

ciclo: [DAM]

MÓDULO DE [PROGRAMACIÓN]

[Tarea Nº 08]

Alumno:

[Juan Carlos Filter Martín]

[15456141A]

**Contenido**

[1. Documentos que se adjuntan a este informe. 3](#__RefHeading___Toc85_705307559)

[2. Descripción de la tarea 3](#__RefHeading___Toc87_705307559)

[3. RA08\_c) Se han instalado sistemas gestores de bases de datos orientados a objetos. 4](#__RefHeading___Toc89_705307559)

[A) Proyecto Tarea8: Crear un proyecto llamado Tarea8. 4](#__RefHeading___Toc91_705307559)

[B) Preparar para trabajar con una BDOO DB4o. 5](#__RefHeading___Toc93_705307559)

[Descargar Bd4o. 5](#__RefHeading___Toc95_705307559)

[Añadir la biblioteca de Db4o al proyecto java de NetBeans. 7](#__RefHeading___Toc97_705307559)

[4. RA08\_e) Se han creado bases de datos y las estructuras necesarias para el almacenamiento de objetos. 11](#__RefHeading___Toc99_705307559)

[A) Fichero Tarea8: Se va a trabajar con una BD que almacenaremos en un fichero “Tarea8”. 11](#__RefHeading___Toc463_1569325315)

[5. RA08\_f) Se han programado aplicaciones que almacenen objetos en las bases de datos creadas. 12](#__RefHeading___Toc103_705307559)

[A) Clase Articulo: para crear los objetos que almacenaremos en la BD. 12](#__RefHeading___Toc105_705307559)

[6. RA08\_g) Se han realizado programas para recuperar, actualizar y eliminar objetos de las bases de datos. 14](#__RefHeading___Toc107_705307559)

[A) Clase GestorBD: Toda la gestión de la información se hará a través de una clase. 14](#__RefHeading___Toc109_705307559)

[B) Clase ControladorErrores: Se va a crear una clase para controlar los Errores. 19](#__RefHeading___Toc109_705307559_Copia_1)

[C) JFrame form: Parte gráfica para que sean llamado los distintos eventos de la interfaz 20](#__RefHeading___Toc109_705307559_Copia_1_)

[7. RA08\_h) Se han realizado programas para almacenar y gestionar tipos de datos estructurados, compuestos y relacionados. 22](#__RefHeading___Toc107_705307559_Copia_1)

[Botón Insertar 22](#__RefHeading___Toc477_1569325315)

[Interfaz gráfica 22](#__RefHeading___Toc479_1569325315)

[Código 23](#__RefHeading___Toc481_1569325315)

[Botón Buscar 25](#__RefHeading___Toc477_1569325315_Copia_1)

[Interfaz gráfica 25](#__RefHeading___Toc479_1569325315_Copia_1)

[Código 26](#__RefHeading___Toc481_1569325315_Copia_1)

[Botón Borrar 28](#__RefHeading___Toc477_1569325315_Copia_2)

[Interfaz gráfica 28](#__RefHeading___Toc479_1569325315_Copia_2)

[Código 29](#__RefHeading___Toc481_1569325315_Copia_2)

[Botón Modificar 31](#__RefHeading___Toc477_1569325315_Copia_3)

[Interfaz gráfica 31](#__RefHeading___Toc479_1569325315_Copia_3)

[Código 32](#__RefHeading___Toc481_1569325315_Copia_3)

[Botón Mostrar 34](#__RefHeading___Toc477_1569325315_Copia_4)

[Interfaz gráfica 34](#__RefHeading___Toc479_1569325315_Copia_4)

[Código 35](#__RefHeading___Toc481_1569325315_Copia_4)

[Botón Salir 37](#__RefHeading___Toc477_1569325315_Copia_5)

[Interfaz gráfica 37](#__RefHeading___Toc479_1569325315_Copia_5)

# Documentos que se adjuntan a este informe.

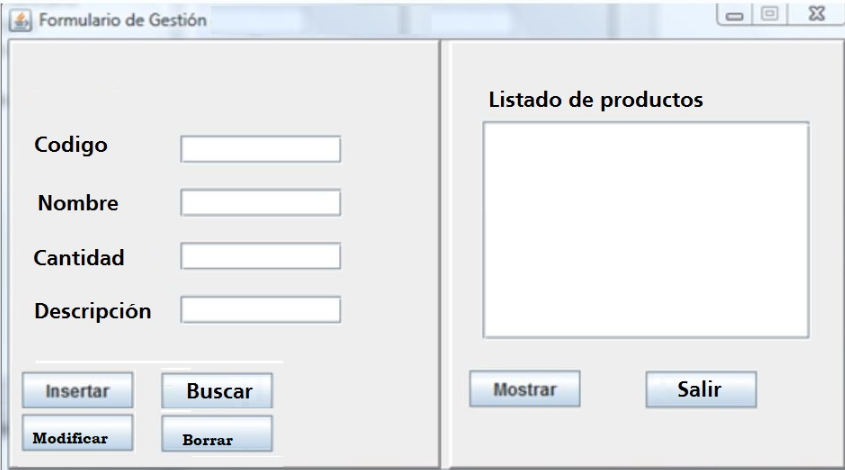
A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Proyecto Java “Tarea8”

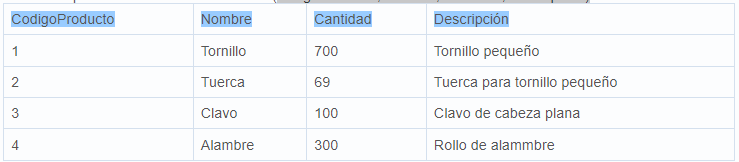
# Descripción de la tarea

Realizar una aplicación Java que tendrá un entorno gráfico tal como el que se muestra en la siguiente figura:

*(Orientativo, se puede realizar el diseño que desee siempre que la aplicación no pierda funcionalidad)*



En cada producto se debe introducir (codigoProducto, Nombre, Cantidad, Descripción)



*Esto es un ejemplo, se pueden crear los datos que se deseen*

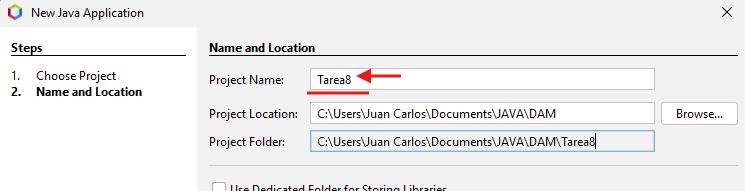
# RA08\_c) Se han instalado sistemas gestores de bases de datos orientados a objetos.

## Proyecto Tarea8: Crear un proyecto llamado Tarea8.

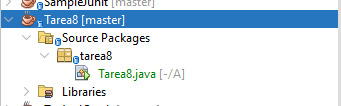
File > New Project > Java with Ant > **Java Application**

## 

Ponemos de nombre al proyecto **Tarea8**

****

El proyecto ya estaría creado



## Preparar para trabajar con una BDOO DB4o.

### Descargar Bd4o.

Bd4o es un motor de base de datos orientado a objetos. Este está disponible para entornos Java

Para hacer uso de esta base de datos tendremos que entrar en la web del fabricante y descargar la última versión

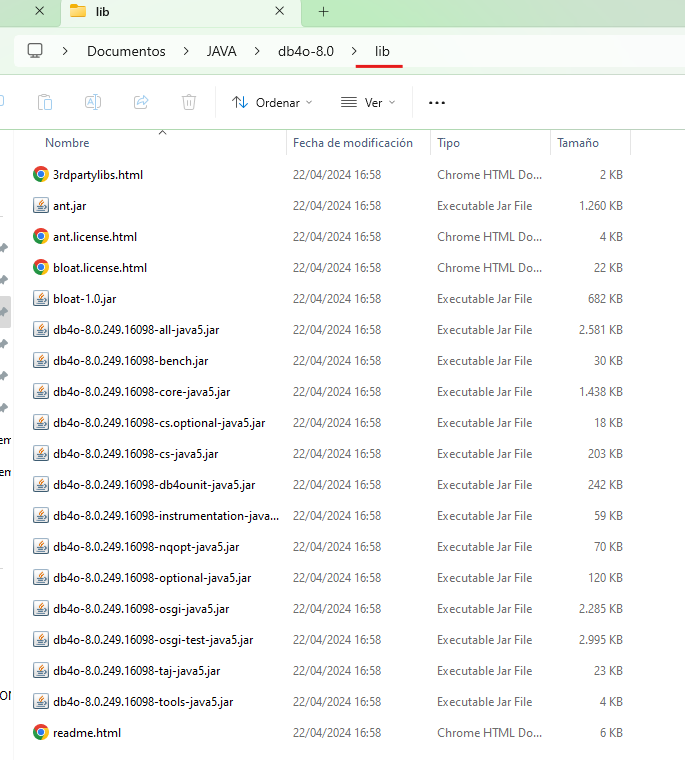


He tenido problemas para entra a la web. Esta captura está

Realizada desde el PDF del tema 8 y se ha descargado Bd4o

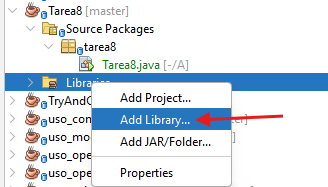
desde el campus

Una vez descargado **se descomprime** y **accedemos a la carpeta que se llama lib**. En esta se encuentran las **bibliotecas necesarias para poder utilizar este Sistema Gestor Base de Datos**

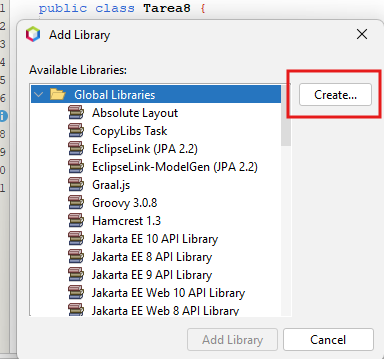
****

### **Añadir la biblioteca de Db4o al proyecto java de NetBeans.**

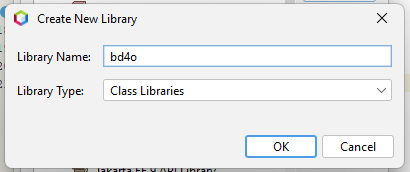
Botón derecho sobre la carpeta Libraries > **Add Library**

****

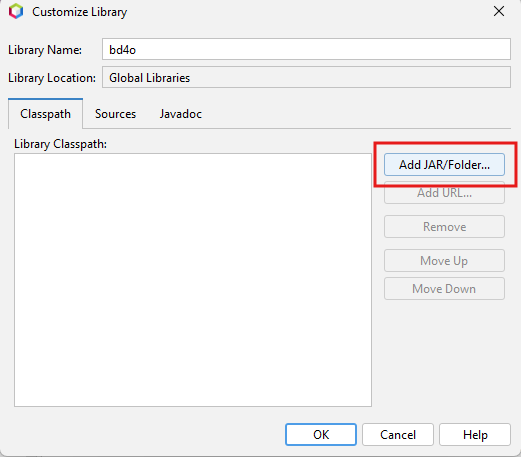
**Pulsamos en Create** para crear una nueva biblioteca

****

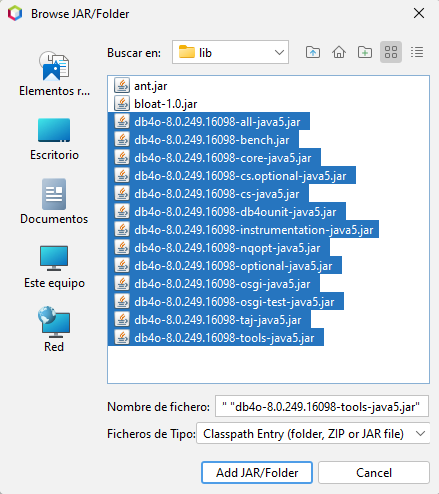
**Asignamos un nombre** a la librería

****

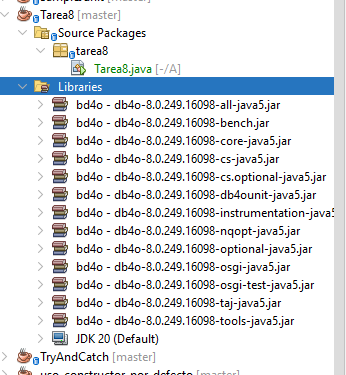
**Add/JAR/Folder** ( Añadir jar/carpeta) para añadir los ficheros a nuestra biblioteca.

****

**Añadimos los que se muestran a continuación**

****

Finalmente en nuestro proyecto debe aparecer en librerías de la siguiente forma estando ya listo para poder trabajar con la Base de datos BD4o

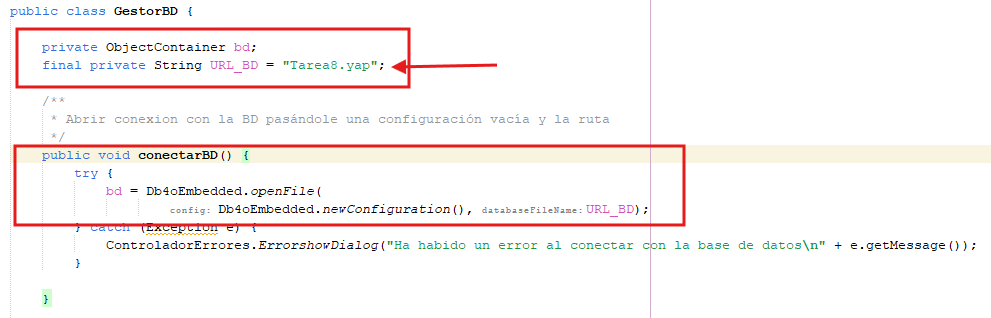
****

# RA08\_e) Se han creado bases de datos y las estructuras necesarias para el almacenamiento de objetos.

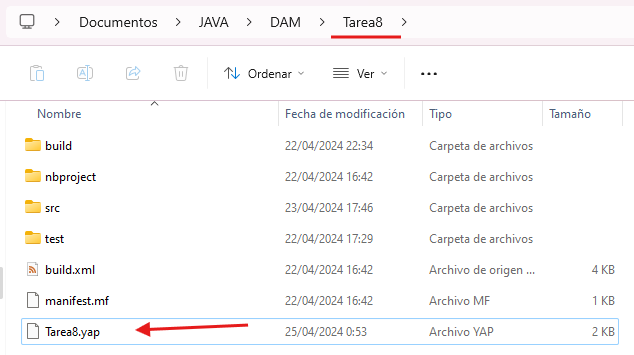
## Fichero Tarea8: Se va a trabajar con una BD que almacenaremos en un fichero “Tarea8”.

En la clase que se va a encargar de gestionar la BD se va a crear una constante con la URL donde se va a encontrar el archivo llamado Tarea8.yap

Aparte tendremos un método para conectar a esa base de datos (Se abre el fichero con una configuración vacía y pasándola la ruta)

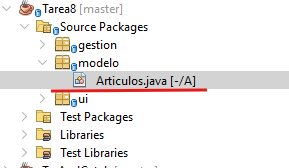
**

Este archivo se encuentra en la ruta del proyecto ya que es una ruta relativa.

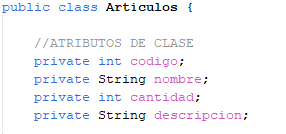
**

# RA08\_f) Se han programado aplicaciones que almacenen objetos en las bases de datos creadas.

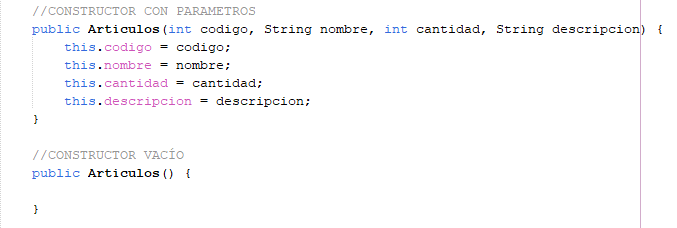
## Clase Articulo: para crear los objetos que almacenaremos en la BD.



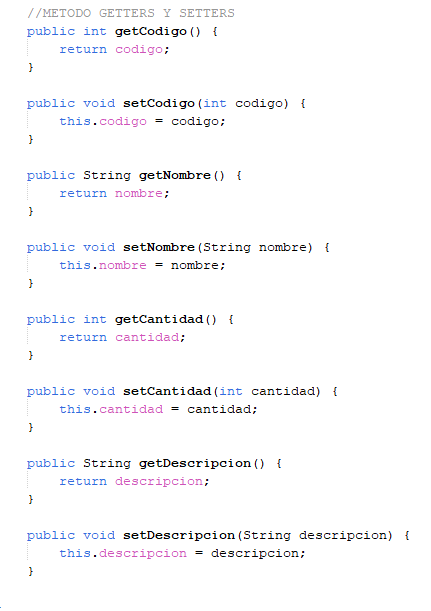
**Atributos de la clase**: *Código, Nombre, Cantidad, Descripción*



**Constructores**: *Con parámetros y sin parámetros.*

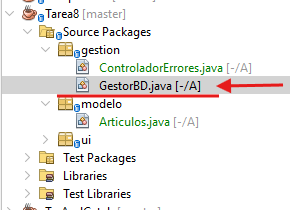
**

***Métodos****: Getters y Setters*

**

# RA08\_g) Se han realizado programas para recuperar, actualizar y eliminar objetos de las bases de datos.

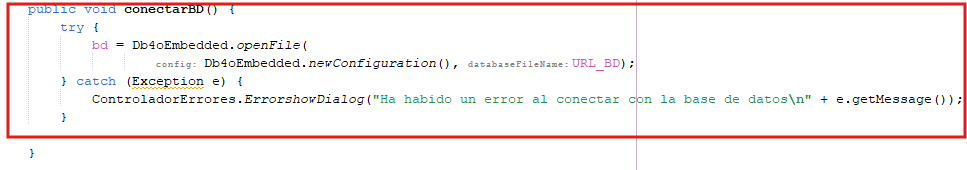
## Clase GestorBD: Toda la gestión de la información se hará a través de una clase.



En esta clase llamada GestorBD se encuentran los distintos métodos para la conexión con la Base de dato (insertar,consultar, actualizar, borrar).

Estos métodos serán llamados desde los distintos eventos de la interfaz.

**Conectar a la base de datos**



**Insertar articulo en la BD**

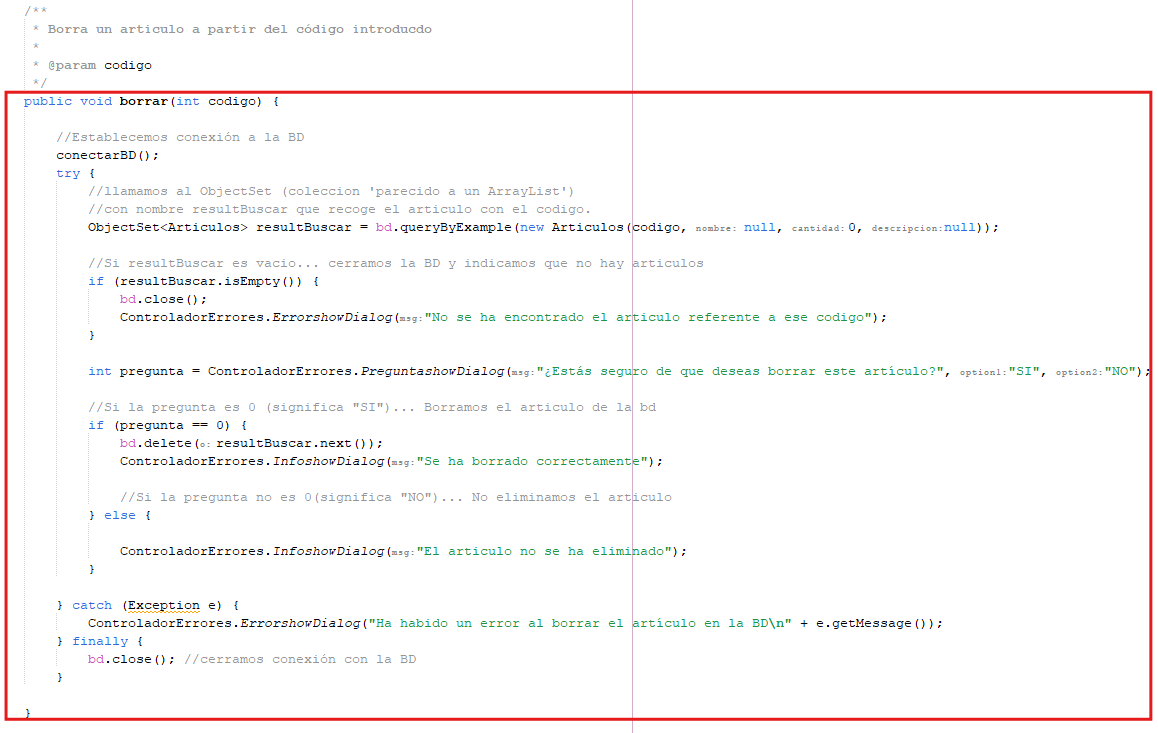


**Buscar articulos**

**Buscar articulo en la BD**

**Borrar articulo**

**Borrar articulo en la BD**





**Actualizar articulo**

**Actualizar el articulo en la BD**

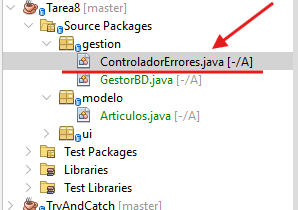
**Mostrar todos los articulos en la BD**



**Método para comprobar si existe el articulo, obteniendo true o false**

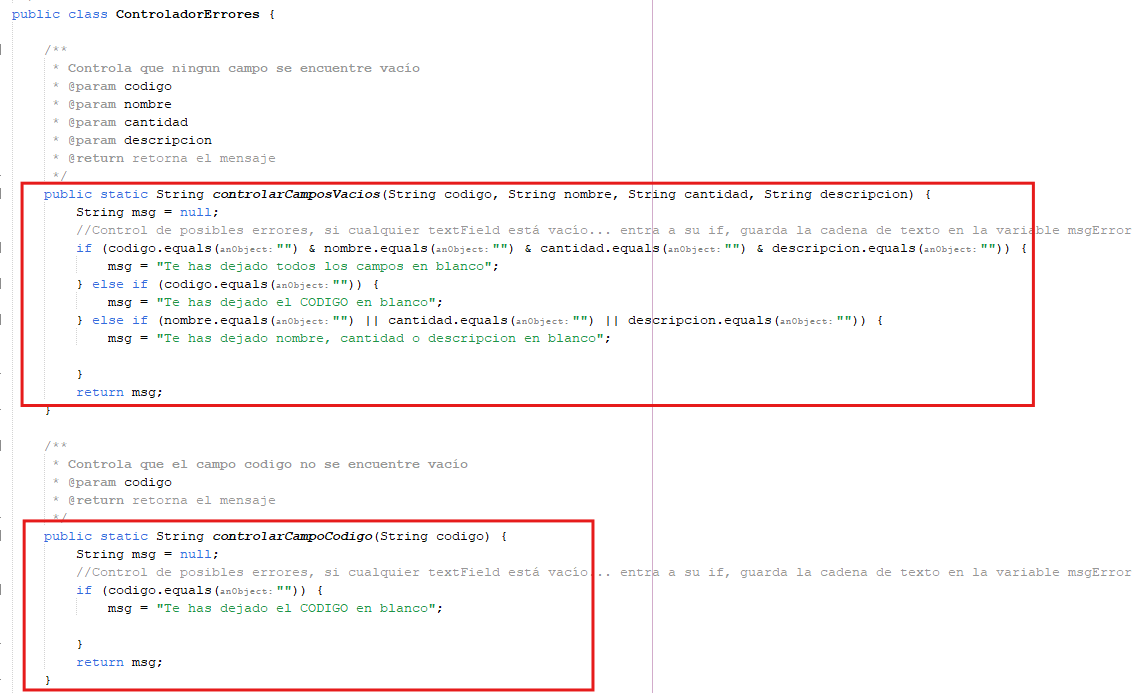
## Clase ControladorErrores: Se va a crear una clase para controlar los Errores.

Esta clase se va a llamar **ControladorErrores**



Va a contener métodos para evitar dejar campos en blancos y posibles errores del programa.

También se van a almacenar los diferentes métodos para mostrar JoptionPanel



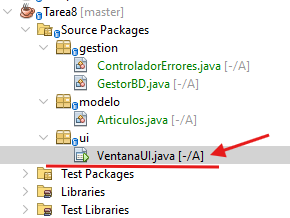
**Método para evitar dejar campos Vacíos**

**Método para evitar dejar el campo código vacío**

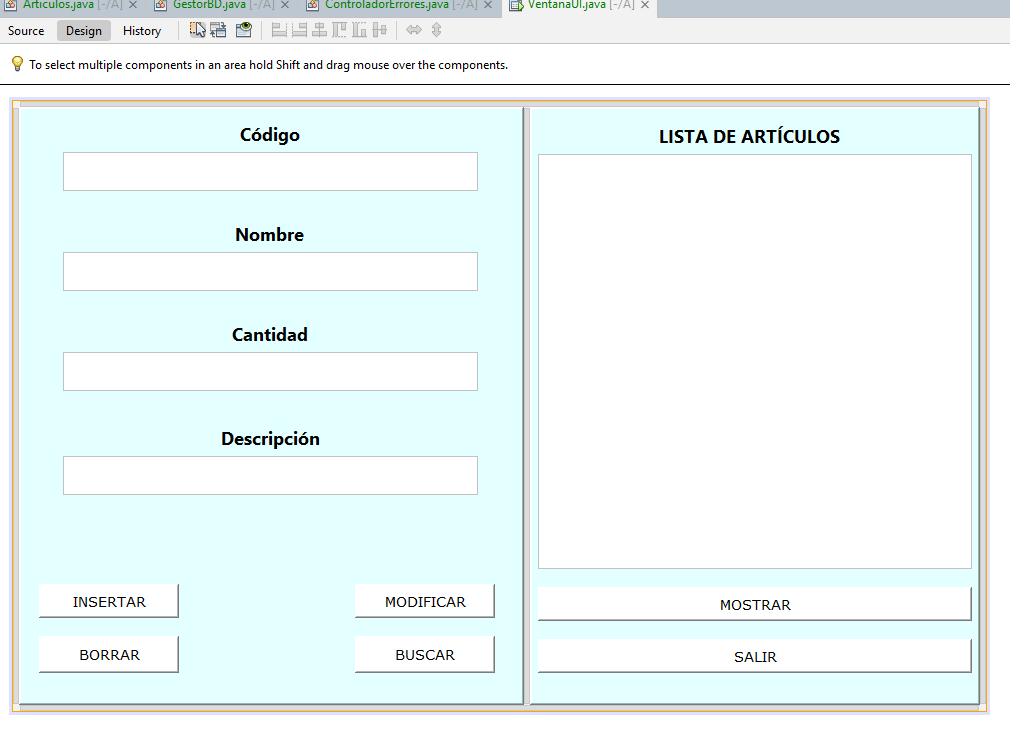
**Métodos para enviar ventanas emergentes de error, pregunta o información**



## JFrame form: P**arte gráfica para que sean llamado los distintos eventos de la interfaz**

**JFrame form** será nombrado como **VentanaUI.**

Contiene una parte gráfica



Como también la parte de código en la cual se llamará a los métodos de la clase GestorBD para que al hacer interacción con los botones realicen su función.

**Ejemplo de una parte del código**



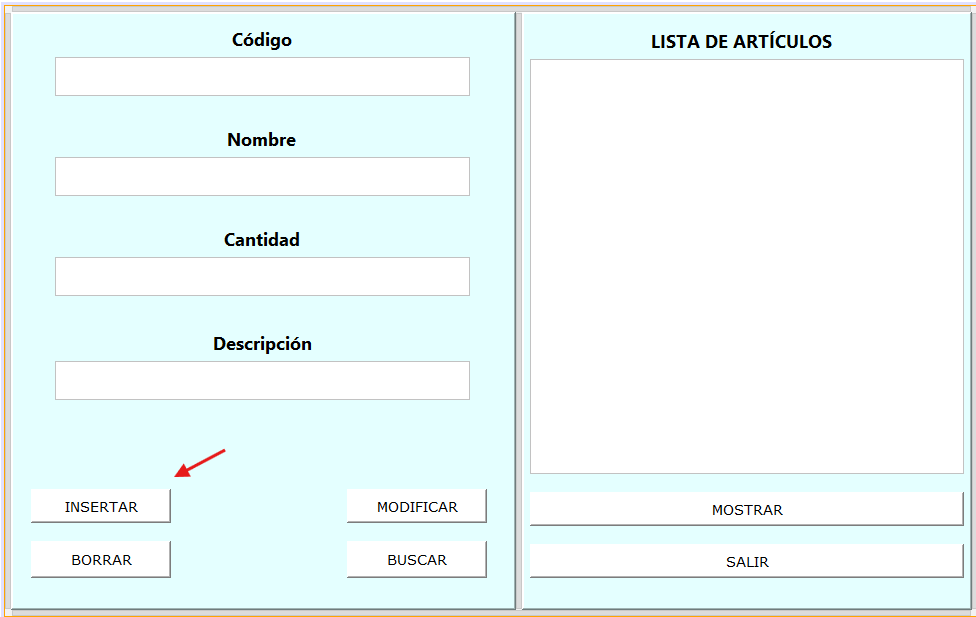
# RA08\_h) Se han realizado programas para almacenar y gestionar tipos de datos estructurados, compuestos y relacionados.

## Botón Insertar

***El objetivo es captar la información tecleada por el usuario de la aplicación y guardarla la base de datos.***

### Interfaz gráfica

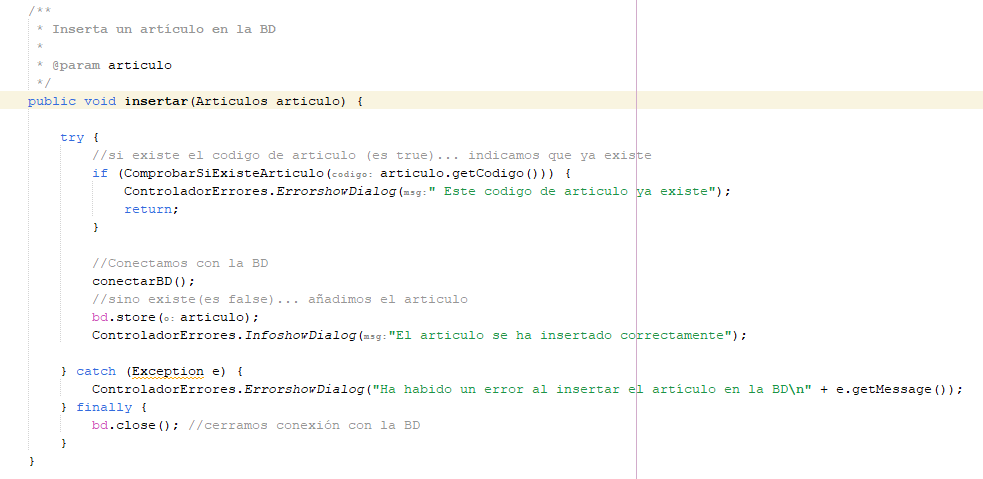
Al rellenar los TextField se añadirá el articulo en la BD, evitando duplicados y controlando que no se dejen campos vacíos.



### Código

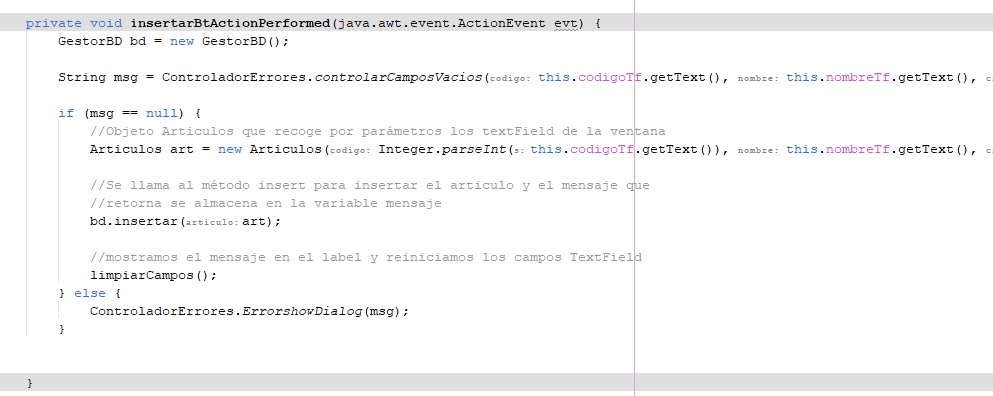
**Clase GestorBD**

* Si el método ComprobarSiExisteArtículo devuelve true entonces indica que ya existe el articulo.
* Si no existe entonces se conecta a la BD, añade el articulo, lo indica mediante una ventana emergente y por ultimo en el finally cierra la conexión con la BD



**Jform VentanaUI**

* Creamos un objeto de la clase GestorBD para llamar a los métodos
* Almacenamos en un String msg el resultado que obtenemos con el método ControlarCamposVacios de la clase controlarErrores (al ser static se llama directamente a la clase sin tener que crear un objeto de esta)
* Si msg es null significa que no hay campos vacíos entonces creamos un objeto de la clase articulo (indicándole por parámetros los TextField)
* Posteriormente insertamos el articulo en la base de datos con el método insertar de la clase GestorBD y limpiamos los campos TextField
* En caso de que obtuviéramos algún mensaje mediante el método controlarCamposVacios este enviará ese mensaje mediante una ventana emergente y no almacenaría el articulo en el fichero



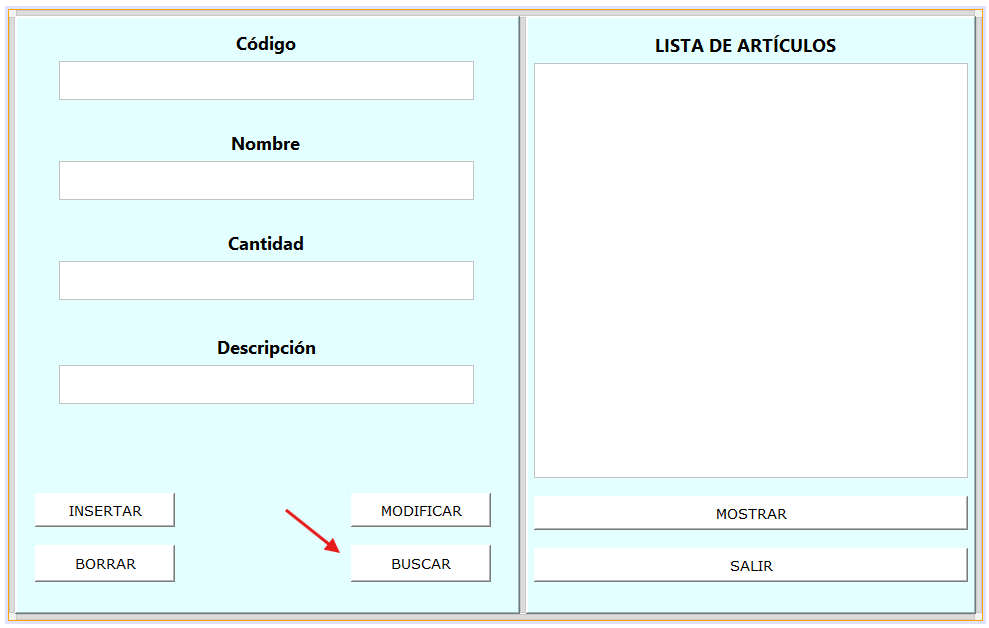
## Botón Buscar

***El usuario escribirá un código en el campo correspondiente, al hacer click en buscar, el programa buscará en la BD el producto con ese código.***

***Si no existe dará un mensaje de error indicándolo.***

***Si existe, rellenará el resto de campos con los datos que corresponden al producto con el código tecleado.***

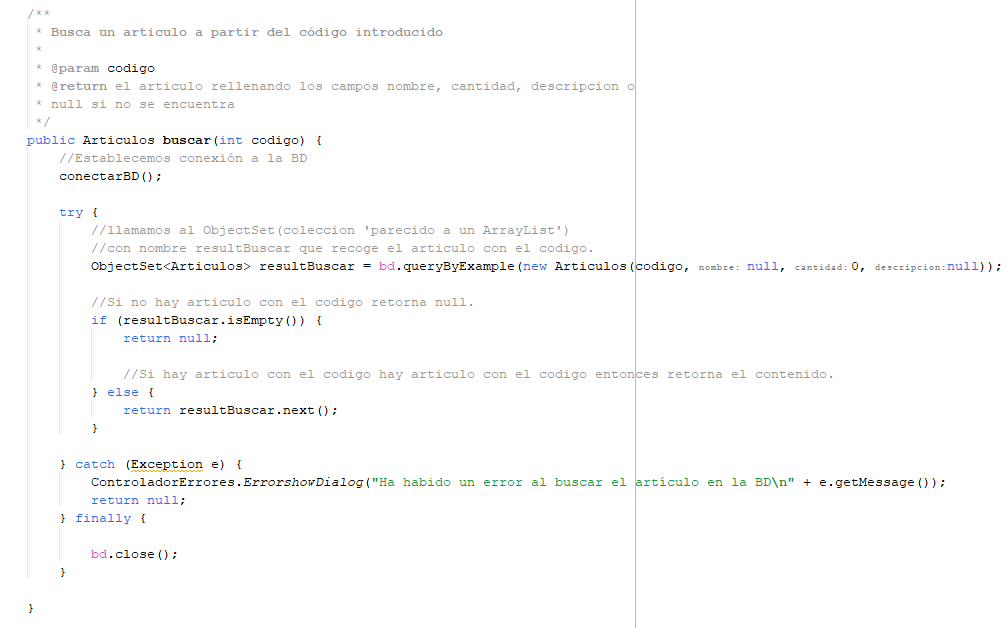
### Interfaz gráfica

Al rellenar el TextFiel código este buscará el articulo en la BD, evitando que no se deje el campo código en blanco.

### Código

**Clase GestorBD**

* Establecemos conexión con la BD
* Creamos un objeto de la colección ObjectSet que va a recoger el articulo con el código indicado.
* Este comprobará si existe o no
* Si no existe entonces se va a retornar null y si existe se va a retornar el articulo.
* Por ultimo en el finally cerramos la bd .



**Jform VentanaUI**

* Creamos un objeto de la clase GestorBD para llamar a los métodos
* Almacenamos en un String msg el resultado que obtenemos con el método ControlarCampoCodigo de la clase controlarErrores (al ser static se llama directamente a la clase sin tener que crear un objeto de esta)
* Si msg es null significa que el campo código no está vacío entonces buscamos el articulo (indicándole por parámetros el código introducido en el TextField)
* Si el resultado de la búsqueda que obtengamos es diferente a null significa que si ha encontrado un articulo entonces rellenamos los campos restantes (nombre, cantidad, descripción) en los TextField.
* Si se diera el caso de obtener como resultado de la búsqueda null (no hay articulo) enviaríamos un mensaje indicándolo
* En caso de que obtuviéramos un mensaje mediante el método controlarCampoCodigo este enviará ese mensaje en una ventana emergente indicando que está vacío el TextField código.

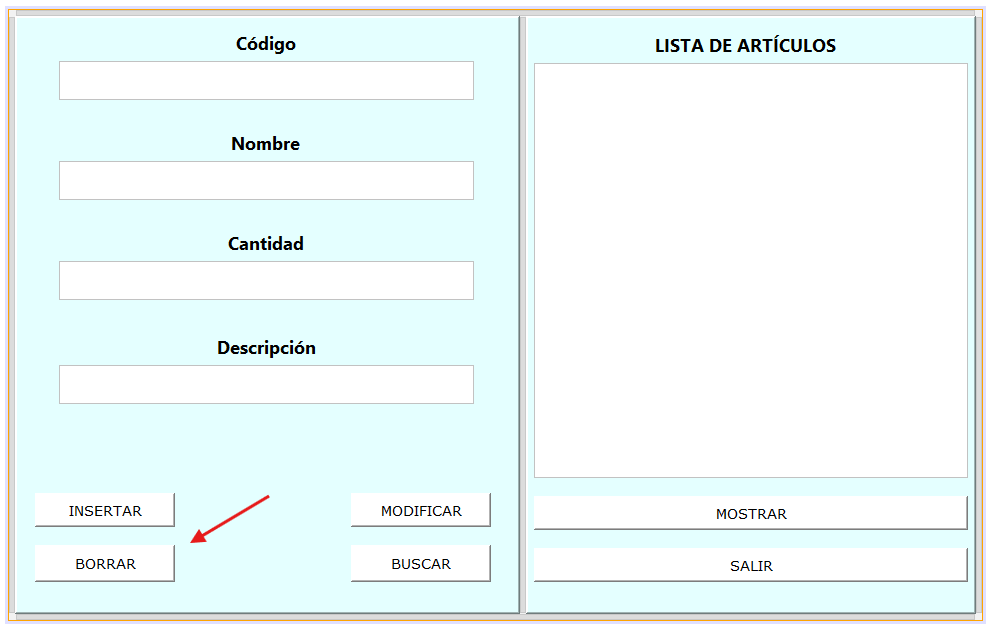


## Botón Borrar

***El usuario escribirá un código en el campo correspondiente, al hacer click si existe ese producto, el programa preguntará si está seguro de eliminar ese producto si responde que sí se eliminará definitivamente..***

### Interfaz gráfica

Al rellenar el TextFiel código este borrará el articulo en la BD, evitando que no se deje el campo código en blanco y preguntando antes de eliminarlo si realmente desea borrar el articulo.



### Código

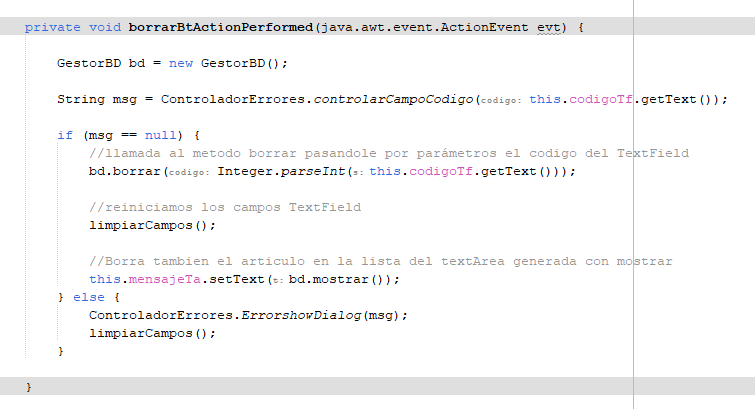
**Clase GestorBD**

* Establecemos conexión con la BD
* Creamos un objeto de la colección ObjectSet que va a recoger el articulo con el código indicado.
* Este comprobará si existe o no
* Si no existe entonces mandará una ventana emergente indicándolo.
* En cambio si existe lanzará una pregunta si realmente desea eliminarlo.
* Si la respuesta a la pregunta es SI entonces eliminará el articulo e indicándolo que se ha borrado.
* Si la respuesta a la pregunta es NO entonces no eliminará el articulo y se indicará mediante un mensaje.
* Por ultimo en el finally cerramos la bd .



**Jform VentanaUI**

* Creamos un objeto de la clase GestorBD para llamar a los métodos
* Almacenamos en un String msg el resultado que obtenemos con el método ControlarCampoCodigo de la clase controlarErrores (al ser static se llama directamente a la clase sin tener que crear un objeto de esta)
* Si msg es null significa que el campo código no está vacío entonces llamamos al método borrar indicándole por parámetros el código introducido en el TextField.
* Ese método hace la función de la búsqueda del articulo mediante el código, si se encuentra pregunta si desea borrarlo y realiza su eliminación.
* Posteriormente limpiamos los campos TextField y borramos también el articulo en la lista del TextArea mediante el método mostrar(realmente lo que hace es sobrescribir esta lista con los nuevos cambios)
* En caso de que obtuviéramos un mensaje mediante el método controlarCampoCodigo este enviará ese mensaje en una ventana emergente indicando que está vacío el TextField código.

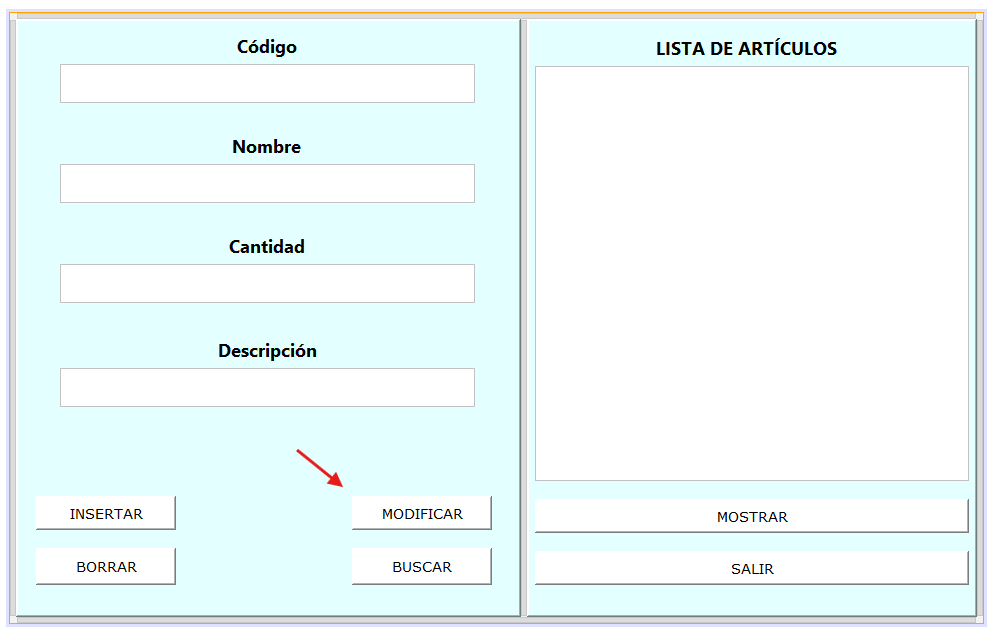


## Botón Modificar

***El usuario escribirá un código en el campo correspondiente, a modificar (código), en el resto de campos se pondrán los nuevos valores. Al pulsar sobre modificar si existe ese producto se modificarán sus datos.***

### Interfaz gráfica

Al rellenar el TextFiel código este modificará el articulo en la BD, evitando que no se deje el campo código en blanco.



### Código

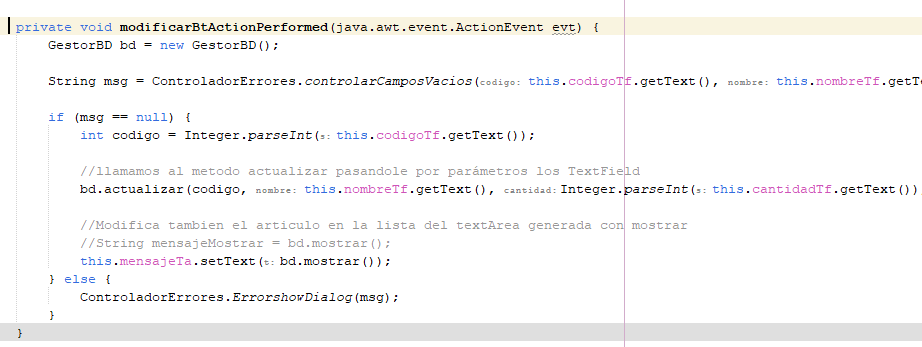
**Clase GestorBD**

* Establecemos conexión con la BD
* Creamos un objeto de la colección ObjectSet que va a recoger el articulo con el código indicado.
* Este comprobará si existe o no
* Si no existe entonces mandará una ventana emergente indicándolo.
* En cambio si existe se crea un objeto articulo del resultado de la búsqueda y realizará la siguiente operación:
* Si el nombre, la cantidad y la descripción pasada por parámetros no es null/0 le asignamos ese nuevo cambio mediante los métodos setter de la clase articulo y añadimos el articulo a la BD con los cambios realizados
* Por ultimo en el finally cerramos la bd .



**Jform VentanaUI**

* Creamos un objeto de la clase GestorBD para llamar a los métodos
* Almacenamos en un String msg el resultado que obtenemos con el método ControlarCamposVacios de la clase controlarErrores (al ser static se llama directamente a la clase sin tener que crear un objeto de esta)
* Si msg es null significa que no hay campos vacíos entonces creamos un objeto de la clase articulo (indicándole por parámetros los TextField)
* Llamamos al método actualizar indicándole por parámetros los TextField, para que busque el articulo y si es encontrado mediante el código entonces realizar una actualización de este.
* Se ha implementado también la actualización de la lista que se va a mostrar en el TextArea mediante el método mostrar de la clase GestorBD (sobrescribe la lista actualizando la modificación del artículo)
* En caso de que obtuviéramos algún mensaje mediante el método controlarCamposVacios este enviará ese mensaje mediante una ventana emergente y no podrá actualizar el articulo indicado.

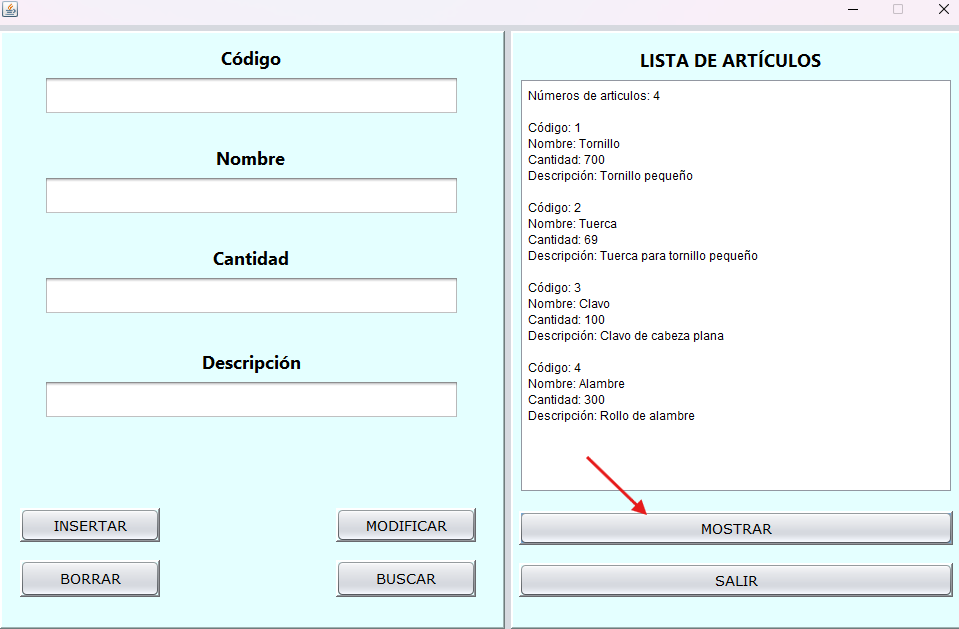


## Botón Mostrar

***Muestra dentro del cuadro de la derecha una lista con todos los datos registros en el fichero.***

### Interfaz gráfica

Al pulsar sobre este botón la función que va a desempeñar es mostrar en la lista de articulos (TextArea) todos los articulos que contiene el fichero.



### Código

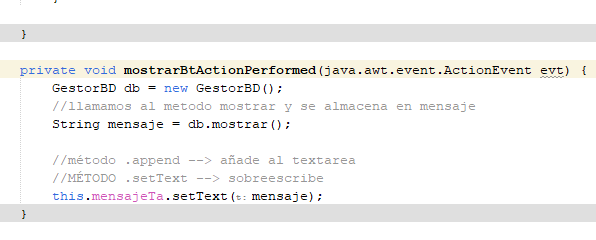
**Clase GestorBD**

* Establecemos conexión con la BD
* Creamos un objeto de la colección ObjectSet que va a recoger los articulos.
* Este comprobará si existen artículos o no
* Si no existe almacenará en la variable resultado que no se han encontrado articulos
* En cambio si existe almacenará en la variable resultado que se han encontrado (números de articulos) y entrará en el bucle while recorriendo cada articulo y recogiendo su código, nombre, cantidad y descripción.
* Todo esto será almacenado en resultado mediante una concatenación y es lo que va a devolver este método al ser llamado
* Por ultimo en el finally cerramos la bd .



**Jform VentanaUI**

* Creamos un objeto de la clase GestorBD para llamar al método.
* Llamamos al método mostrar.
* Este método va enviar un String que va a ser almacenado en la variable mensaje
* Por último la variable mensaje se va a mostrar en el TextArea mediante el método setText (para sobrescribir el contenido)

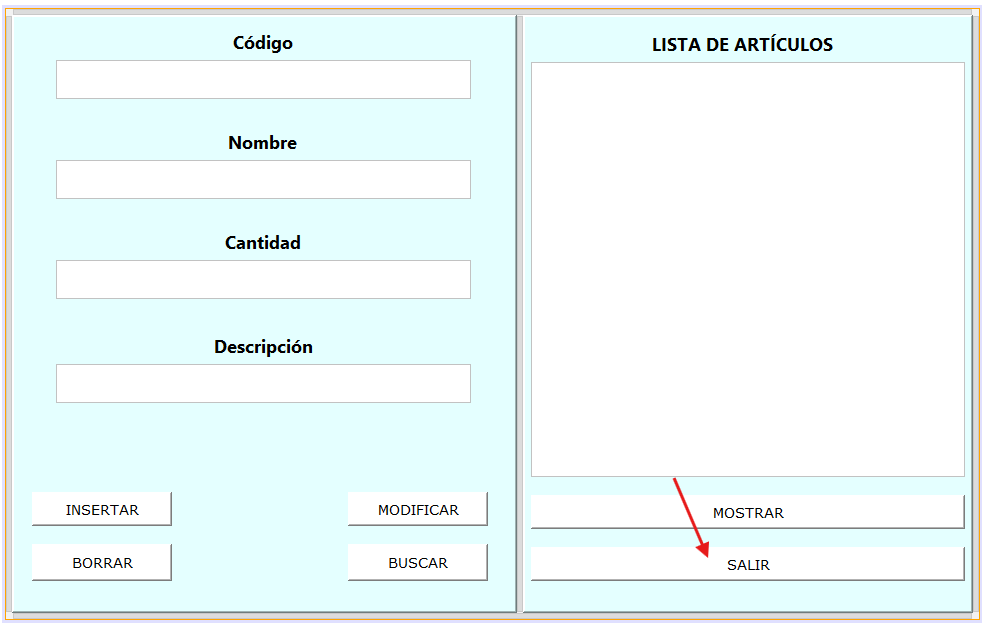


## Botón Salir

***Cierra la aplicación.***

### Interfaz gráfica

Botón para cerrar la aplicación



**Jform VentanaUI**

* Simplemente indicamos mediante System.exit(0) para salir.

