

Proyecto integrador

**Gestor De inventario**

*Matías A. Muro Díaz*

*TSDS 1º año*

Indice

[Gestor de inventario(ABM) 3](#_Toc27497)

[I. Introducción 3](#_Toc5314)

[1.1 Definición de ABM 3](#_Toc17308)

[II. Fundamentacion 3](#_Toc16715)

[2.1 Situacion problematica 3](#_Toc18613)

[2.2 Objetivos 3](#_Toc4997)

[III. Desarrollo del Programa ABM en Java 4](#_Toc10394)

[3.1 Objetivo del Programa ABM en Java 4](#_Toc32444)

[3.2 Arquitectura del Programa 4](#_Toc6754)

[3.3 Interfaz de Usuario 4](#_Toc4799)

[3.4 Conexión a la Base de Datos 5](#_Toc5992)

[IV. Ventajas del Programa ABM en Java 6](#_Toc4384)

[4.2 Mantenimiento Sencillo 6](#_Toc23057)

[4.3 Seguridad y Estabilidad 6](#_Toc13660)

[V. Proyecciones a futuro 7](#_Toc14291)

[5.1 Distincion de usuarios 7](#_Toc3930)

[5.2 Agregar un medio de facturacion 7](#_Toc22573)

[5.3 Método de acceso a datos 7](#_Toc2797)

[5.4 Mejoras de interfaz gráfica 7](#_Toc6327)

[VI. Código 8](#_Toc21549)

[6.1 Código main 8](#_Toc11899)

[6.2 Diseño de interfaz gráfica 9](#_Toc2031)

[6.3 Código de interfaz gráfica 9](#_Toc22314)

[6.4 Modelo de producto 12](#_Toc22689)

[6.5 Conexión a base de datos (singleton) 13](#_Toc16626)

*[VII. Conclusiones](#_Toc21328)* [14](#_Toc21328)

# **Gestor de inventario(ABM)**

## I. Introducción

### 1.1 Definición de ABM

#### Algunas de las ventajas del uso de ABM incluyen la reducción de errores en la gestión de datos, la mejora en la eficiencia del proceso y una mayor capacidad para realizar análisis y reportes precisos. Además, el uso de ABM permite un mayor control sobre el acceso a los datos, lo que aumenta la seguridad y privacidad de la información. En resumen, el uso de ABM es fundamental para cualquier empresa o organización que maneje grandes cantidades de información y desee mantenerla organizada y segura.

## II. Fundamentación

**2.1 Situación problemática**

Hoy en dia todos los negocios tienen un sistema de control de stock ya que a medida que el negocio crece se hace mas difícil mantener un registro de los productos que el negocio tiene y en principio esto supone contratar personal extra para mantener registro de ellos. Ademas de que el problema se multiplica al incorporar nuevas sedes y si se quisiera coordinar un stock entre distintas sedes seria casi imposible mantenerlo constantemente actualizado.

**2.2 Objetivos generales**

En resumen, con este programa se propone:

* Reducir el personal necesario para estas tareas repetitivas
* Reducir el tiempo de acceso a la información de productos
* Conectar múltiples sucursales

**2.3 Objetivos especificos**

Estos objetivos se cumplen a corto plazo

* Administrar productos
* Tener interfaz amigable para el operador
* Persistir los productos en una base de datos
* Respaldar información
* Reducir la posibilidad de error humano
* Centralizar la información

## III. Desarrollo del Programa ABM en Java

### 3.1 Objetivo del Programa ABM en Java

#### El propósito principal de este programa es proporcionar una interfaz intuitiva y eficaz para interactuar con una base de datos, permitiendo realizar operaciones de inserción, eliminación y modificación de registros de manera sistemática y organizando la información en un punto de acceso común para múltiples puestos de trabajo. Eliminando muchas tareas repetitivas y agilizando cuestiones de papeleo en cuanto al stock al momento de una compra

### 3.2 Arquitectura del Programa

El programa está desarrollado en Java, aprovechando las ventajas de la programación orientada a objetos. Se estructura en módulos que facilitan la comprensión y mantenimiento del código, garantizando la escalabilidad del sistema.

Destacando la aplicación del patrón de diseño singleton para mantener un único acceso a la base de datos por cada usuario, evitando superposición de los datos ingresados y una distribución en capas para mantener separados a la capa de presentación y a la capa de repositorio con acceso a la base de datos

### 3.3 Interfaz de Usuario

Se ha diseñado una interfaz de usuario intuitiva mediante la librería Swing propia de java, mediante el constructor gráfico integrado en el IDE Netbeans, esta interfaz permite a los usuarios interactuar fácilmente con la base de datos sin tener conocimiento de manejo de la consola o terminal.

La interfaz proporciona formularios para la inserción de datos, opciones de búsqueda y de eliminación por medio del código único de un producto.

### 3.4 Conexión a la Base de Datos

Para asegurar la protección de la información almacenada en la base de datos, se ha implementado una conexión segura que hace uso de las capacidades de JDBC (Java Database Connectivity). Esto garantiza la compatibilidad con una base de datos relacional MySQL mediante consultas personalizadas, lo que permite un acceso seguro y eficiente a los datos. De esta manera, se puede trabajar con confianza en la manipulación y gestión de los registros almacenados en la base de datos sin temor a errores en el proceso.

## IV. Ventajas del Programa ABM en Java

La elección de Java como lenguaje de programación tiene múltiples ventajas, entre ellas la portabilidad y versatilidad que ofrece. Gracias a la portabilidad, los programas desarrollados en Java pueden funcionar en diferentes plataformas sin necesidad de modificar el código fuente, lo que permite un uso más eficiente del tiempo y recursos invertidos en el desarrollo. Además, esta característica garantiza la versatilidad del programa en diferentes tipos de hardware, asegurando su funcionamiento óptimo sin importar las especificaciones técnicas del dispositivo utilizado.

### 4.2 Mantenimiento Sencillo

La estructura modular del programa facilita las actualizaciones y el mantenimiento. Las modificaciones o expansiones pueden realizarse de manera eficiente sin afectar otras partes del sistema.

### 4.3 Seguridad y Estabilidad

Java proporciona mecanismos robustos de seguridad, garantizando la integridad de los datos y protegiendo contra posibles amenazas. La estabilidad del programa se ve reforzada por la gestión de excepciones y la detección temprana de errores.

Ademas de integrar una arquitectura de capas que permite la independencia de cada paquete

## V. Proyecciones a futuro

### 5.1 Distincion de usuarios

Se puede proponer una distincion entre los administrativos, quienes tendrán acceso a borrar y modificar y agregar los datos de la base de datos, y los vendedores quienes podrán realizar consultas y en caso de realizar una venta reducir el numero de stock

### 5.2 Agregar un medio de facturacion

Puede incorporarse un servicio de un tercero para registrar e imprimir facturas y así mantener registro de las ventas y su monto desde un punto de vista fiscal.

### 5.3 Método de acceso a datos

Por fines de mantenibilidad se podría utilizar un framework de mapeo objeto relacional (ORM) para mantener la conexión de la base de datos, dejando de lado las consultas que puedan contener errores de tipeo y facilitando la lectura del código por parte de nuevos programadores.

### 5.4 Mejoras de interfaz gráfica

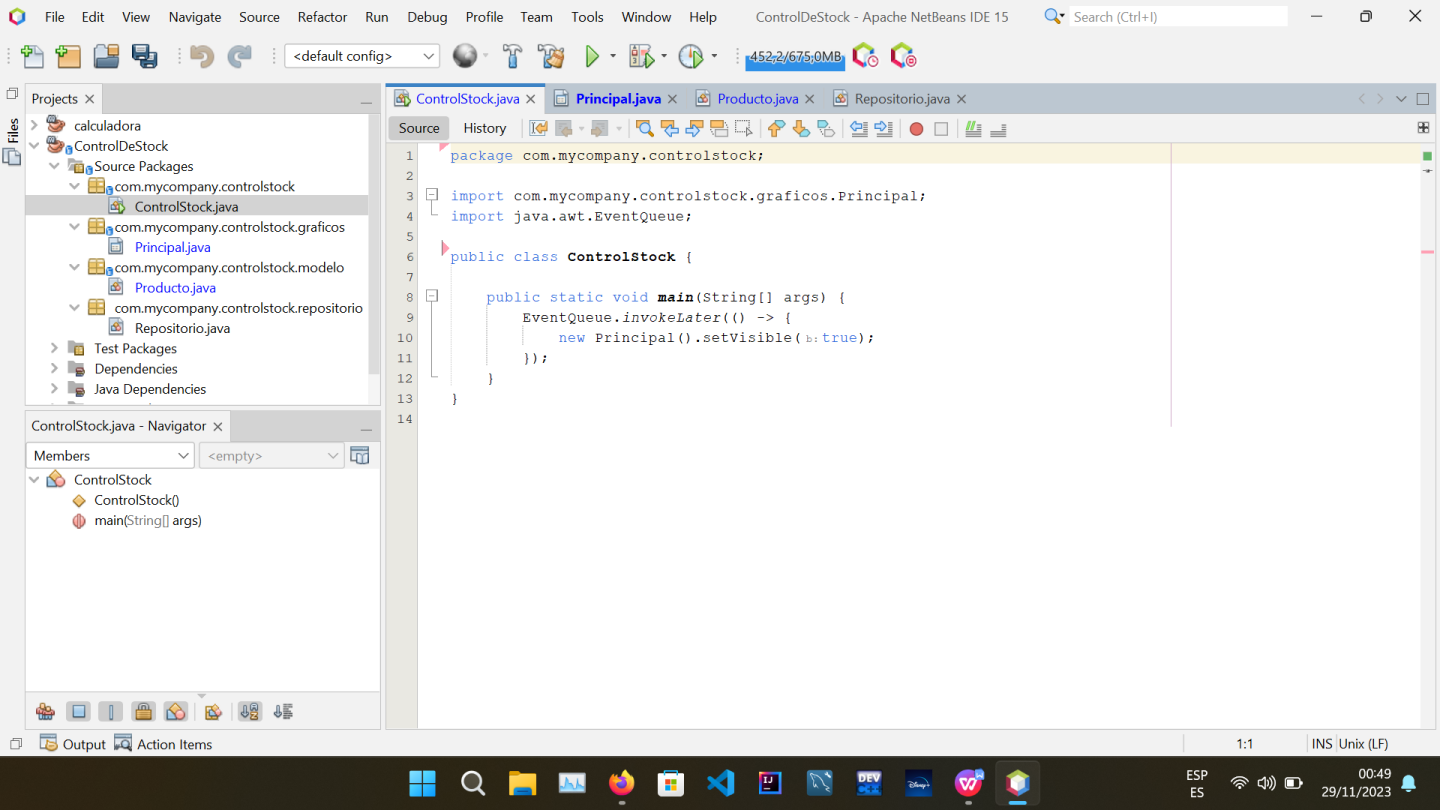
La interfaz actual es funcional y facilita mucho el uso, sin embargo podrían utilizarse librerías gráficas mas estéticas y que admitan accesibilidad mejorada para personas que lo requieran.

## VI. Código

A continuación adjunto imágenes del código empleado para este proyecto, cómo extra también adjunto el link al repositorio de github <https://github.com/agustindiaz59/ISET>

### 6.1 Código main

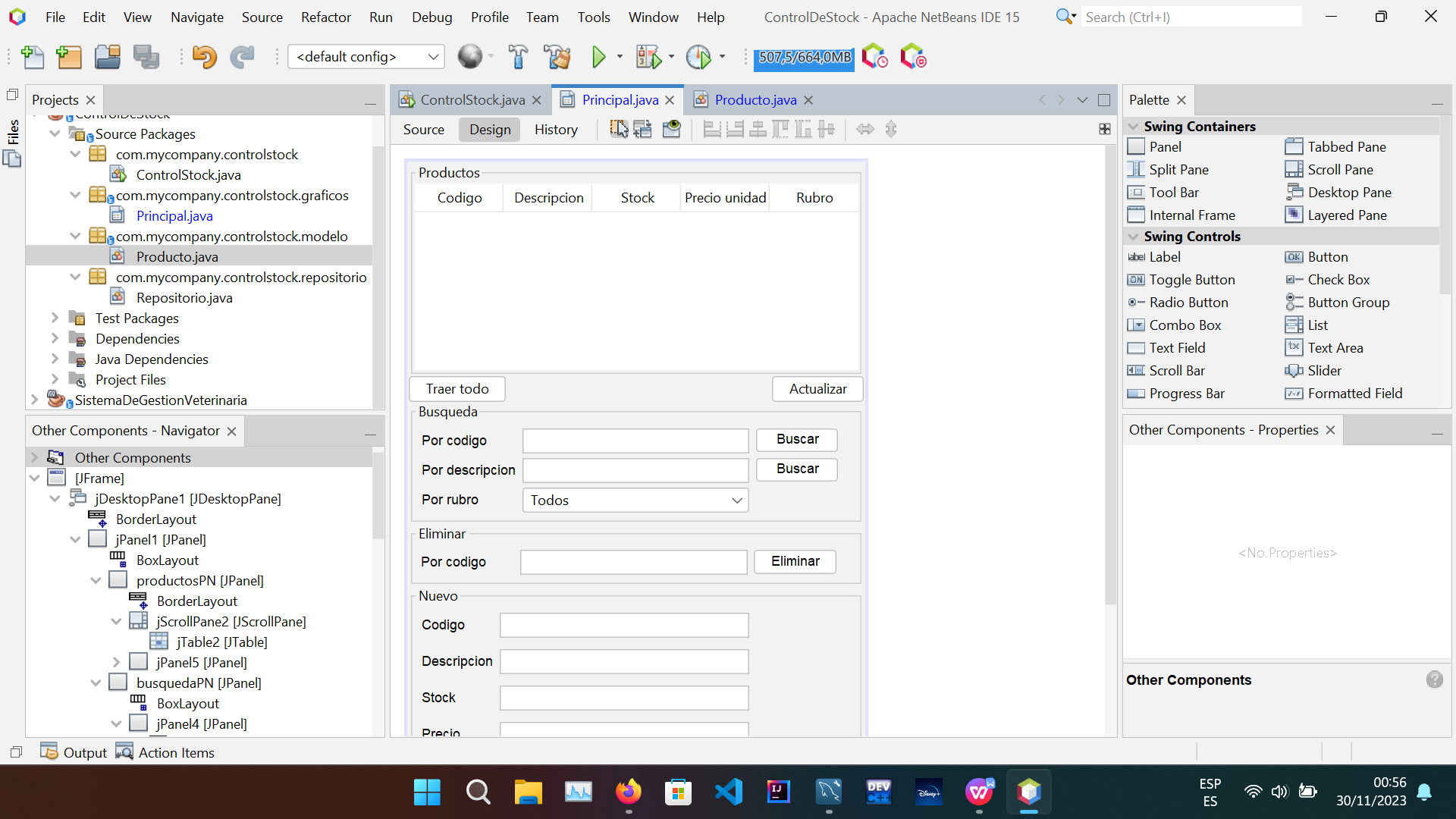
En esta clase lo único que hice fue llamar a la interfaz gráfica, donde se ejecutara el resto de la lógica.



### 6.2 Diseño de interfaz gráfica

En esta sección Netbeans nos permite diseñar la interfaz gráfica que deseemos mediante ‘Drag and Drop’, es decir arrastrando los componentes a la posición deseada, permitiéndonos elegir tanto componente como contenedores disponibles en la librería de Swing incluida en Java.

Ademas podemos manejar el código fuente de cada componente y el manejo de eventos que ocurran durante la ejecución, como pueden ser los clics, presionar teclas, cerrar la ventana, etc.



***6.3 Lógica de la interfaz gráfica***

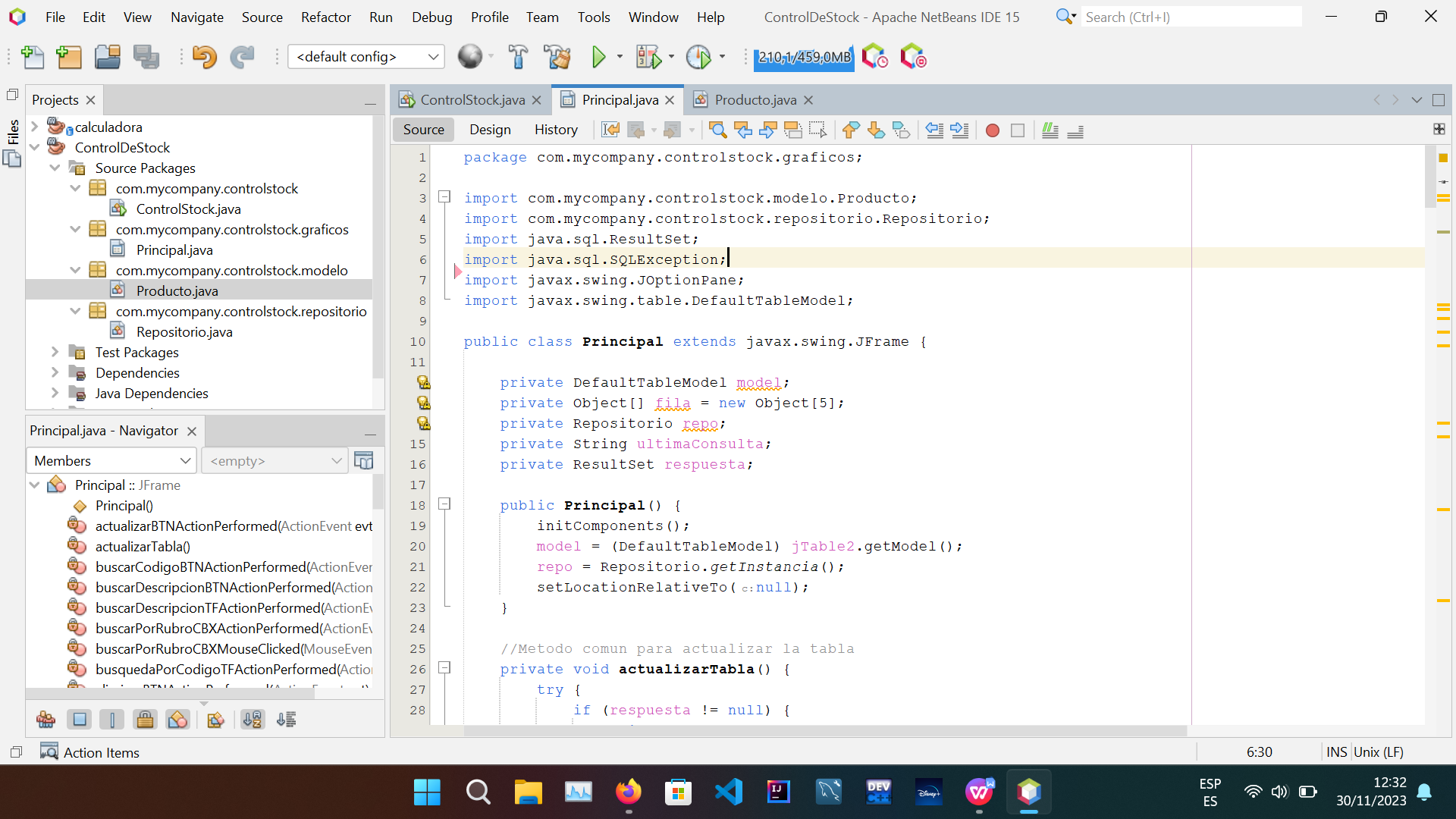
En esta sección esta definida gran parte de la lógica del programa, cada componente de la interfaz gráfica esta enlazado con una función que es llamada cuando ocurre un evento genérico sobre ese componente.

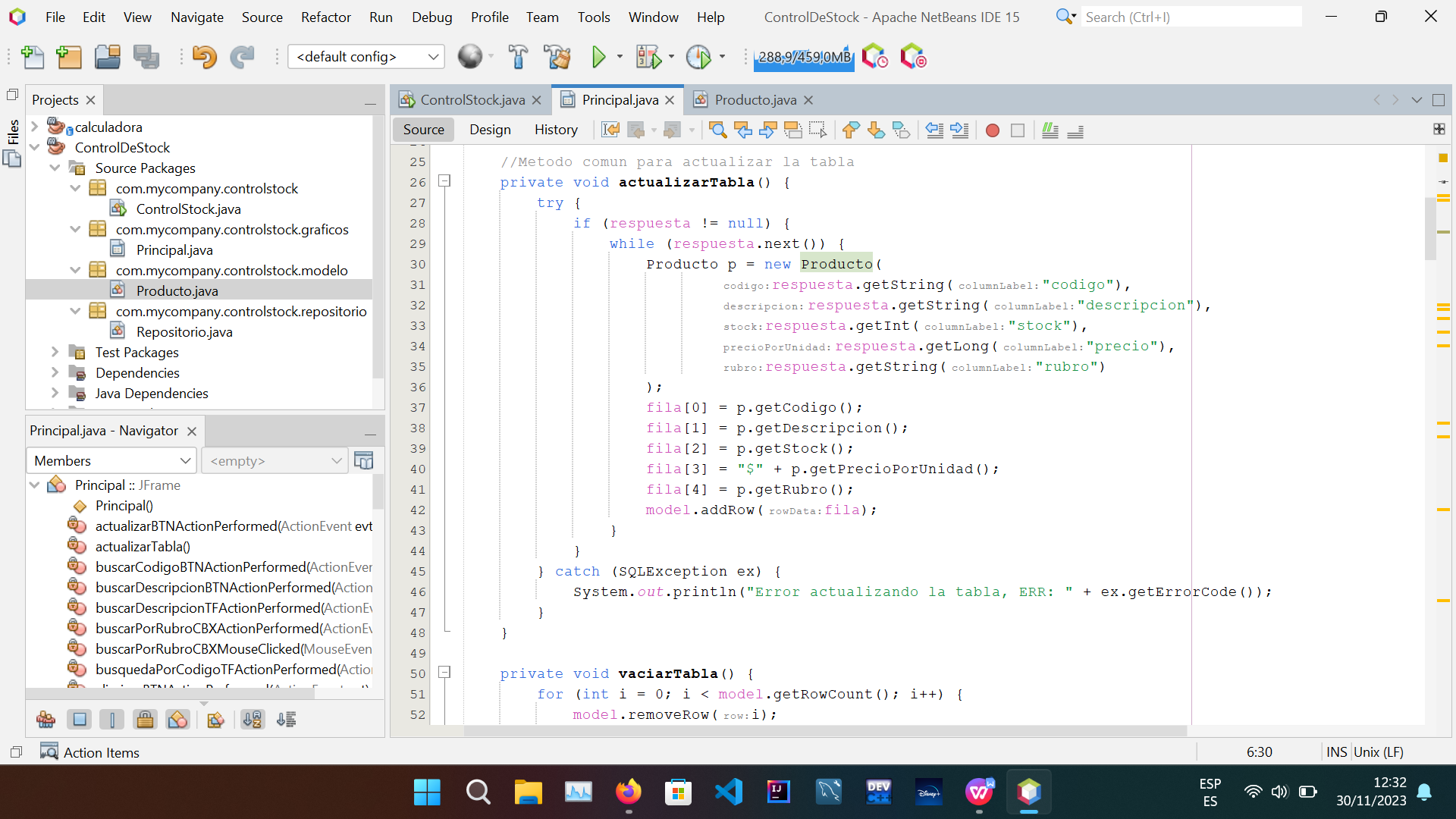
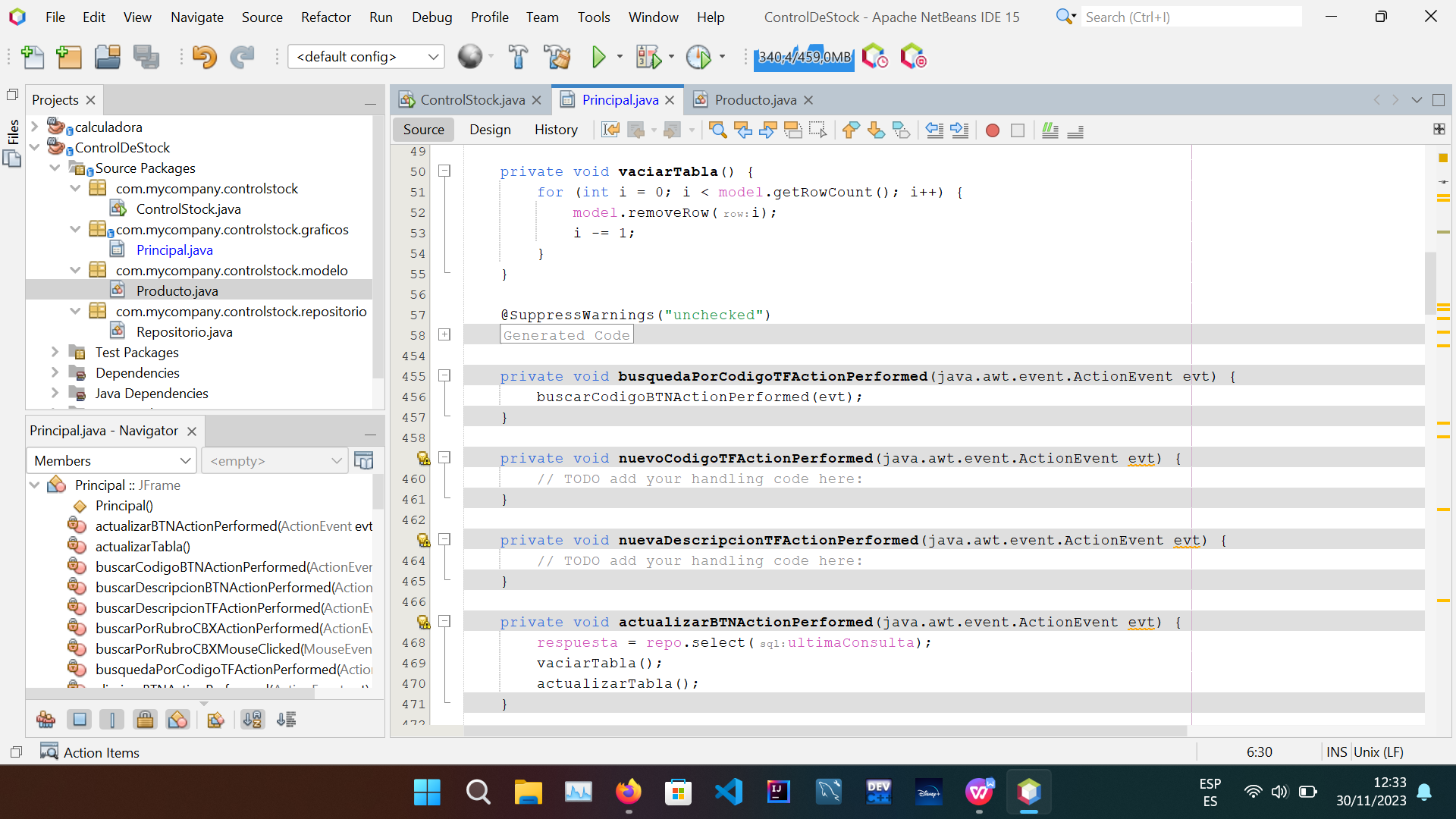
Antes de todas las funciones tenemos los atributos, en este caso son utilizados para iterar sobre la grilla de datos, y para conectarse a la base de datos mediante otra clase.

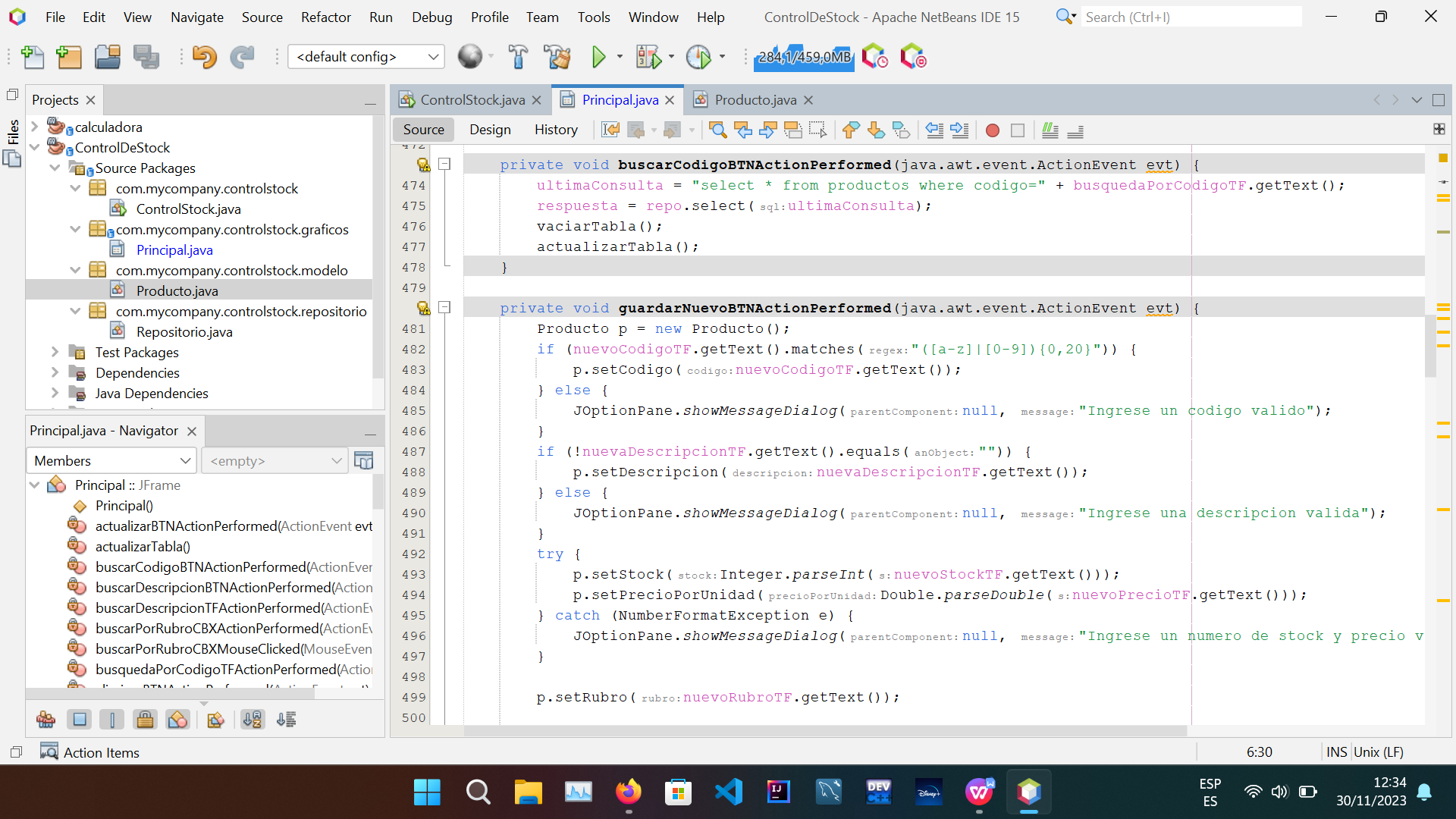
Luego tenemos el constructor que se encarga de llamar a la función que inicializa a todos los componentes, esta función la define automáticamente en Netbeans y no puede modificarse o rompería la integridad del proyecto.

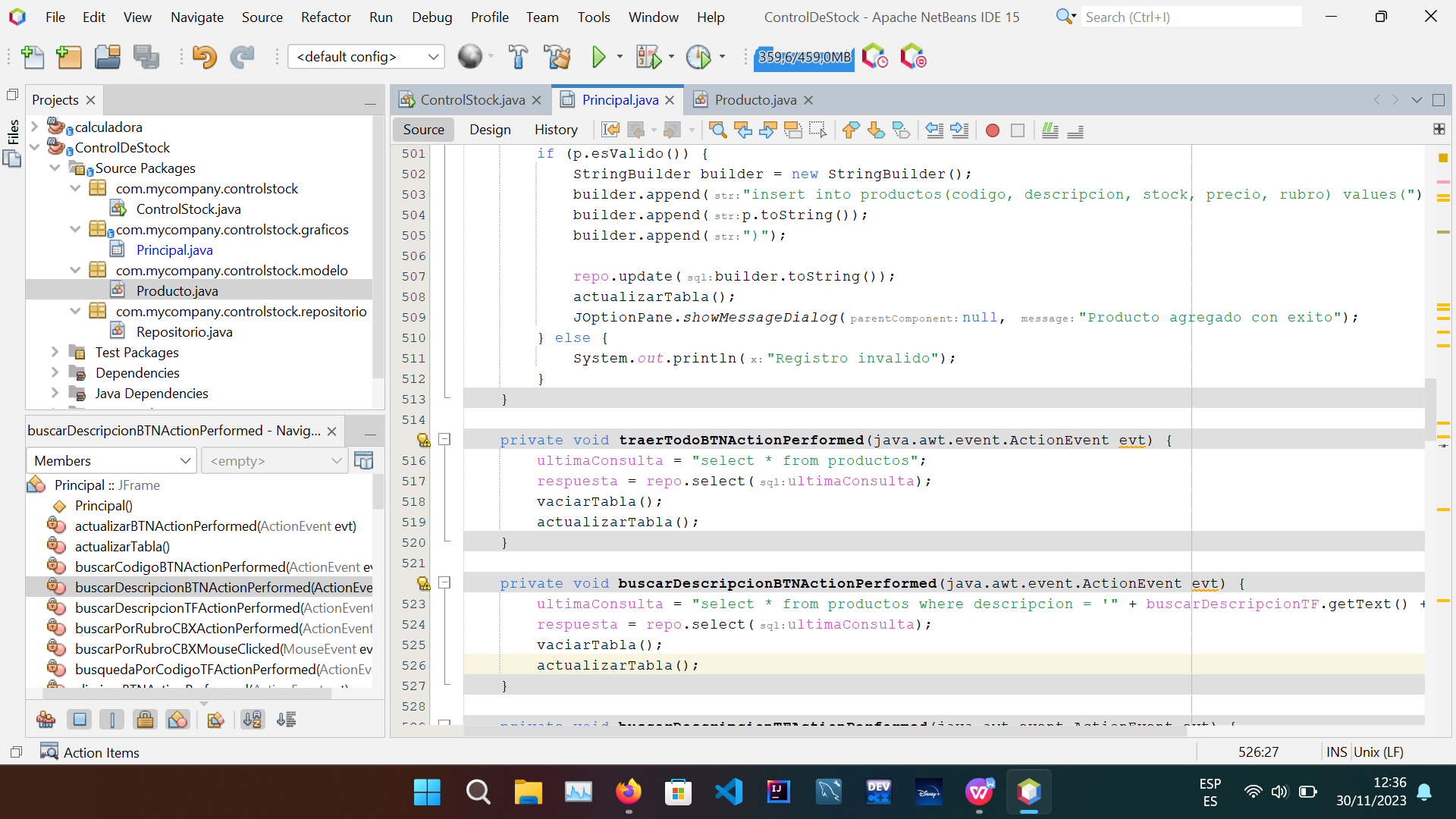
La mayoría de las funciones son auto descriptivas, aunque caben destacar algunas:

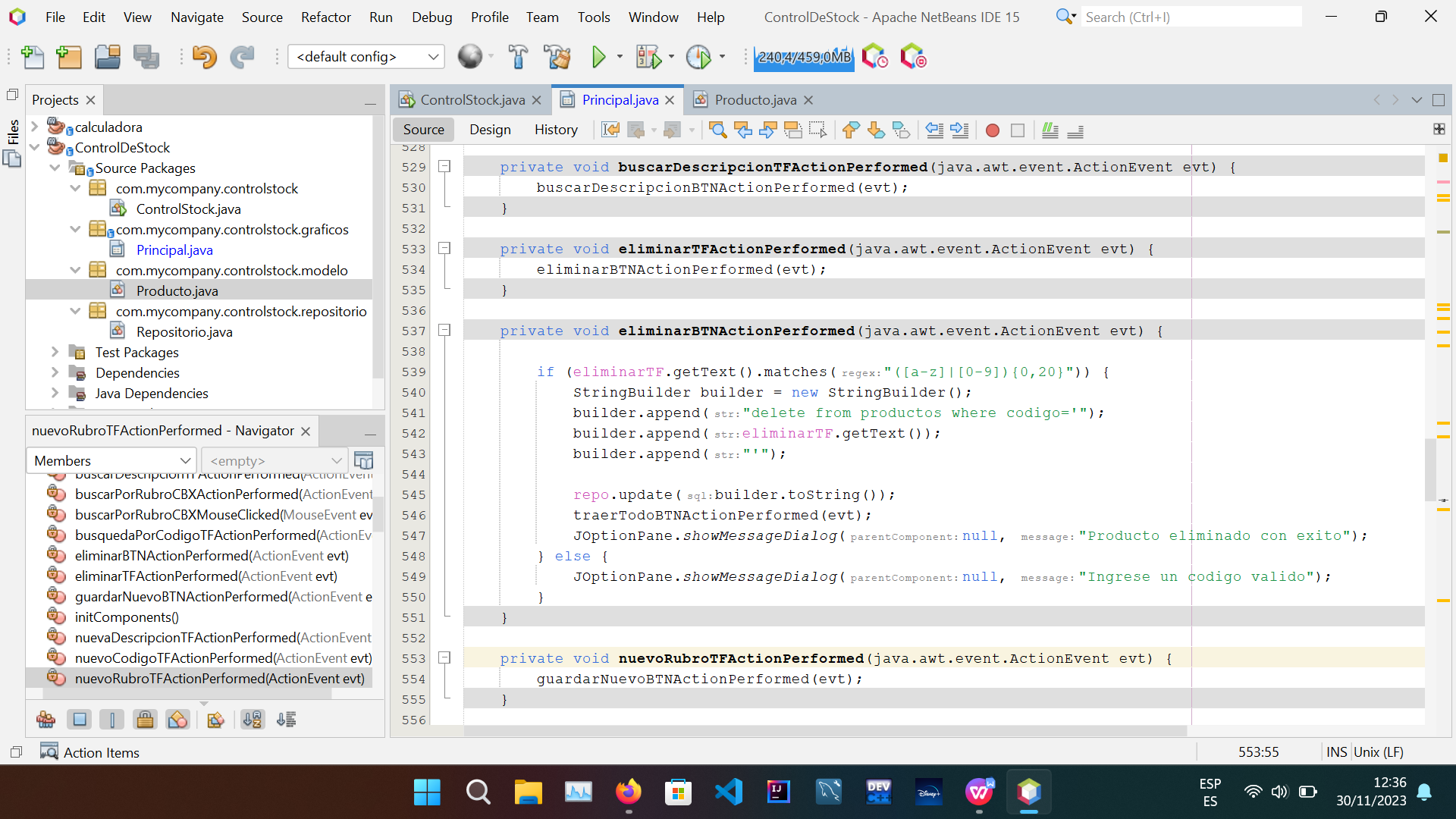
* **actualizarTabla()**: es la responsable de actualizar los datos de la tabla, la cual utiliza los atributos correspondientes a la base de datos para hacer la consultas y utilizando esos datos para escribir sobre la grilla los respectivos campos.
* **guardarNuevoBTNActionPerformed():** esta función es la encargada de tomar el texto de los ‘text field’ y validarlos con los datos que la base de datos espera recibir, y arrojando excepciones en caso de que se encuentre un error. Si no se encuentran errores se realizara un insert en la base de datos con el nuevo producto.
* **eliminarBTNActionPerformed()**: Similar a la anterior, esta se encarga de tomar el texto del text field de eliminar y validarlo, si el texto ingresado tiene un formato valido se realizara un delete borrando el producto con el respectivo código.
* **buscarPorRubroCBXActionPerformed()**: esta función esta destinada al ‘Combo box’ para buscar productos filtrándolos por alguno de los rubros disponibles.

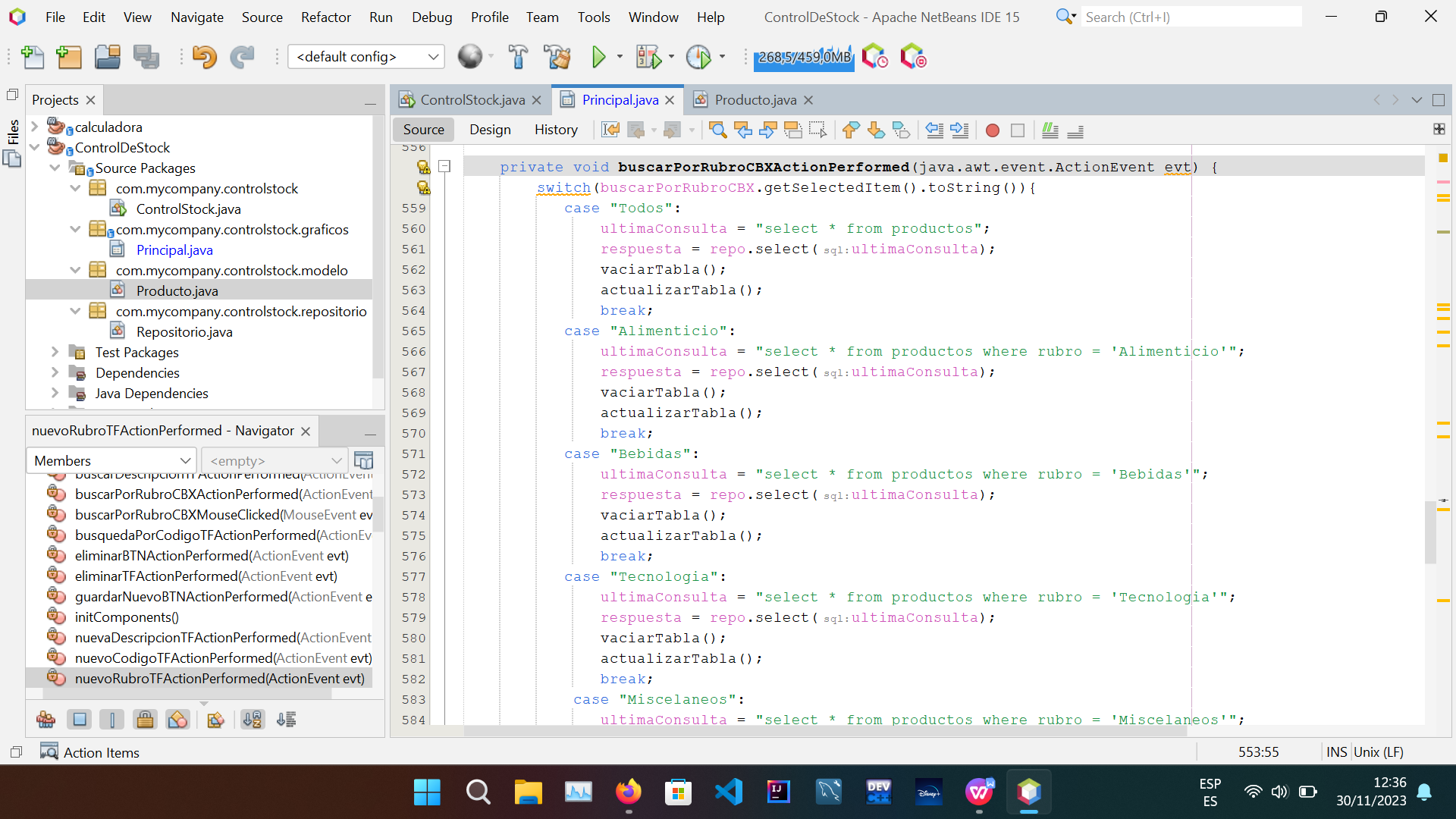


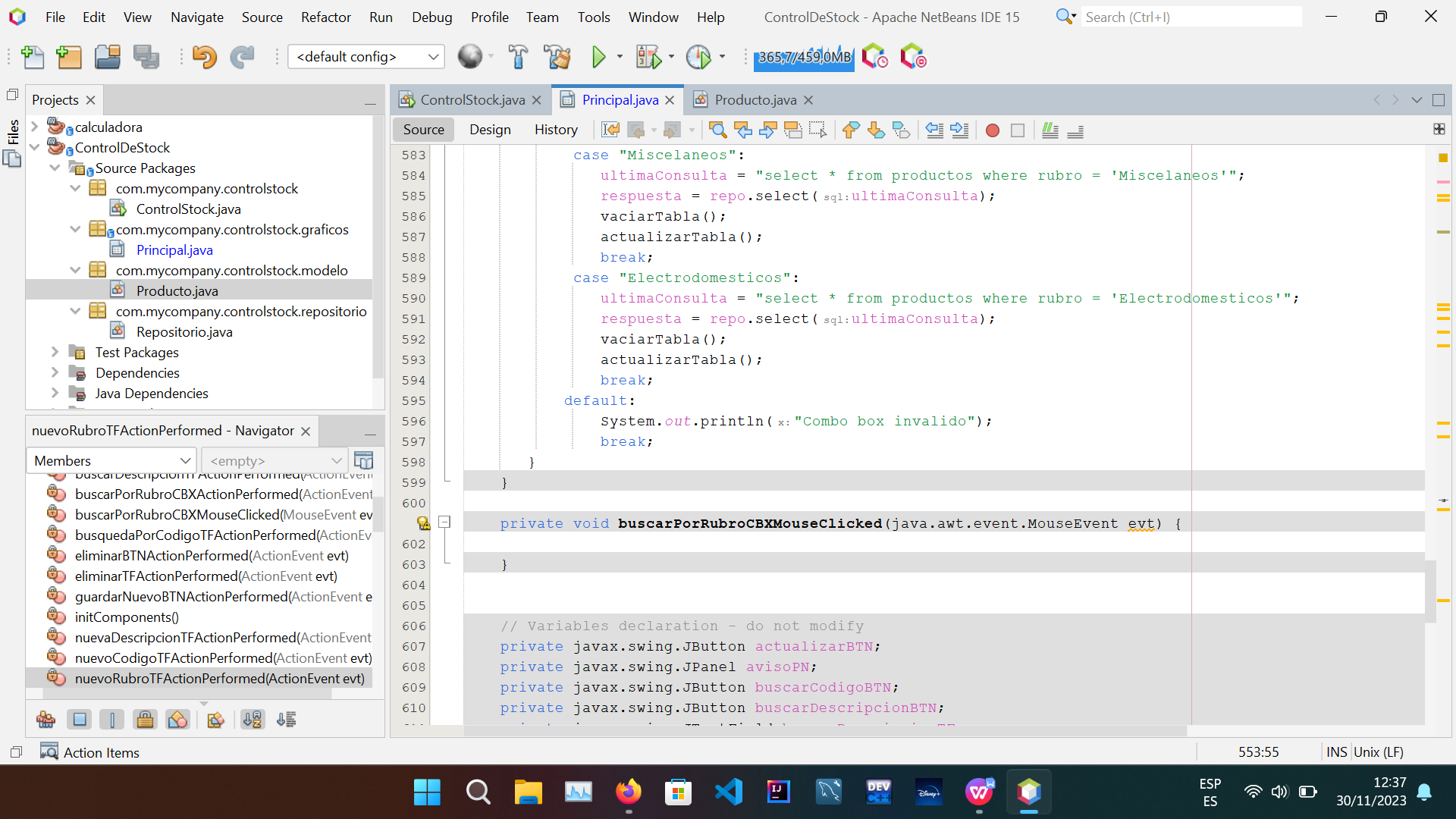






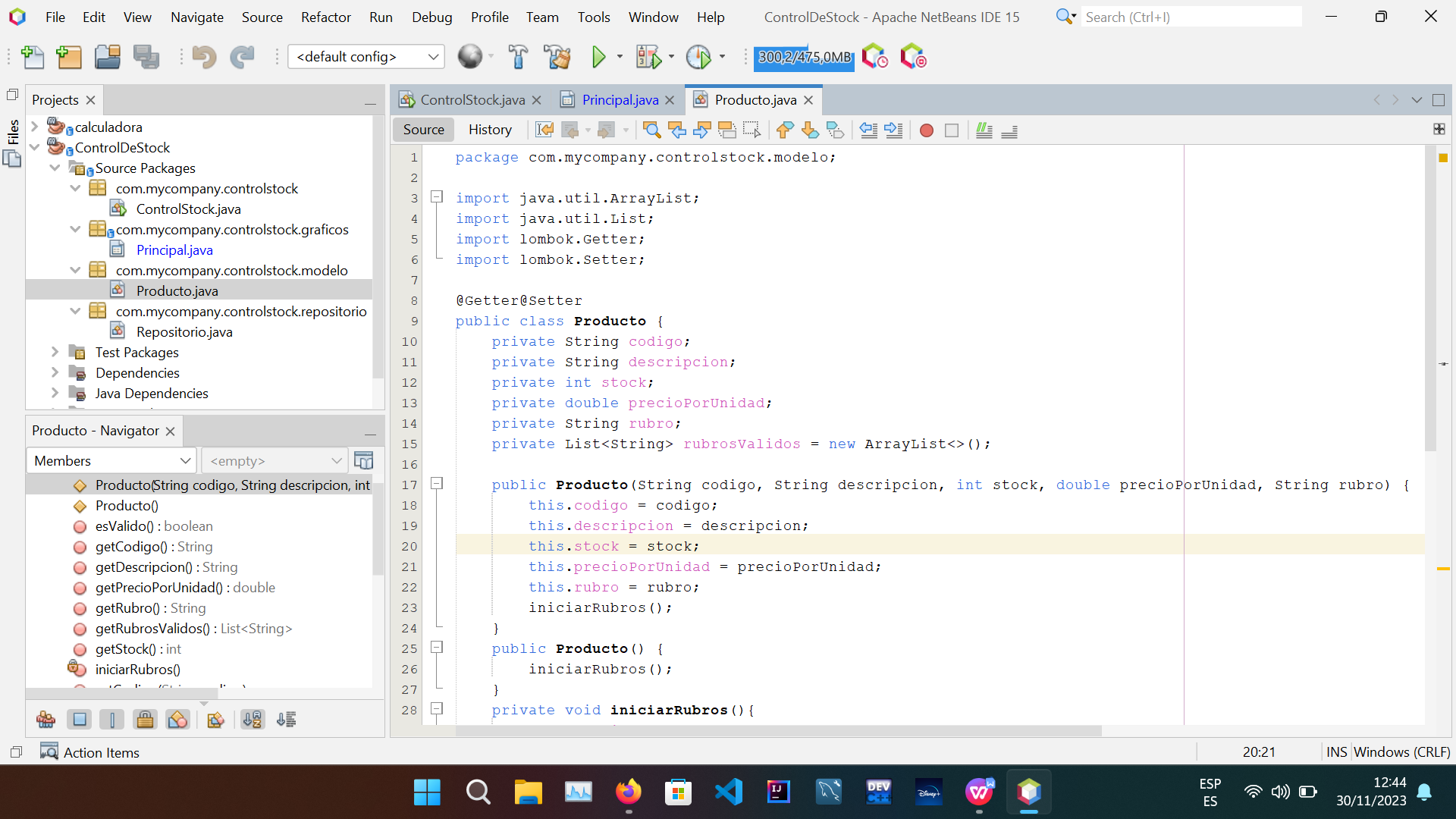


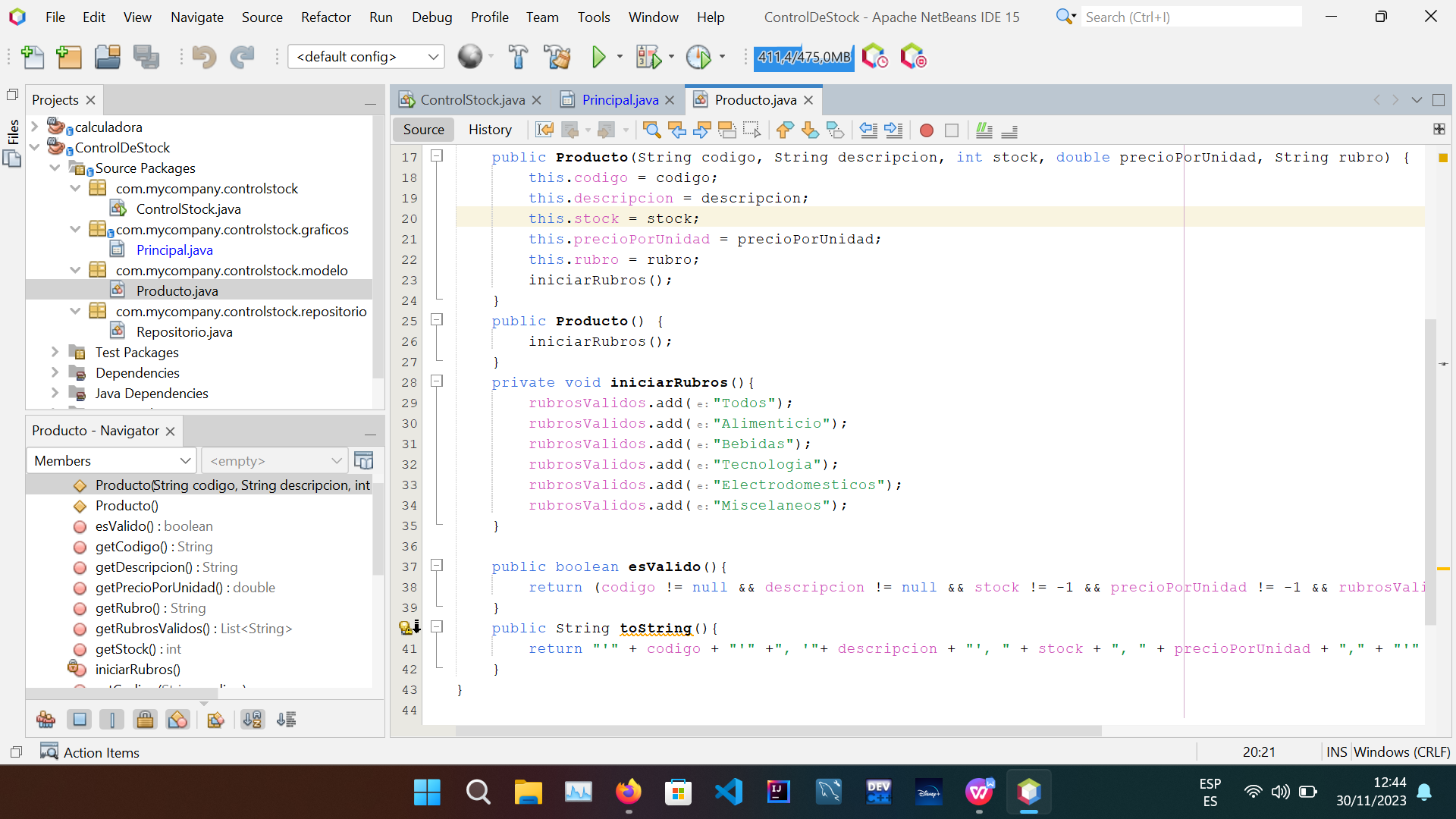




***6.4 Modelo de producto (Java)***

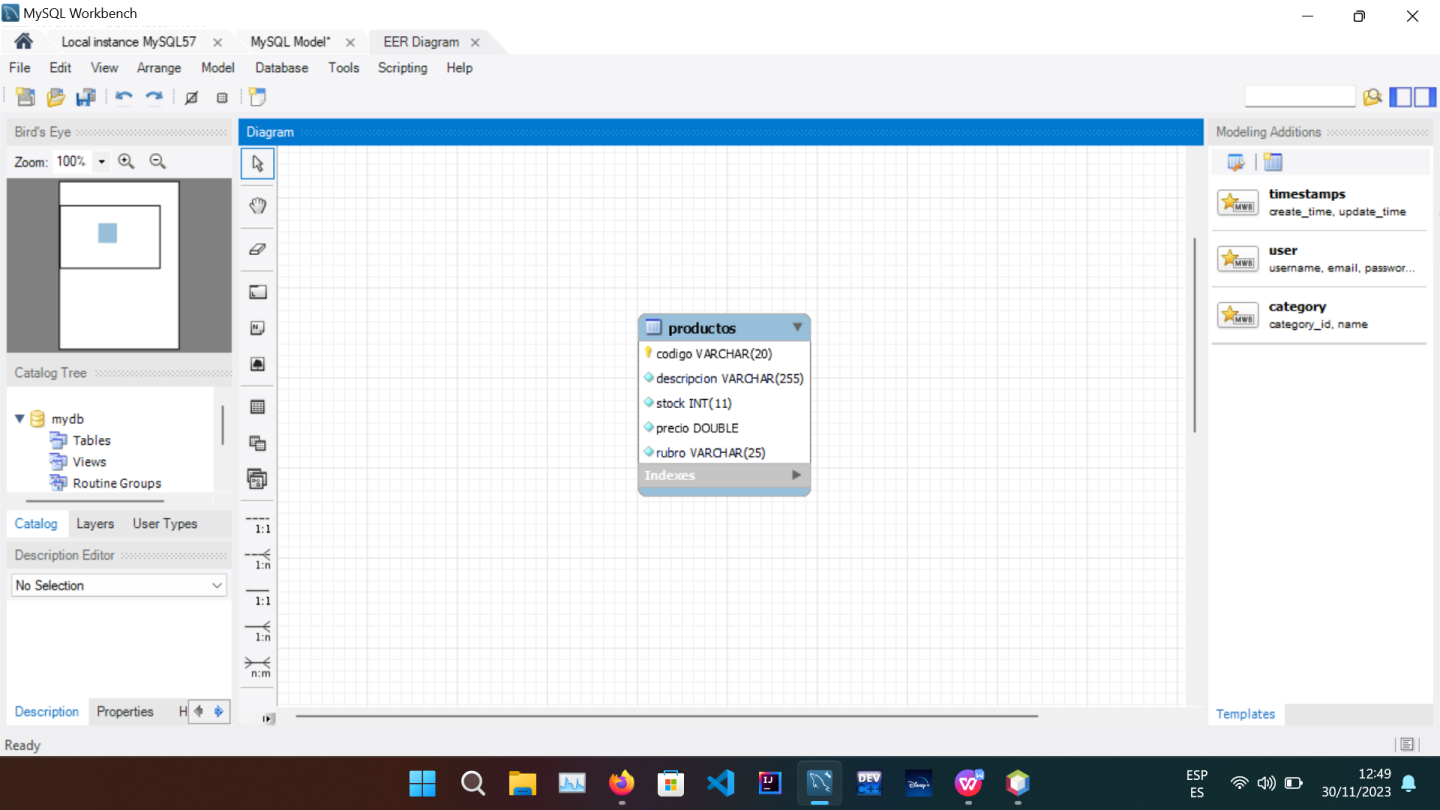
En esta parte defino los atributos que tendrá cada producto para poder manipularlo desde la lógica, ademas de sus constructores y un métodos toString para poder realizar las consultas, un método para validar que los campos estén llenados correctamente antes de enviarlos en una consulta SQL, y por ultimo un método en el que se agregan la lista de rubros validos para luego validarlos.

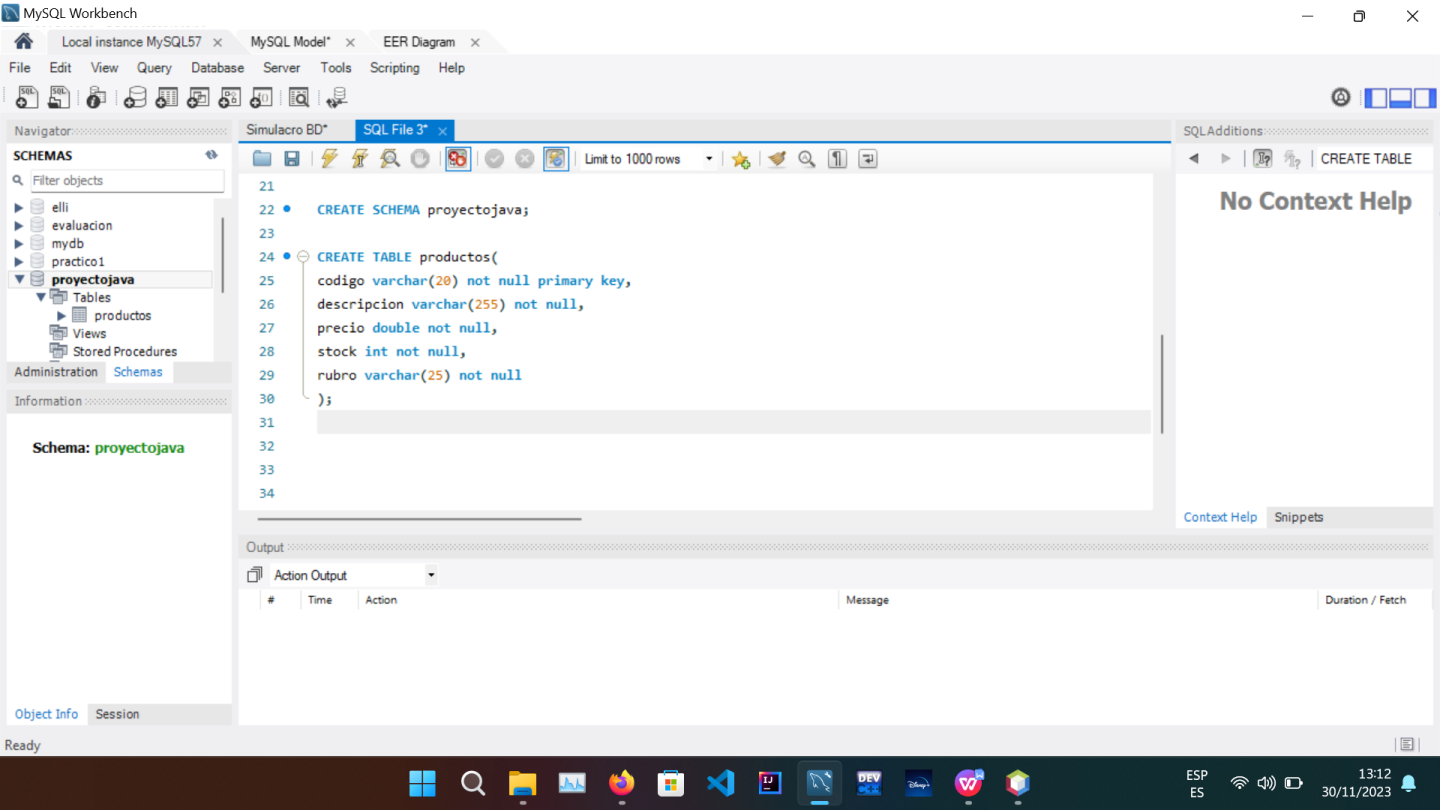




***6.5 Modelo relacional (MySQL)***

En este caso utilizo una base de datos MySQL para almacenar los productos, no considere necesario relaciones ni foreign keys al ser una única entidad con atributos propios, reflejándose como una única tabla de productos.





***6.6 Modelo Entidad-Relacion (DER)***

El DER es útil cuando se desea abstraerse del motor de datos a utilizar y se desea representar las entidades y las relaciones entre ellos.

descripción

código

PRODUCTOS

Rubro

Precio

Stock

***VII. Conclusiones***

En conclusión, este programa es útil para mantener un control centralizado del stock de un negocio, actualizando los valores en tiempo real, agilizando ventas, facturaciones y demás tramites internos que dependan del stock y precio actual de los productos registrados.

Su interfaz gráfica simplifica el uso sin necesidad de aprender a usar la consola, y encontrando todos los botones mas fácilmente para un usuario sin experiencia.

La base de datos puede persistir tantos productos como se deseen incluso cuando el programa se finalice o la computadora se apague.

***Bibliografia***

***Glosario IT. (s.f.). ABM.*** Recuperado de [https://www.glosarioit.com/ABM](https://www.glosarioit.com/ABM" \t "https://chat.openai.com/c/_new)

***Baeldung. (s.f.).*** Recuperado de [https://www.baeldung.com/](https://www.baeldung.com/" \t "https://chat.openai.com/c/_new)

***Refactoring Guru. (s.f.).*** Recuperado de [https://refactoring.guru/](https://refactoring.guru/" \t "https://chat.openai.com/c/_new)

***YouTube. (s.f).*** Recuperado de <https://www.youtube.com/@TodoCode>