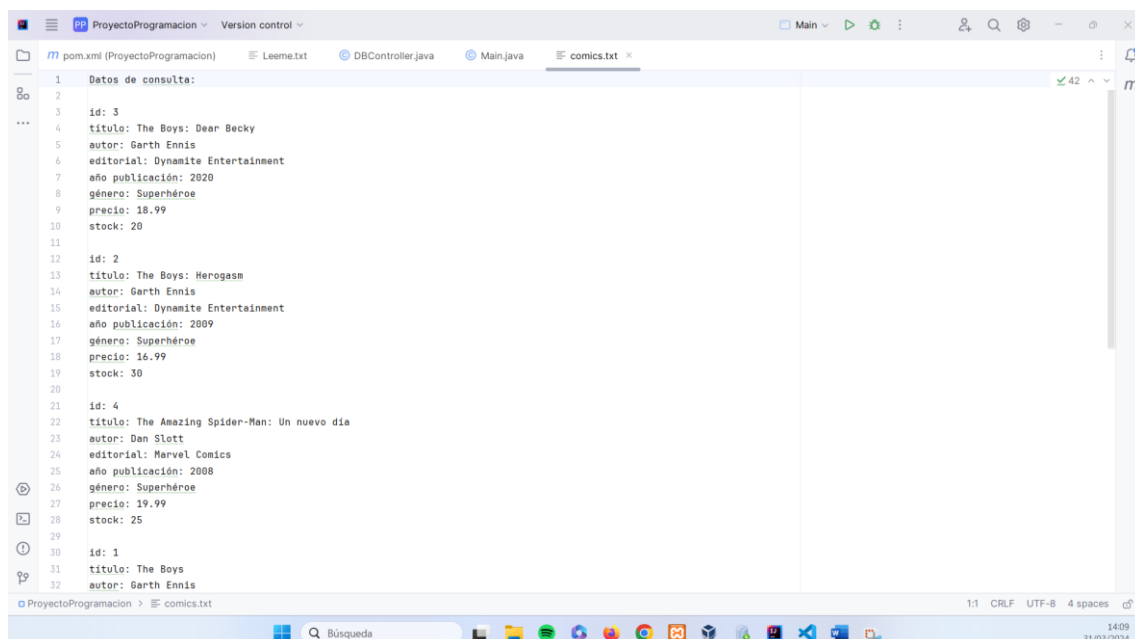
## Resultados

Los documentos se dejan en la carpeta del proyecto creados por si se quieren revisar ya que la ruta del objeto file es relativa a la raíz de este “comics.+extensión”.

Se deja además un ejemplo de txt si no se producen resultados.

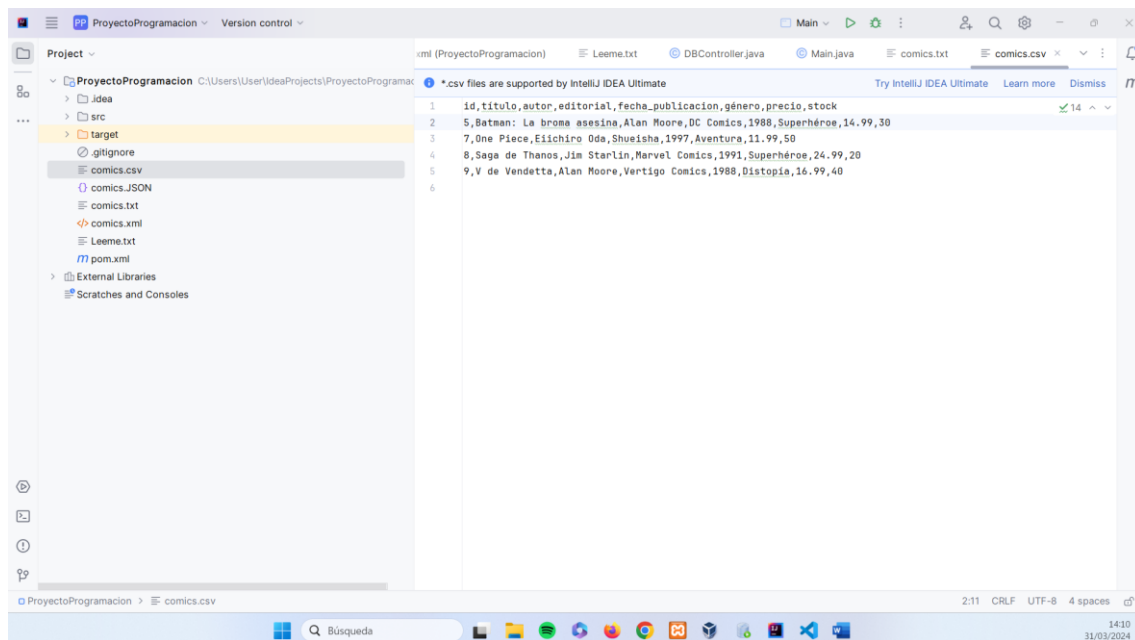
“ejemplo.txt” y se cambia luego la ruta a “comics.txt” con el ejemplo de la ejecución del programa adjuntada en el apartado anterior.

Txt:

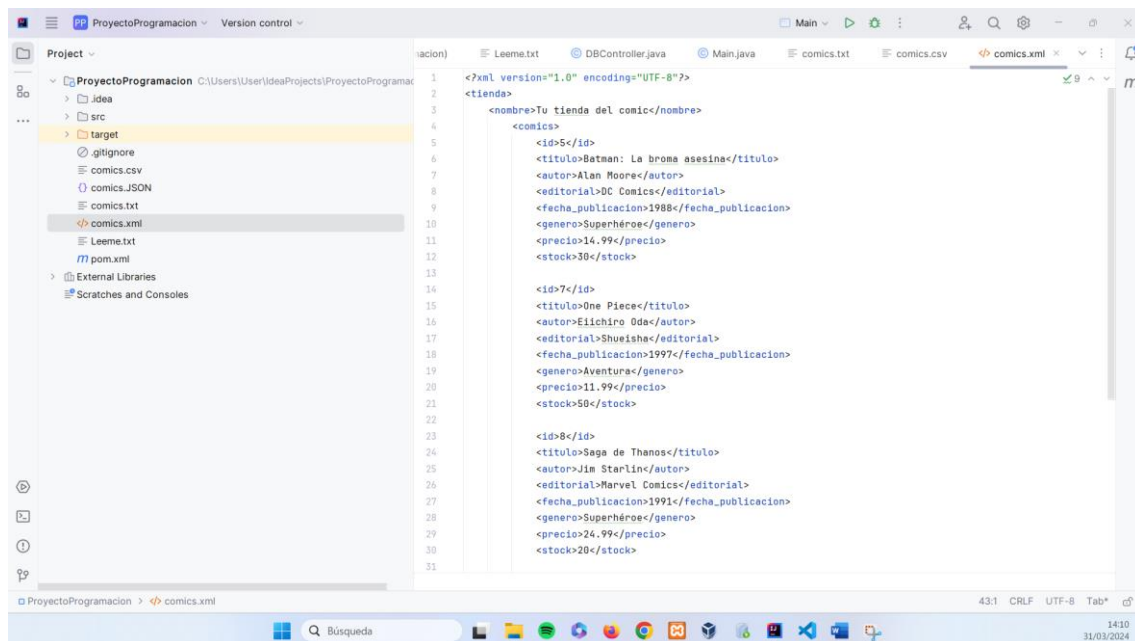


```
1 Datos de consulta:
2
3 id: 3
4 titulo: The Boys: Dear Becky
5 autor: Garth Ennis
6 editorial: Dynamite Entertainment
7 año publicación: 2020
8 género: Superhéroe
9 precio: 18.99
10 stock: 20
11
12 id: 2
13 titulo: The Boys: Herogasm
14 autor: Garth Ennis
15 editorial: Dynamite Entertainment
16 año publicación: 2009
17 género: Superhéroe
18 precio: 16.99
19 stock: 30
20
21 id: 4
22 titulo: The Amazing Spider-Man: Un nuevo día
23 autor: Dan Slott
24 editorial: Marvel Comics
25 año publicación: 2008
26 género: Superhéroe
27 precio: 19.99
28 stock: 25
29
30 id: 1
31 titulo: The Boys
32 autor: Garth Ennis
```

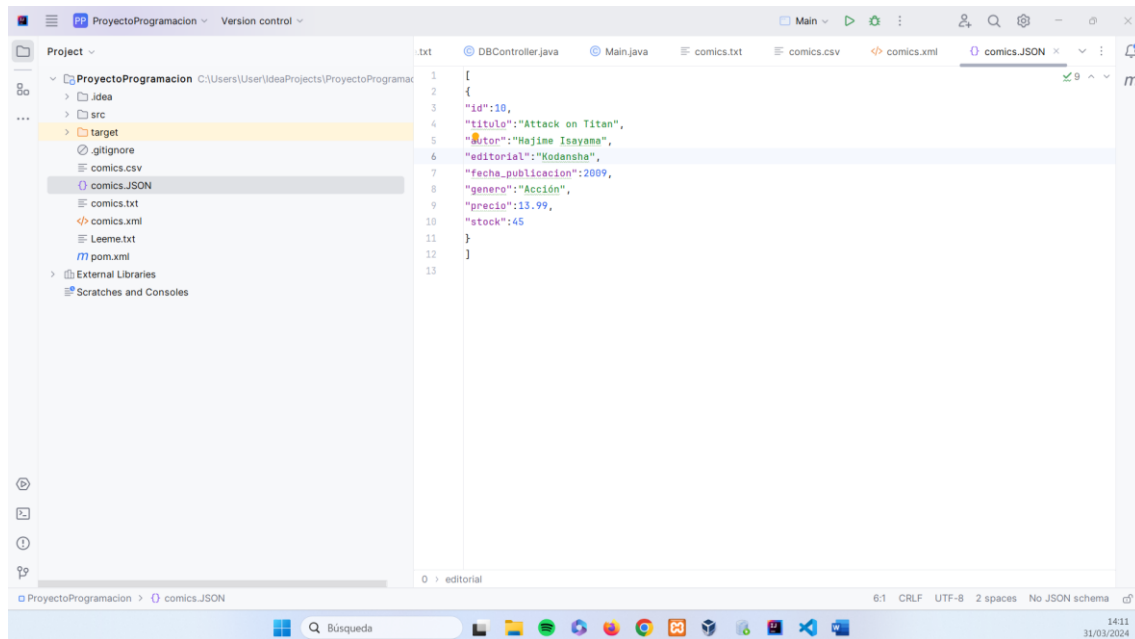
## Csv:



## XML:



## JSON:



### Ejemplo.txt:

