Selene Pacheco Rodríguez

RA01. UD01. Gestión de la información almacenada en Ficheros

Índice:

1- Introducción

1.1- Descripción del proyecto

Este proyecto es un gestor de productos desarrollado en Java utilizando JavaFX. Permite a los usuarios gestionar un inventario de productos, incluyendo la adición, edición, eliminación y visualización de productos de manera eficiente y amigable. También permite guardar una copia de los productos en formato CSV o en formato JSON, así como importar tus propios productos en formato CSV o formato JSON.

1.2- Objetivos y beneficios

Los objetivos del proyecto eran simples: crear una aplicación que gestionase una lista de productos en un fichero XML.

Los usuarios obtendrán los siguientes beneficios usando este programa:

- Podrán acceder a un sistema de gestión sencillo que reduce la posibilidad de errores manuales, mejora la eficiencia operativa y proporciona una visión clara del estado del inventario.
- Podrán exportar e importar sus productos en formato JSON o CSV, consiguiendo así que la transición de otra herramienta de gestión de productos a la nuestra sea más sencilla.

1.3- Tecnologías utilizadas

Hemos utilizado las siguientes tecnologías:

- Lenguaje de programación: Java.
- Marco de trabajo: JavaFX para la creación de la interfaz gráfica.
- Almacenamiento de datos: Se usan archivos XML para el almacenamiento de la información.
- Bibliotecas para XML: JAXB para la serialización y deserialización de objetos Java a XML y viceversa.

1.4- Ventajas y los inconvenientes de las distintas formas de acceso

Se ha utilizado la biblioteca JAXB en lugar de usar DocumentBuilder y DOM dado que es la tecnología que hemos aprendido en este tema y se ha querido plasmar en el proyecto. Sin embargo, hay que tener en cuenta que para el uso que hemos hecho, hubiese sido más recomendable el uso de DocumentBuilder y DOM, dado que permiten la manipulación directa del XML, tiene mayor rendimiento con grandes volúmenes de información y tiene un mayor control sobre el formato del XML. Si el proyecto albergase muchos productos, su rendimiento se vería bastante reducido con el uso de JAXB en comparación con el uso de DocumentBuilder y DOM.

1.5- Requisitos del sistema

Como es un programa bastante sencillo, únicamente sería necesario tener instalada la JVM(Java Virtual Machine) para que el programa pudiese correr en la máquina. Por el restos de requisitos, la mayoría de los equipos podrían correr este programa sin problema.

2- Arquitectura del proyecto

2.1- Diagrama general de la arquitectura.

El siguiente diagrama muestra la arquitectura general del proyecto. Ir a ANEXO I.

Elementos del diagrama:

- Modelo o Business Logic Layer: En el modelo encontramos las clases Producto y Productos, las cuales constituyen el modelo. Es la representación lógica del negocio y la estructura de datos de la aplicación. En esta sección se definen las entidades del proyecto.
- Vista: La vista son aquellos ficheros con extensión .fxml que definen la vista. En el diagrama de arquitectura no salen al no ser clases Java. Debe haber una vista por cada pantalla. En nuestro caso solo tenemos la vista productos.fxml, controlada por el controlador ProductosController.

- Controlador: El controlador es la clase ProductosController en este caso. Debe haber tantas clases Controller como vistas .fxml existan. El controlador está en la parte de Presentation Layer en el diagrama de arquitectura, indicando que es parte de la vista, dado que la controla(aunque realmente no es parte de la vista, pero el diagrama de aplicación únicamente incluye las clases Java).

2.2- Explicación del Patrón MVC

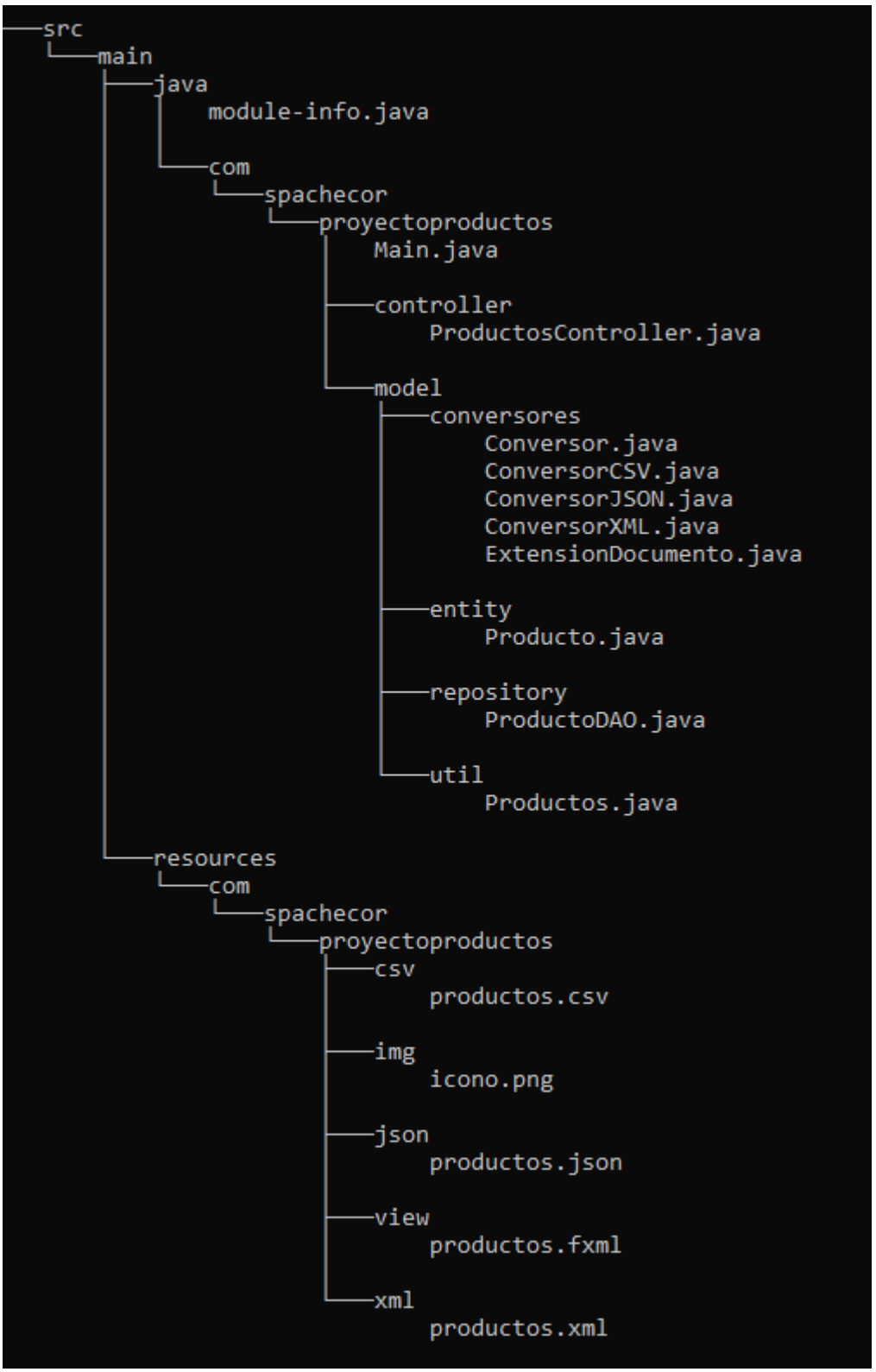
El patrón Modelo-Vista-Controlador (MVC) es un patrón arquitectónico que separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. Esta separación permite una mejor organización del código, facilitando su mantenimiento y escalabilidad. El patrón MVC, contiene los siguientes elementos:

- Modelo: El modelo representa la estructura de los datos y la lógica de negocio. En este proyecto, el modelo contiene clases que definen las entidades, así como métodos para acceder y manipular los datos, como agregar, editar o eliminar productos.
- Vista: La vista es responsable de la presentación de los datos al usuario. Utiliza componentes de JavaFX, como Scene, Stage, Button, TableView, etc., para crear una interfaz amigable y visualmente atractiva.
- Controlador: El controlador recibe las entradas del usuario desde la vista y ejecuta las acciones necesarias en el modelo. Por ejemplo, cuando un usuario hace clic en un botón para agregar un nuevo producto, el controlador recoge los datos ingresados, actualiza el modelo y luego actualiza la vista para reflejar los cambios.

3- Estructura del proyecto

3.1- Ubicación del patrón MVC en el proyecto y descripción de la estructura

La estructura del proyecto sigue una organización estándar basada en el patrón MVC, que ayuda a mantener el código limpio y modular. A continuación se presenta un esquema de la estructura de carpetas y archivos:



En este esquema, podemos ver claramente cada sección separada, teniendo el modelo en src/main/java/com/spachecor/proyectoproductos/model, la vista en src/main/resources/com/spachecor/proyectoproductos/view y el controlador en src/main/java/com/spachecor/proyectoproductos/controller.

3.2- Explicación de cada paquete

En la parte de src/main/java, que es donde tenemos nuestras clases Java, podemos ver claramente las siguientes secciones: controller(controlador del MVC) y model(modelo del MVC). Sin embargo, dentro del modelo encontramos una serie de utilidades:

- Conversores: el paquete conversores contiene los conversores de XML a JSON, JSON a XML, XML a CSV y CSV a XML.
- Entidad: en el paquete entity encontramos la clase Producto, que es el molde de los productos.
- Repositorio: en el paquete repository encontramos la clase ProductoDAO, que es una clase tipo DAO o Data Access Object. Cuando hablamos de DAO estamos hablando de un patrón de diseño muy clásico en el cual una clase se encarga de las operaciones de persistencia contra una tabla de la base de datos. Sin embargo, en este caso, la persistencia la haríamos en un XML en vez de en una base de datos.
- Utilidades: en el paquete util encontramos la clase Productos, que es una colección de productos. Su utilidad es literalmente eso, ser una colección de productos.

3.3- Notas sobre la organización del código

3.3.1- Consistencia en la nomenclatura

Se sigue una convención de nomenclatura consistente para las clases y métodos, facilitando la comprensión del código. Por ejemplo, las clases se nombran con mayúscula inicial (Producto, Proveedor), y los métodos utilizan notación camelCase (agregarProducto, eliminarProducto).

3.3.2- Documentación

Se incluye documentación JavaDoc en las clases y métodos, proporcionando descripciones claras de su funcionalidad y uso. Esto ayuda a otros desarrolladores (o a ti mismo en el futuro) a comprender rápidamente el propósito de cada parte del código.

4- Modelo

4.1- Descripción de las entidades

En este proyecto la entidad principal (y única) es el Producto. Un producto contiene nombre, precio y cantidad. Dos productos son iguales entre sí si tienen el mismo nombre. La clase Producto es una clase sencilla, que incluye los atributos que he mencionado con su constructor, sus getters y sus setters, además de los métodos sobreescritos (de la clase Object) equals(para saber cuando 2 objetos son iguales) y toString(que modifica como se muestra el objeto por consola). También implementa la interfaz Comparable y su método compareTo(también destinado a decidir cuando dos productos son iguales). Por otro lado, también tiene las anotaciones necesarias para el funcionamiento de la biblioteca JAXB.

En utilidades, la clase Productos es una colección de productos, que ha sido creada para el uso de la biblioteca JAXB. Esta clase define una lista de productos, y también la forma de buscar y agregar productos a la colección Prodcuctos, así como la posibilidad de obtener la lista de los productos en un objeto tipo List<>

En repository tenemos la clase ProductoDAO, que se crea como un CRUD de los productos en el fichero XML que hace de registro de los productos. Contiene la opción de agregar, eliminar, actualizar y listar los productos (en la clase Productos podemos buscar un producto en concreto de la colección de productos), así como un método que se encarga de construir el XML cuando se tenga la lista de Productos.

En los conversores tenemos la clase abstracta Conversor, de la que heredan el ConversorXML, ConversorCSV y Conversor JSON. La clase Conversor define la estructura del método abstracto convert, que es el que convierte un fichero de un formato a otro y que lo implementan sus hijos(ConversorXML, etc.), además de un método para obtener la extensión de un documento. Los conversores que implementan esta clase Conversor deben de implementar el método convert para

definir como se convierten los ficheros a los formatos correctos. El método convert devuelve el fichero convertido o lanza una excepción si no puede convertirlo.

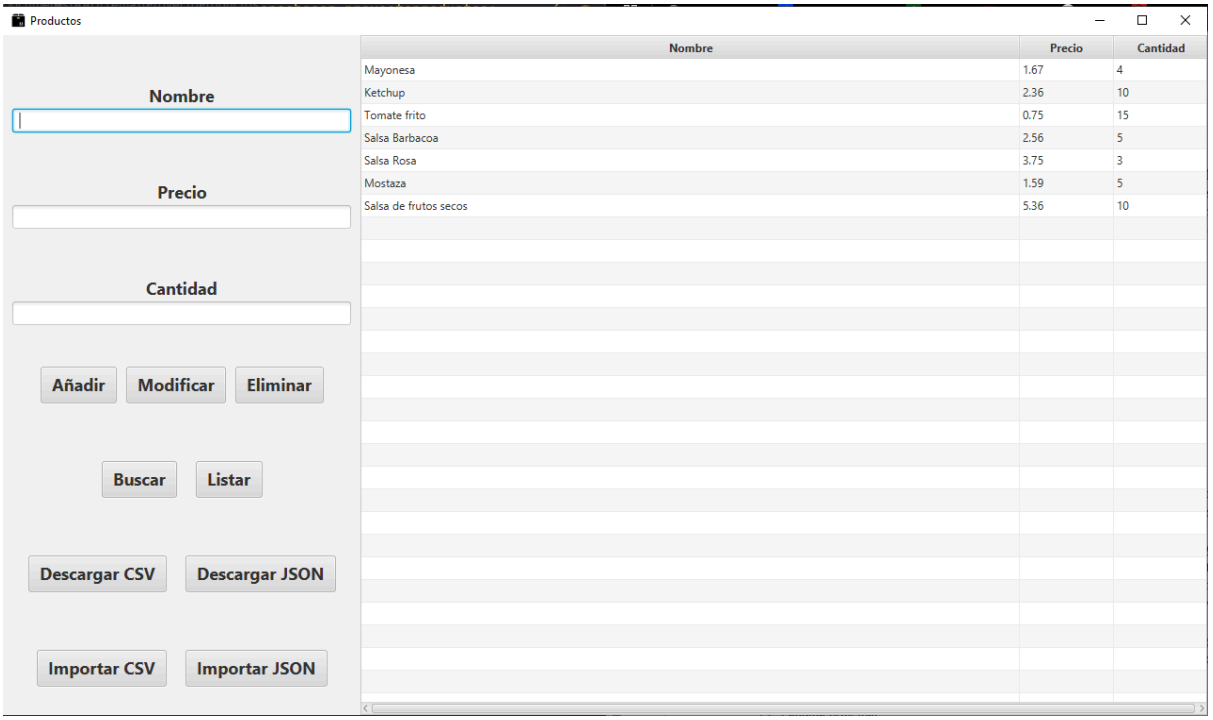
4.2- Relación entre clases

Las relaciones más relevantes entre las clases antes mencionadas son:

- Relación entre Producto y Productos: la clase Productos es una colección de Producto, con lo cual un objeto de Productos contiene una lista de Producto con varios productos(o vacía, pero contiene la lista)
- Relación entre Producto, Productos y ProductoDAO: la clase ProductoDAO se encarga de la relación entre la entidad Producto y el XML donde se guardan los registros de los productos existentes. ProductoDAO utiliza un objeto del tipo Producto o un objeto del tipo Productos dependiendo de si necesita la lista o el producto en concreto, y devuelve un Producto o un objeto de Productos dependiendo de lo que deba devolver.
- Relación entre Conversor y ConversorXML, ConversorCSV y Conversor JSON: es una relación de herencia entre una clase abstracta y otras tres clases que implementan su método abstracto.

5- Vista

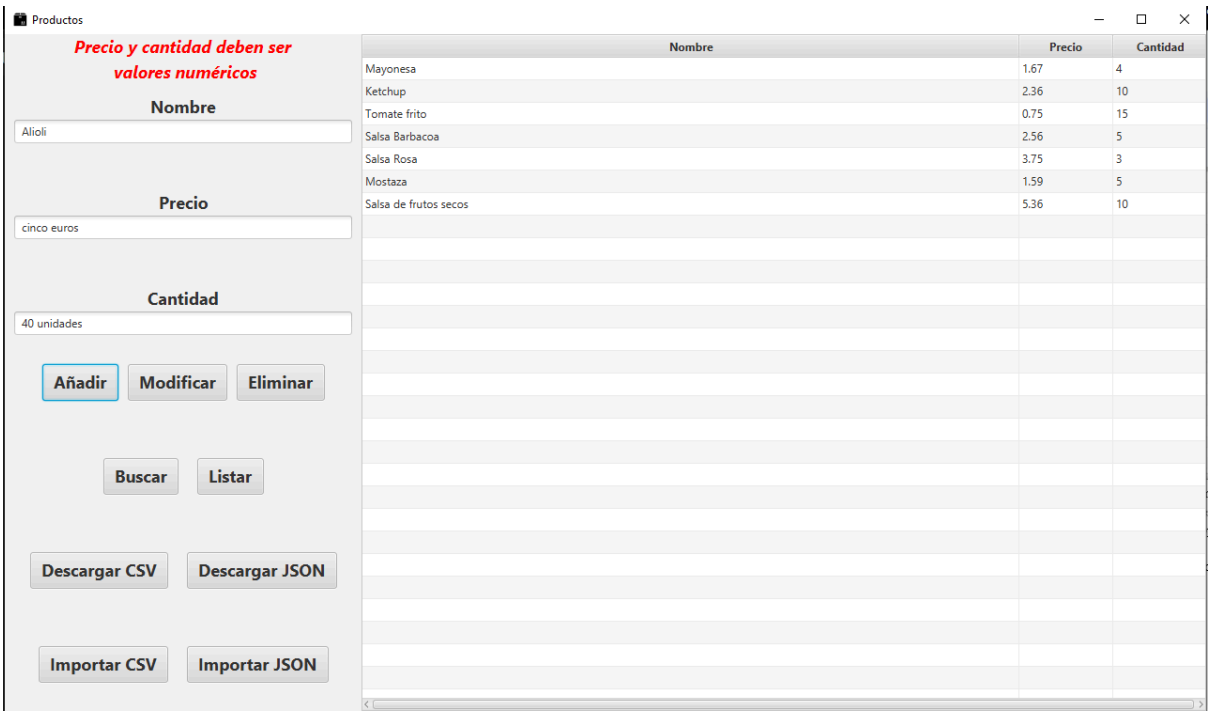
5.1- Pantalla principal: Explicación de la vista y de su propósito.



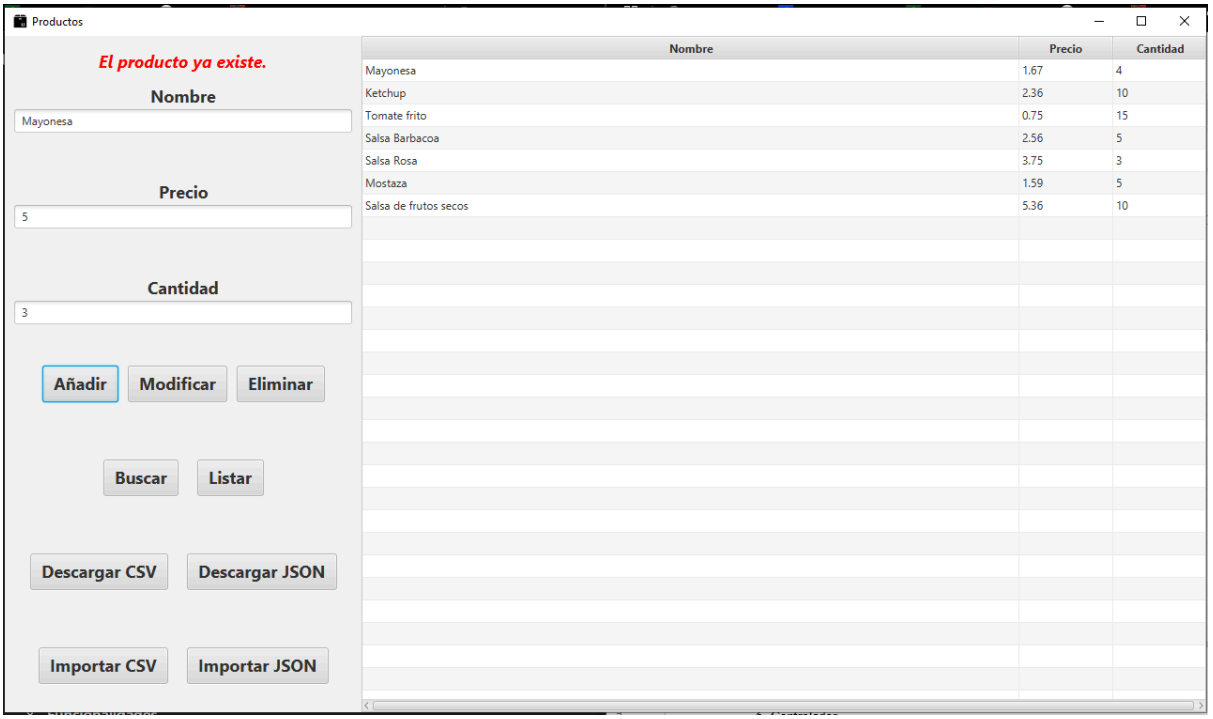
La pantalla principal es una interfaz sencilla e intuitiva que permite añadir, modificar, eliminar y buscar productos, así como listar la lista de productos almacenada en el XML, descargar una lista con los productos almacenados en formato CSV o JSON e importar una lista propia de productos en formato CSV o JSON.

Tiene una serie de restricciones, como:

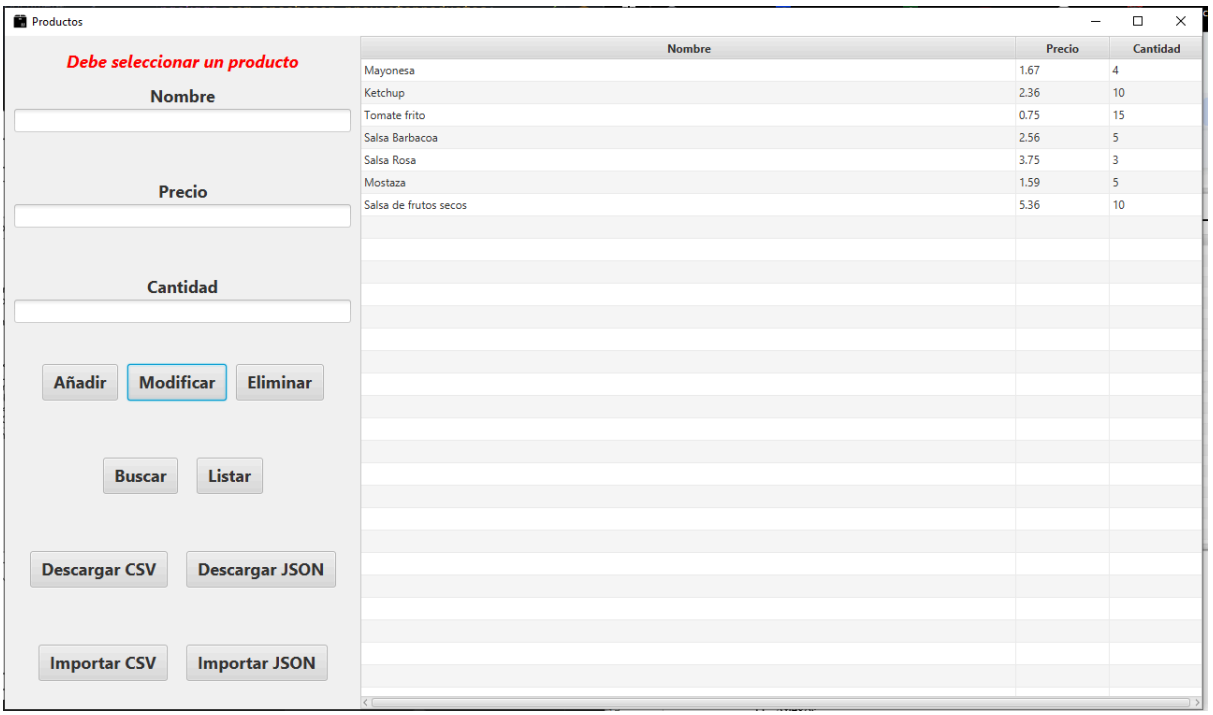
- No poder introducir cadenas de caracteres donde deben ir cifras



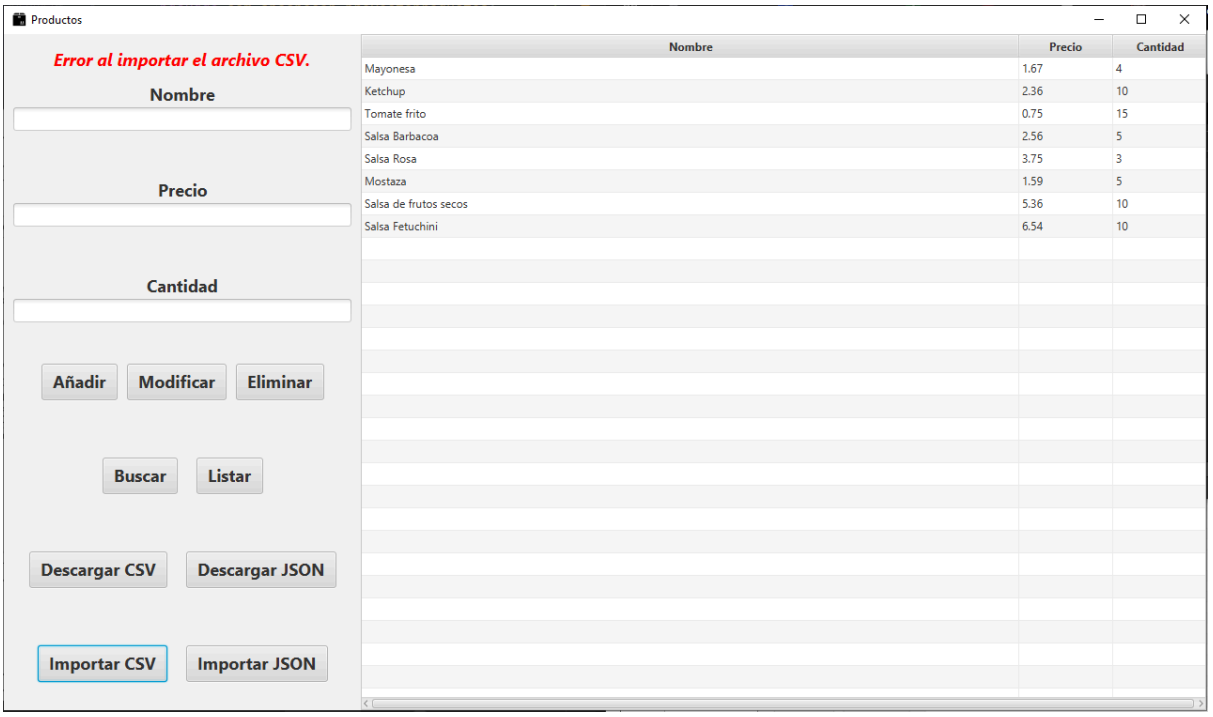
- No poder ingresar un elemento repetido



- No poder modificar o eliminar sin tener un elemento seleccionado



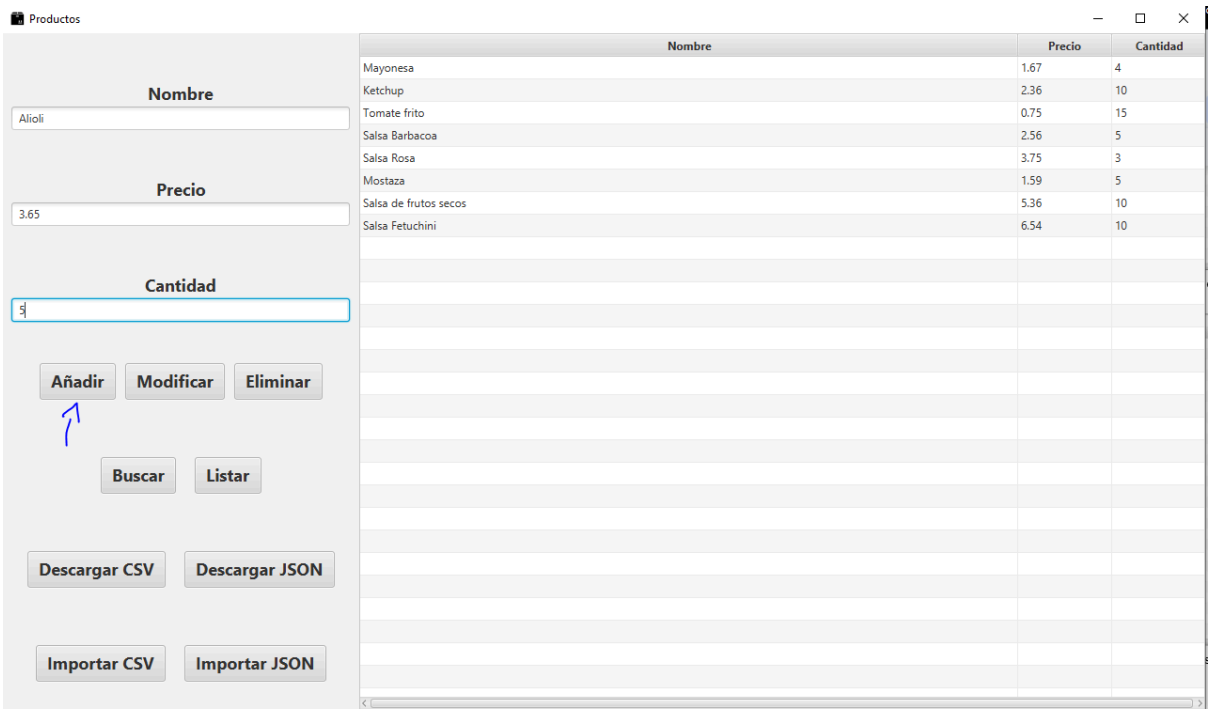
- No poder importar un archivo en un formato incorrecto(ej: que el csv o el json tengan fallos internos o que la extensión no coincida)

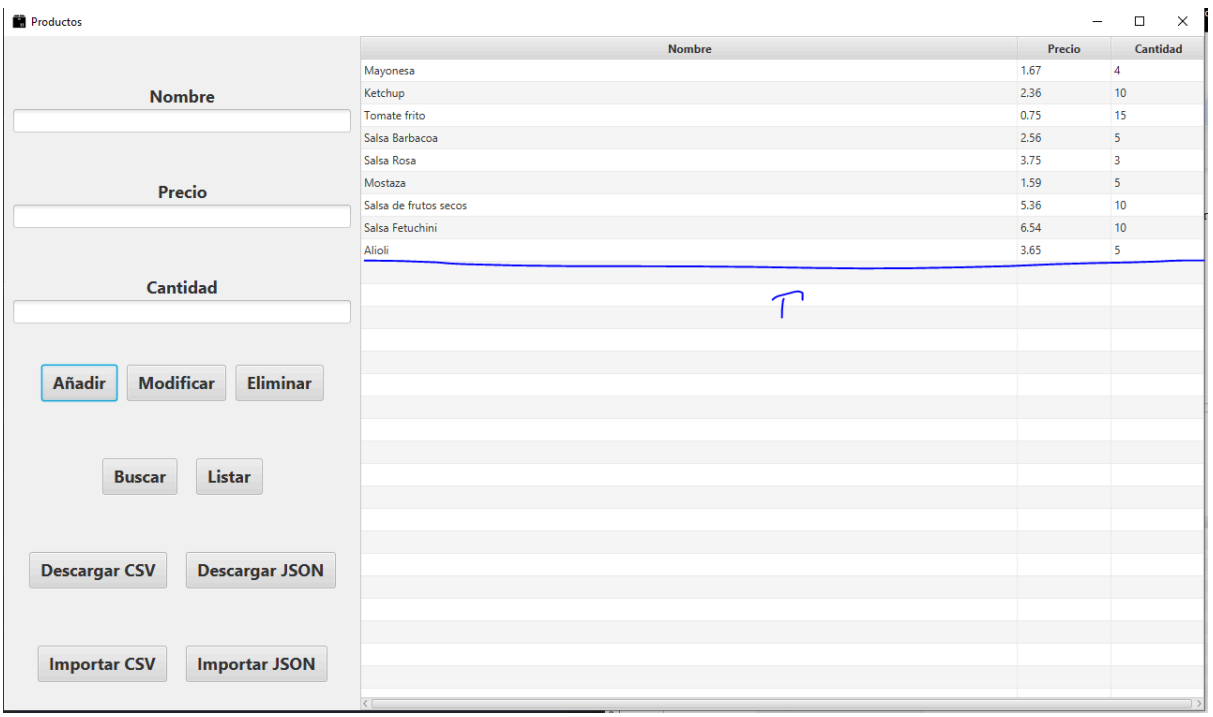


Podemos observar que la vista es muy simple, sin estilos muy elaborados. Todos los avisos salen en rojo en la parte superior del formulario.

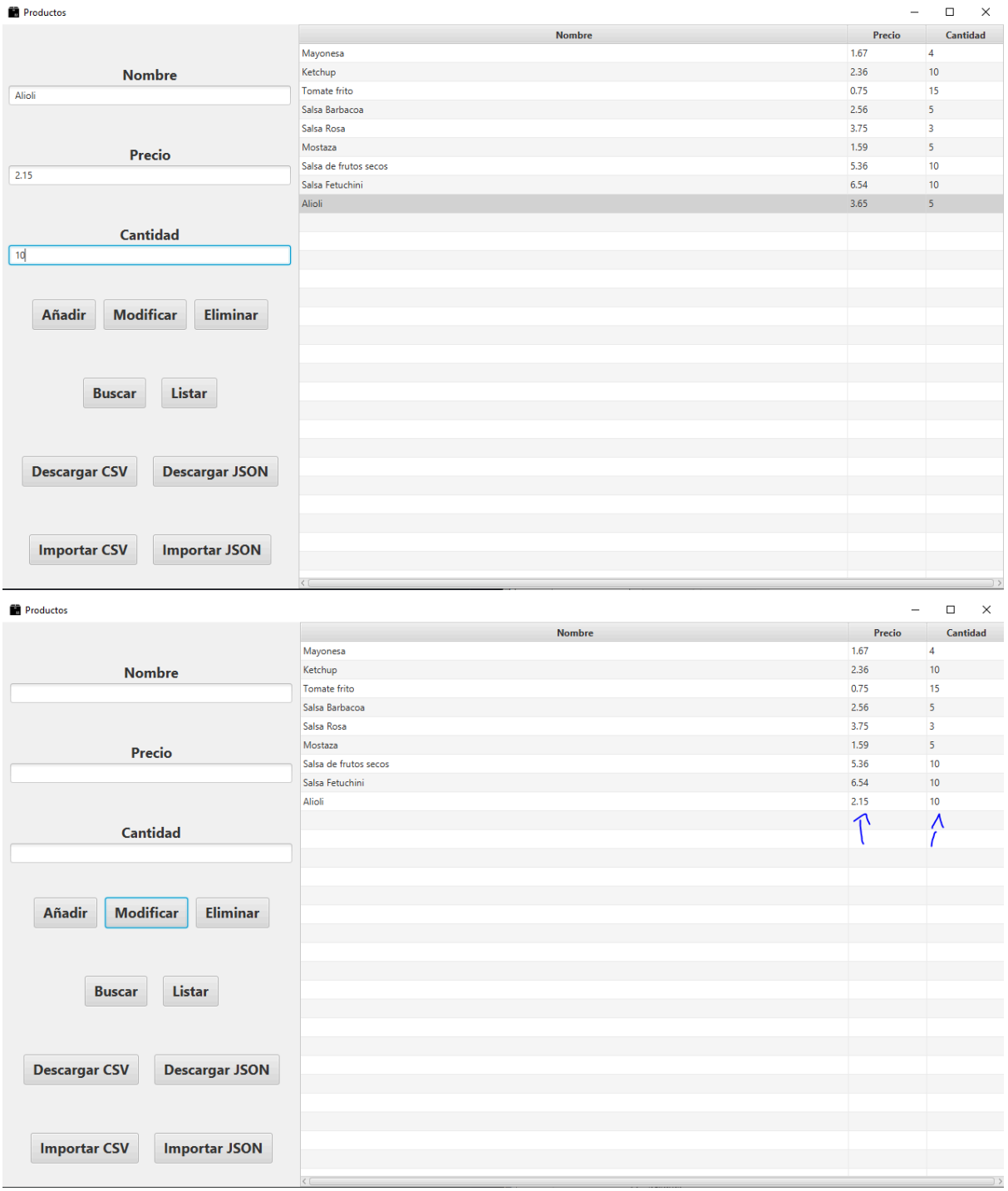
Veamos todas las funcionalidades:

- Para agregar un elemento deberemos completar sus atributos y pulsar sobre “Añadir”

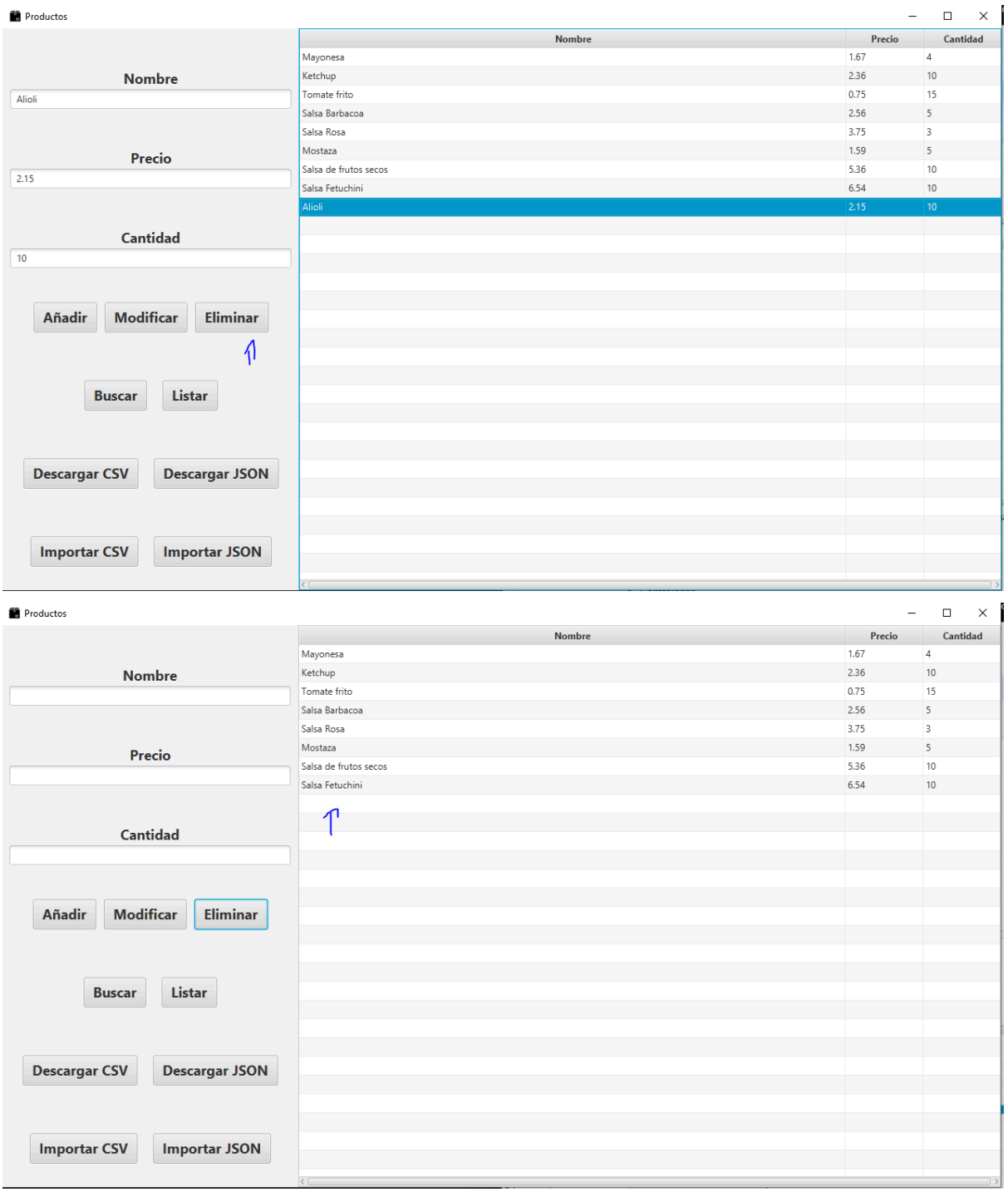




- Para modificar un elemento deberemos de pulsar sobre él en la tabla de la izquierda, y modificar los campos(que se nos autorrellenarán en la parte izquierda cuando pulsemos el producto de la tabla), y cuando lo tengamos pulsaremos sobre “Modificar”.

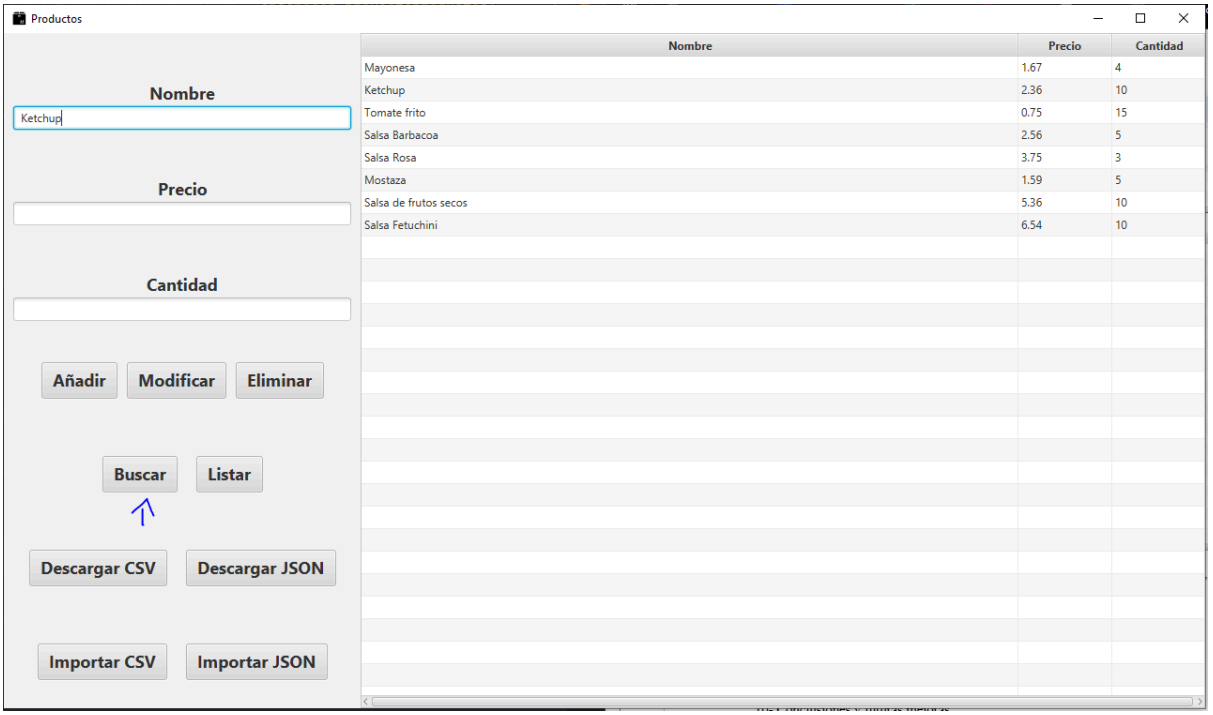


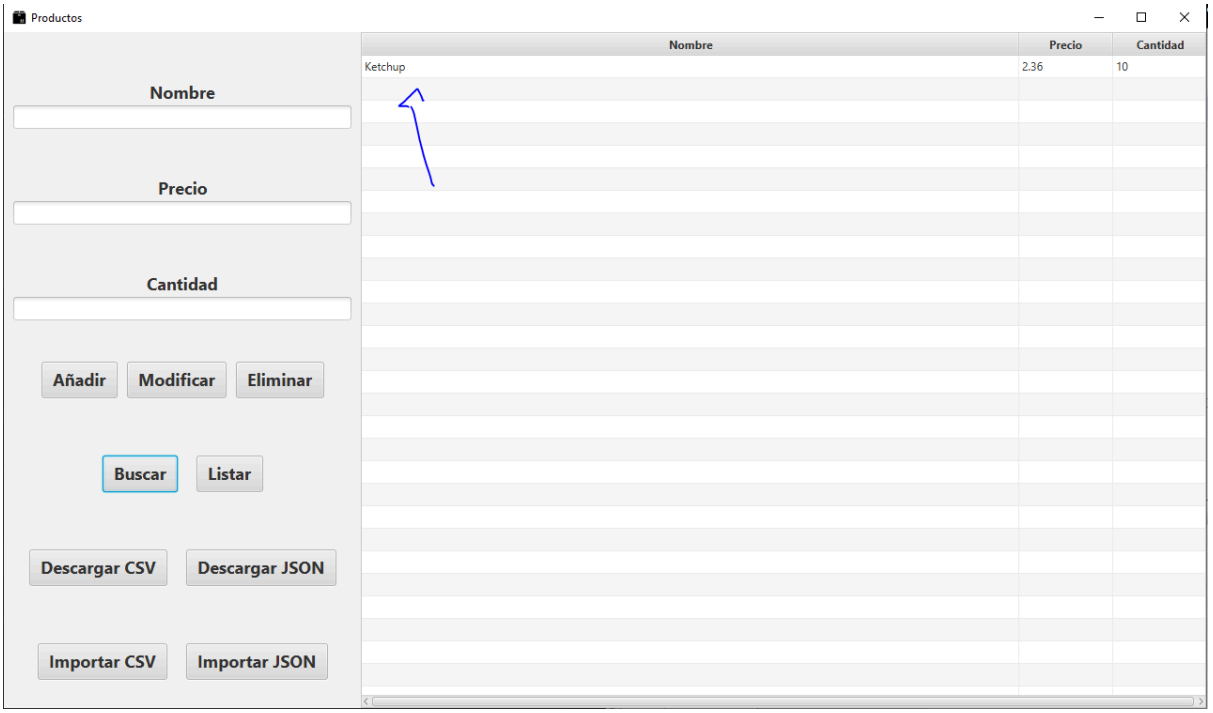
- Para eliminar deberemos pulsar sobre el elemento de la tabla a eliminar y pulsar sobre el botón “Eliminar”.

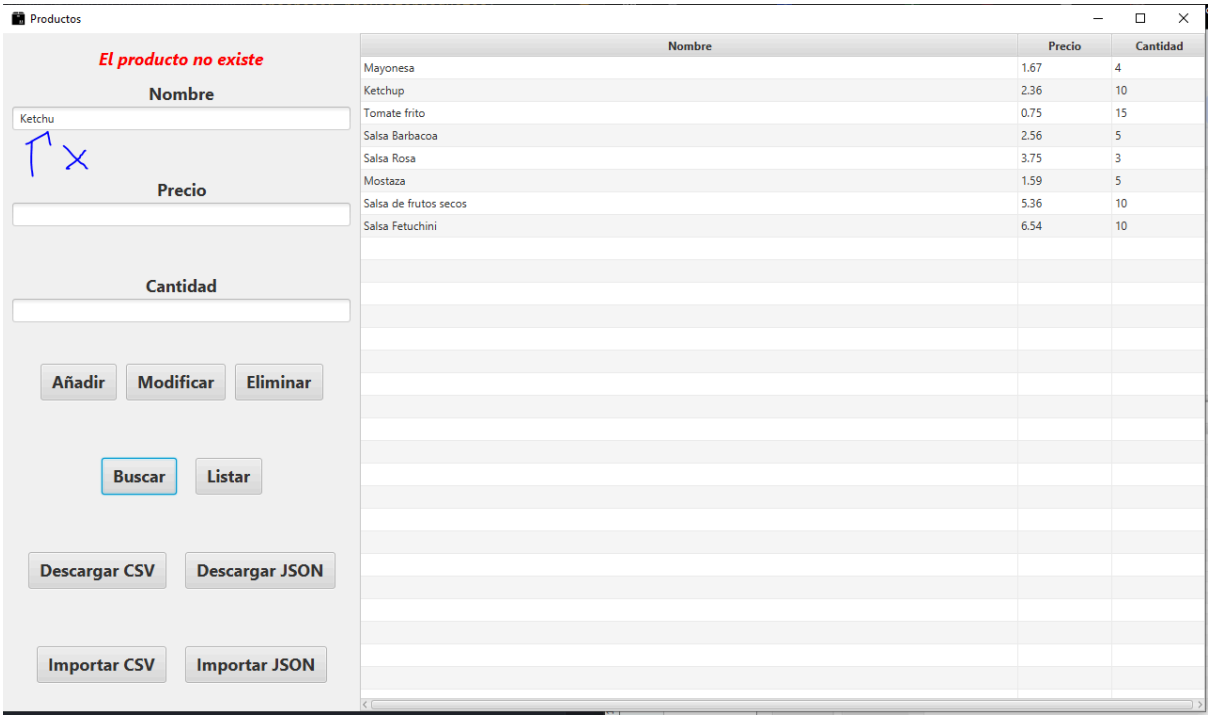


(Ya no está)

- Para buscar un elemento debemos introducir su nombre y darle a “Buscar”.


Productos
Nombre
Ketchup
Precio
Cantidad
Añadir
Modificar
Eliminar
Buscar
Listar
Descargar CSV
Descargar JSON
Importar CSV
Importar JSON
Nombre
Precio
Cantidad
Mayonesa 1.67 4
Ketchup 2.36 10
Tomate frito 0.75 15
Salsa Barbacoa 2.56 5
Salsa Rosa 3.75 3
Mostaza 1.59 5
Salsa de frutos secos 5.36 10
Salsa Fetuchini 6.54 10





(Si escribimos incorrectamente el nombre, no encontrará el producto)

- Para listar los productos (muy útil después de una búsqueda para volver a ver la lista de productos completa) deberemos pulsar sobre “Listar”.
- Para descargar la lista de productos en formato CSV deberemos pulsar sobre “Descargar CSV”, se nos abrirá una ventana para elegir el lugar donde queremos que se nos descargue, y cuando le demos a guardar lo tendremos. Para descargar la lista en formato JSON es el mismo procedimiento pero con el botón “Descargar JSON”.
- Para importar nuestros propios productos desde un formato CSV o JSON deberemos pulsar sobre el botón de “Importar CSV” o “Importar JSON”(según corresponda), elegir el fichero

en nuestro equipo y darle a abrir. Esto actualizará los productos a los nuevos que contenga el fichero seleccionado.

6- Controlador

6.1- Manejo de eventos

En nuestro proyecto únicamente tenemos un controlador, ProductoController. Éste define en su interior los métodos que acciona cada botón o interacción que genera un cambio. Éstos son:

- Los eventos de los botones, siendo aquellos vinculados a cada botón y que hacen que los botones, al ser pulsados, realicen la operación que indican en la palabra reflejada en ellos. Ejemplo: función asociada al botón "Eliminar":

```
@FXML  new *
private void eliminarProducto() {
    labelNotificacion.setText("");
    //tomamos el producto que se ha seleccionado
    Producto p = tablaProductos.getSelectionModel().getSelectedItem();
    //eliminamos p de la tabla y del xml
    this.productos.remove(p);
    this.tablaProductos.refresh();
    productoDAO.eliminarProducto(p);
    this.limpiarCampos();
    this.tablaProductos.getSelectionModel().clearSelection();
}
```

Esta función primero limpia la notificación superior (las letras en rojo que nos indican que algo no estamos haciendo bien o que algo ha sucedido), luego toma el producto que hemos seleccionado en la tabla y lo elimina tanto de la tabla(el visual) como del xml, para un borrado permanente. Luego refresca la tabla para mostrar la tabla sin el elemento eliminado, limpia los campos y deselecciona el elemento que se haya quedado seleccionado.

- El evento vinculado a la tabla, que toma el producto que hayamos seleccionado.

```
@FXML  new *
private void seleccionar(){
    labelNotificacion.setText("");
    //tomamos el producto que se ha seleccionado
    Producto p = tablaProductos.getSelectionModel().getSelectedItem();
    //siempre que no sea null, asignamos los valores en los inputs
    if(p != null){
        inputNombre.setText(p.getNombre());
        inputPrecio.setText(p.getPrecio().toString());
        inputCantidad.setText(p.getCantidad().toString());
    }
}
```

7- Funcionalidades

Como ya hemos explicado anteriormente y de manera bastante amplia, este proyecto tiene las funciones de manipular una lista de productos almacenada en un xml.
En este manejo, se incluye:

- Añadir productos.
- Eliminar productos.
- Modificar productos.
- Buscar productos por su nombre.
- Listar todos los productos.
- Importar una lista de productos en formato CSV o JSON.
- Exportar una lista de productos, que son los productos almacenados en el XML, en un archivo CSV o JSON.

7.1- Casos de uso y ejemplos

Este programa tiene un claro uso para gestionar una lista de productos de manera más eficiente en comparación a gestionarlo a mano, ya sea en una lista escrita o a ordenador. Tiene todas las funciones esenciales más un toque de compatibilidad entre distintos equipos(pudiendo importar y exportar tus productos entre equipos con ficheros CSV o JSON). Es muy útil si no se tienen herramientas más complejas y orientadas a objetivos más específicos. Es tan sencilla que es muy cómoda y amena de usar.

8- Problemas conocidos y soluciones

Se han enfrentado 2 problemas principales:

8.1- Uso de la interfaz gráfica JavaFX

Se decidió utilizar una interfaz gráfica y desde el principio se fijó la biblioteca JavaFX como la opción más sencilla y manejable. El gran problema fue el tenerla que redescubrir de nuevo, dado que llevaba bastante tiempo sin usarla. Este problema se subsanó revisando las documentaciones y consultando en diversos foros según era necesario.

8.2- Uso de ficheros JSON

Se quiso incluir un poco de dificultad al proyecto agregando el uso de ficheros JSON a parte de los ficheros CSV. Fue una decisión correcta dato que es un formato muy estandarizado y conocido. Se optó por la biblioteca Gson para el manejo de los ficheros JSON. El problema fue aprender a manipular estos ficheros, y se subsanó con la documentación online y diversos foros(como stackoverflow).

9- Conclusiones y futuras mejoras

El desarrollo de una aplicación de gestión de productos utilizando Java y JavaFX ha sido un proceso enriquecedor que ha permitido la creación de una herramienta funcional y eficiente. A continuación, se presentan las principales conclusiones del proyecto:

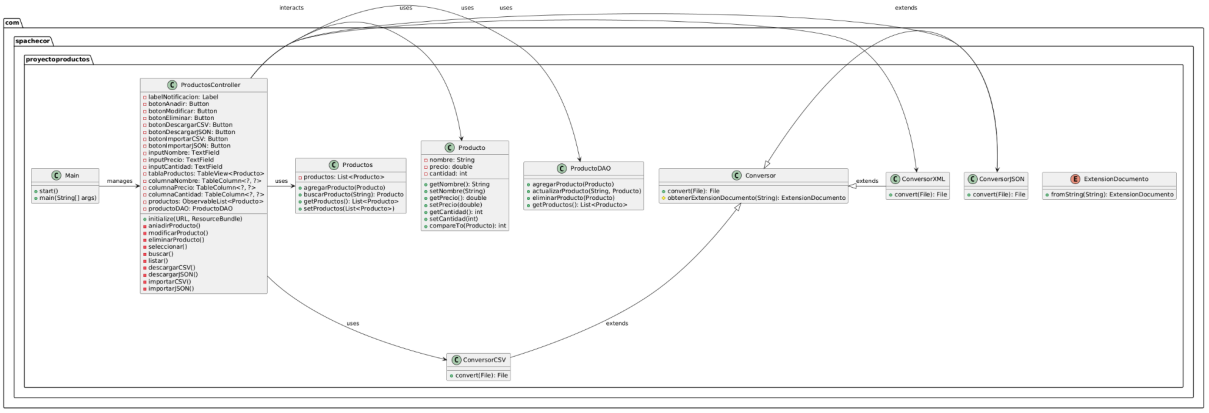
1. Implementación Efectiva del Patrón MVC:
    - La adopción del patrón Modelo-Vista-Controlador (MVC) ha facilitado la organización del código, permitiendo una separación clara entre la lógica de negocio, la interfaz de usuario y la interacción del usuario. Esto ha contribuido a un mantenimiento más sencillo y a una escalabilidad futura del sistema, ya que se pueden realizar modificaciones en un componente sin afectar a los demás.
2. Almacenamiento de Datos en XML:
    - La decisión de almacenar los datos de los productos en archivos XML, utilizando JAXB para su manipulación, ha demostrado ser efectiva. Este enfoque no solo ha simplificado la lectura y escritura de datos, sino que también ha permitido una representación clara y estructurada de la información, facilitando su gestión. La capacidad de editar el archivo XML manualmente en caso de ser necesario es un valor añadido.
3. Integración de Formatos CSV y JSON:
    - La funcionalidad para exportar e importar listas de productos en formatos CSV y JSON, utilizando la biblioteca Gson, ha ampliado la versatilidad de la aplicación. Esta característica permite a los usuarios trabajar con sus propios datos y facilita la integración de la aplicación en flujos de trabajo existentes. La exportación a CSV y JSON no solo es útil para el análisis de datos, sino que también proporciona opciones de respaldo y migración de datos.
4. Interfaz Gráfica Atractiva y Funcional:
    - La implementación de la interfaz gráfica con JavaFX ha permitido crear una experiencia de usuario intuitiva y atractiva. Los componentes gráficos, como botones, tablas y formularios, se han diseñado para ser fácilmente navegables, lo que mejora la accesibilidad y la usabilidad de la aplicación.
5. Facilidad de Uso y Accesibilidad:
    - La aplicación permite a los usuarios añadir, eliminar, modificar y buscar productos de manera eficiente. Estas funcionalidades están diseñadas para ser intuitivas, lo que reduce la curva de aprendizaje y permite a los usuarios gestionar sus productos con facilidad. La posibilidad de listar productos y realizar búsquedas específicas contribuye a una experiencia de usuario fluida.
6. Desarrollo Sostenible y Futuro:
    - Este proyecto sienta las bases para futuras mejoras y ampliaciones. Se pueden considerar funcionalidades adicionales, como la implementación de autenticación de

usuarios, la gestión de categorías de productos o la integración con servicios web. La arquitectura modular y el uso de bibliotecas bien establecidas facilitan la incorporación de nuevas características sin reestructurar significativamente el código existente.

En conclusión, la creación de esta aplicación de gestión de productos ha sido un éxito, logrando cumplir con los objetivos planteados al inicio del proyecto. Se ha logrado construir una herramienta robusta que no solo satisface las necesidades actuales de gestión de productos, sino que también está preparada para adaptarse a futuras exigencias del mercado.

10- Anexos

ANEXO I- Diagrama de clases.



ANEXO II- Diagrama de arquitectura.