

**Nombres:** Julián Guerrero, Isidora Osorio y Martina Sandoval

**Repositorio:** Repositorio [GitHub](#).

**SIA1.1 Realizar un análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto.**

Este proyecto consiste en la gestión del inventario de un supermercado y la compra de productos en un supermercado. En la aplicación podremos entrar como cliente y como empleado. Por lo que el objetivo de nuestro programa es que pueda ser utilizado por una persona o por algún trabajador para que utilice las distintas operaciones a los productos del supermercado.

Esta aplicación ayuda a tener una mejor gestión y un control más preciso de los productos del supermercado. Por esto los empleados tienen distintas funcionalidades para realizar estas tareas de una manera más efectiva.

Las funcionalidades de nuestra aplicación son las siguientes:

**Para el Cliente:**

- **Carrito de Compras:** Permite a los clientes añadir productos fácilmente, eliminarlos y proceder con la compra. Esto hace que la experiencia de compra sea más fluida.
- **Barra de Búsqueda:** Facilita la búsqueda de productos específicos, logrando que el cliente no tenga que revisar producto por producto.
- **Historial de Compras:** Permite a los clientes ver sus compras anteriores.

**Para el Empleado:**

- **Gestión de Inventario:** Funcionalidades para agregar, eliminar y modificar productos del inventario. Estas funciones ayudan a que la modificación del inventario sea más rápida y simple.
- **Barra de Búsqueda:** Similar a la del cliente, pero enfocada en la búsqueda de productos dentro del inventario.

Otras funcionalidades aparte que no son tan principales son la creación de reporte o la función que muestra los datos del cliente o empleado.

Los datos que se utilizarán son:

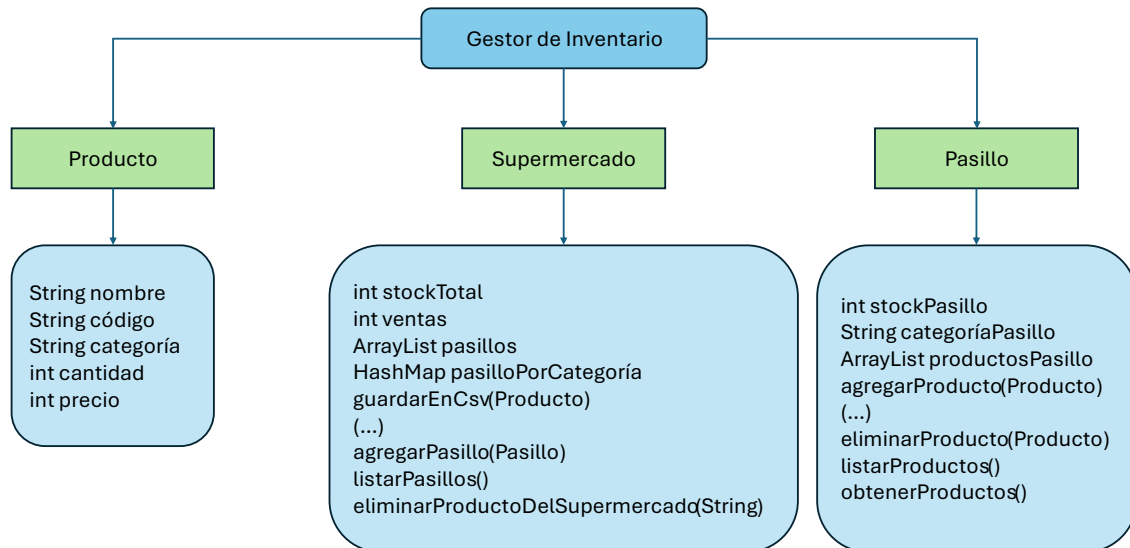
**Datos del Cliente:**

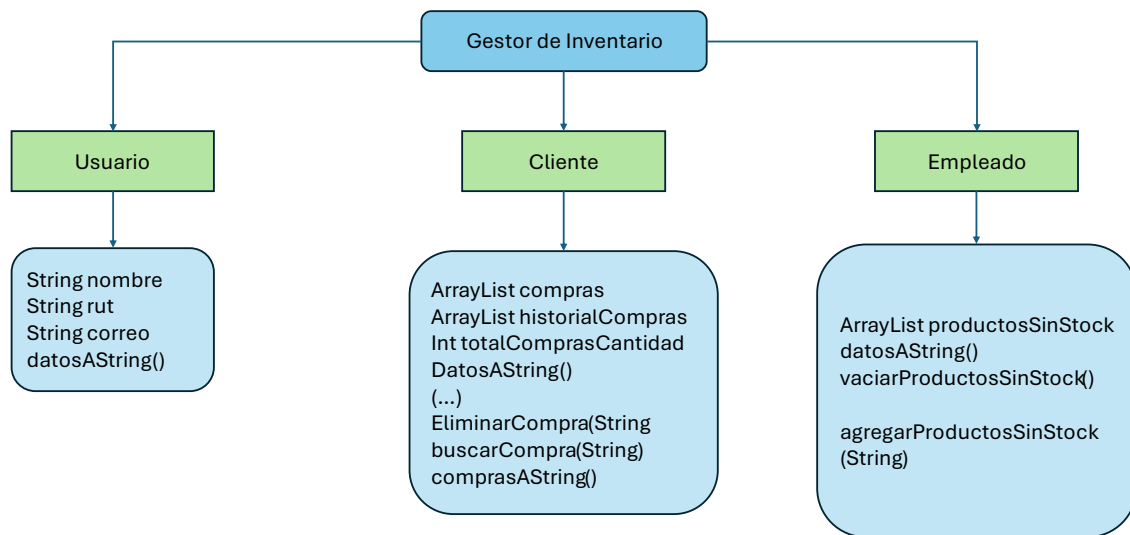
- **Identificación:** Nombre, RUT, correo electrónico.
- **Compras:** Registro de productos comprados, el total gastado o el historial de compra.

#### Datos del Empleado:

- **Identificación:** Nombre, RUT, correo electrónico.
- **Inventario de Productos:** Nombre del producto, código, categoría, stock disponible, precio. Los datos de los pasillos como el nombre o los productos que quedan en ese pasillo, etc. Estos datos son esenciales para una mejor gestión del inventario.

#### SIA1.2 Diseño conceptual de clases del Dominio y su código en Java





### **SIA1.3 Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter).**

Todos los atributos de las clases son privados y todas las clases que están el paquete clases (menos la clase Empleado, debido a que no tiene atributos) tienen sus setter y getter. Por lo que las clases que tienen setter y getter son:

Clase Cliente: líneas 116 a 120.

Clase Pasillo: líneas 170 a 197.

Clase Producto: líneas 45 a 93.

Clase Supermercado: líneas 249 a 264.

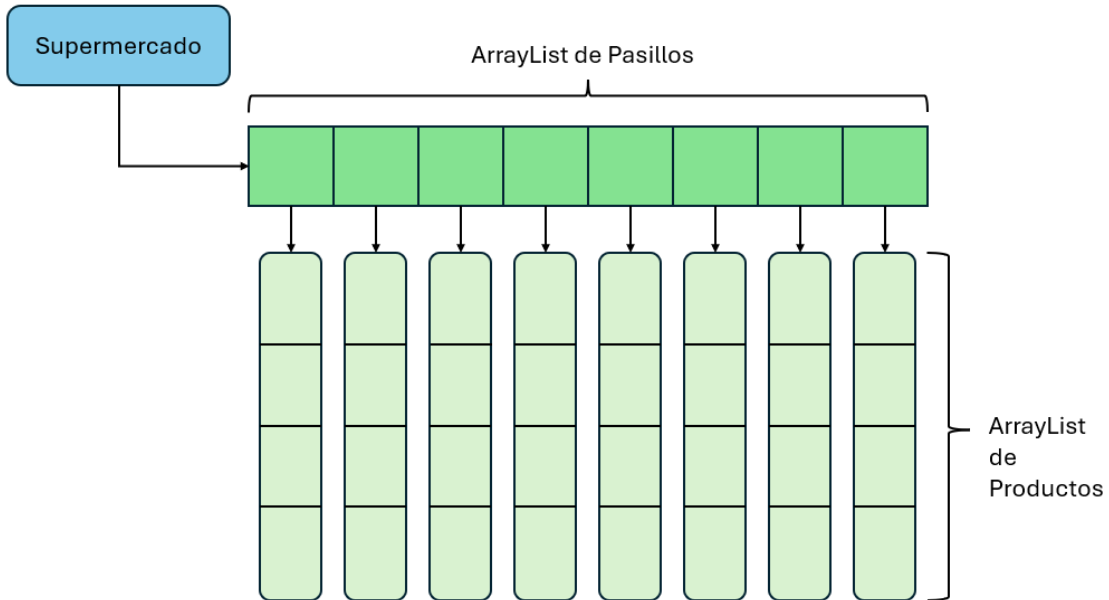
Clase Usuario: líneas 34 a 87.

### **SIA1.4 Se deben incluir datos iniciales dentro del código.**

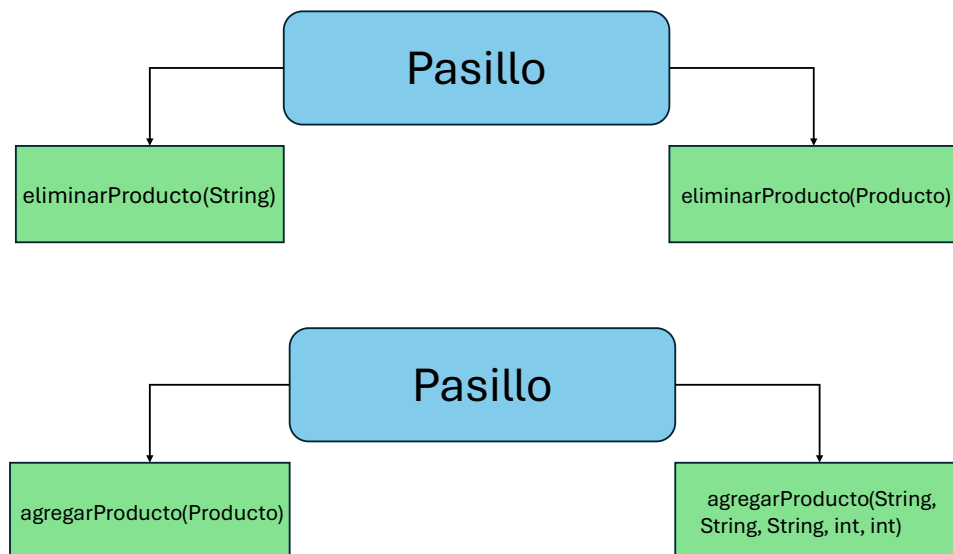
Nuestros datos iniciales se cargarán desde un archivo CSV. Este archivo se encuentra en la carpeta “Other Sources” en la parte de recursos.

La carga de datos se realiza en el método `leerCsv`, que está implementado en la clase `CsvFileReader`. Este método es responsable de abrir el archivo CSV, leer su contenido y almacenar los datos dentro de la aplicación.

**SIA1.5 Diseño conceptual y codificación de 2 colecciones de objetos, con la 2ª colección anidada como muestra la figura. Las colecciones pueden ser implementadas mediante arreglos o clases del Java Collections Framework(JCF).**



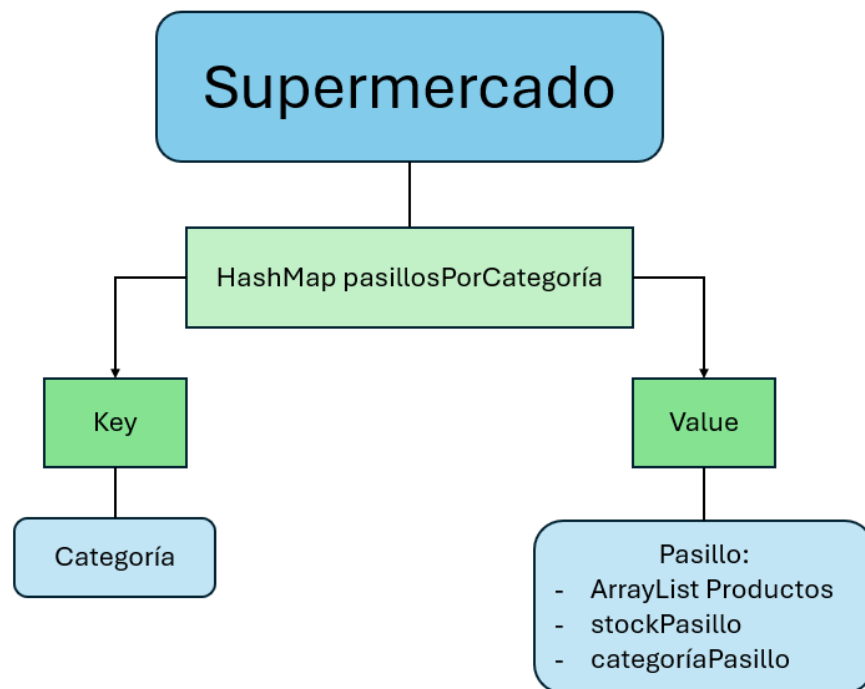
**SIA1.6 Diseño conceptual y codificación de 2 clases que utilicen sobrecarga de métodos (no de constructores)**



En la primera sobrecarga, la diferencia entre los 2 métodos de eliminación de un producto es que en el primero se requiere de un String para poder eliminar el producto, en cambio en el otro método piden el Producto para poder eliminarlo.

Y en la segunda sobrecarga, tenemos métodos que nos permiten agregar productos, el primer método que aparece nos pide el producto completo, pero en el segundo método nos piden los atributos del producto por separado.

### **SIA1.7 Diseño conceptual y codificación de al menos 1 clase mapa del Java Collections Framework**



Como se ve en el dibujo, la clase **Supermercado** tiene un **HashMap pasilloPorCategoría**, este **HashMap** tiene como clave la categoría de cada **Pasillo**, y como valor el **Pasillo** en sí. En la clase **Pasillo** se tiene un **ArrayList** con **Productos**, es decir, que cada **Pasillo** tiene sus distintos productos con sus respectivas características (nombre, código, precio, etc).

**SIA1.8 Se debe hacer un menú para el Sistema donde ofrezca las funcionalidades de: 1) Inserción Manual / agregar elemento y 2) Mostrar por pantalla listado de elementos. Esto para la 2ª colección de objetos (colección anidada) del SIA1.5**

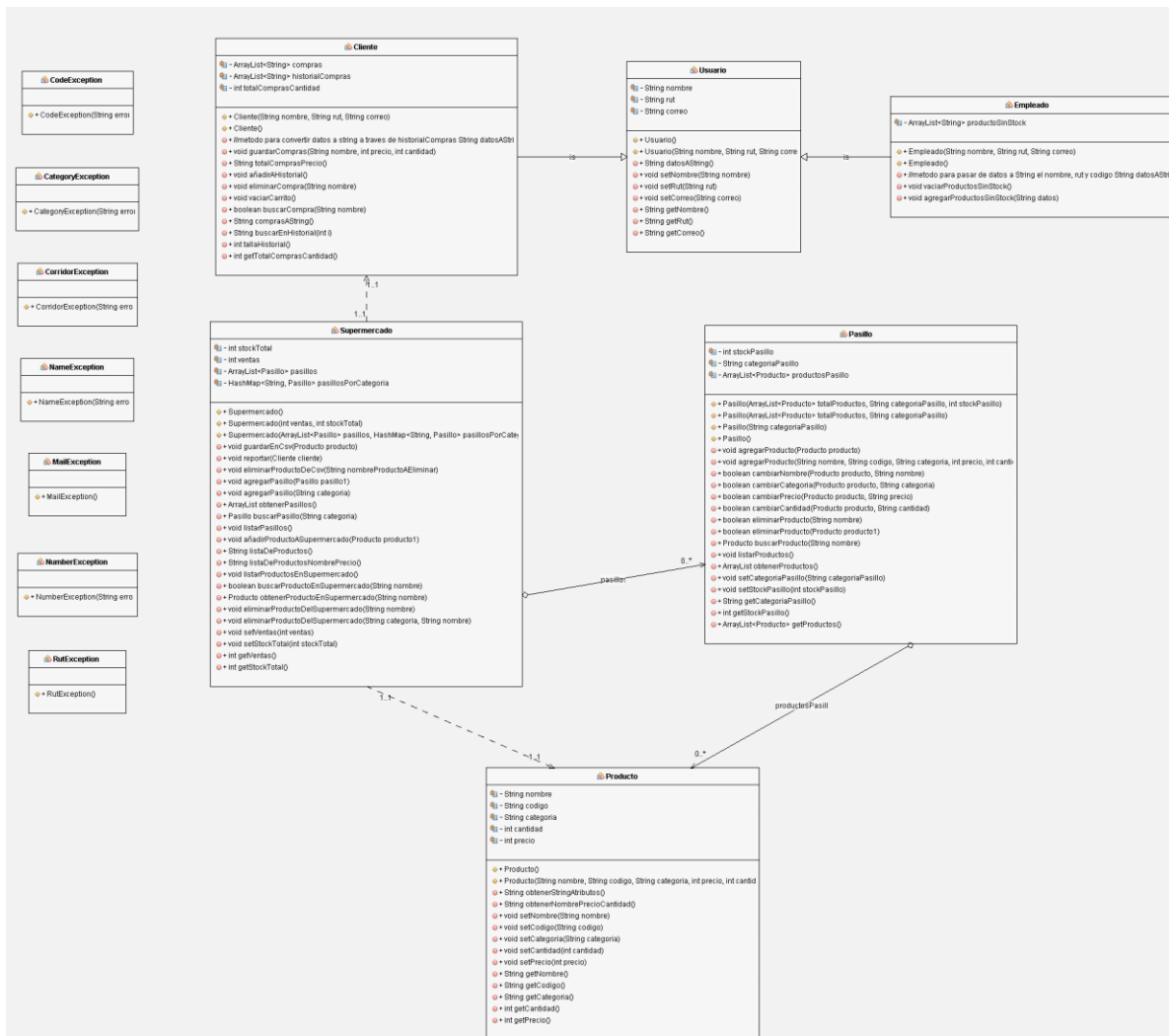
El menú del sistema está implementado en el apartado de Empleado, en este menú se muestran distintas opciones las cuales se pueden ingresar al apretarlas. En las opciones se muestran dos botones que ofrecen funcionalidades de inserción y de mostrar productos.

El menú funciona de la siguiente manera:

- **Agregar producto:** Cuando apretamos esta opción en la ventana se nos abre otra ventana que nos deja agregar datos de nuestro nuevo producto. Los datos que debemos ingresarle al nuevo ítem son, nombre, código, categoría (pasillo), precio y cantidad. En caso de que algún dato ingresado no sea válido saldrá el mensaje correspondiente. Y en caso de que la categoría que se haya puesto no exista, aparecerá un mensaje preguntando si quiere crear un nuevo pasillo.
- **Mostrar/ Eliminar/ Modificar:** Estas 3 funcionalidades se encuentran en la misma opción. Por lo que cuando entramos a esta opción lo primero que observamos es la lista de productos, si queremos eliminar algún producto solo debemos seleccionarlo y apretar en el botón eliminar. Si queremos modificar algún producto tenemos que apretar al producto y darle al botón de modificar. Dentro de este botón saldrá un desplegable que nos preguntará que campo queremos modificar del producto.

### **SIA2.1 Diseño de diagrama de clases UML.**

En el diagrama UML se presentan todas las clases del dominio sin considerar las ventanas ni el controlador. En cada clase se aprecian los atributos y los métodos de cada clase. Y en el UML se muestra la relación que puede tener una clase con la otra.



## SIA2.2 Persistencia de datos utilizando archivo de texto, CSV, Excel, o conexión con DBMS local (ej. MySQL). Utiliza sistema batch (carga datos al iniciar la aplicación y graba al salir)

Los datos son cargados desde un CSV llamado datosSupermercado en la clase CSVFileReader, en esta clase existe el método leerCsv el cual lee el archivo y asigna cada columna a la variable correspondiente. Para la escritura de datos tenemos el método guardarEnCsv que se encuentra en la clase Supermercado. Este método guarda todos los cambios que se hayan hecho en el csv durante la ejecución del programa.

## SIA2.3 La implementación de todas las interfaces gráficas (ventanas) para interactuar con el usuario, considerando componentes SWING.

La herramienta que utilizamos para desarrollar las ventanas de nuestra aplicación fue SWING. Hemos diseñado varias ventanas principales, así como ventanas específicas

para clientes y empleados. Estas ventanas incluyen componentes esenciales como botones, cuadros de texto y tablas, facilitando una navegación intuitiva y eficiente.

Menu Inicio

—

□

×

## Login

**Ingrese sus datos**

**Nombre:**

**Rut:**  (Con guión)  
Ej: 11111111-1

**Correo:**   
Ej: ejemplo@gmail.com

Salir

Aceptar

Menu Cliente

—

□

×

## Comprar producto

Buscar

Carrito

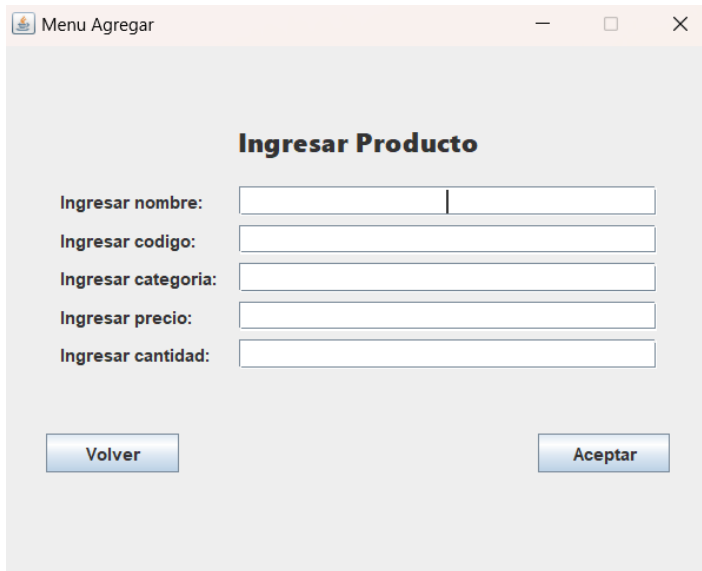
Usuario

| Nombre                      | Precio | Cantidad |
|-----------------------------|--------|----------|
| Pasta dental Ultra Blan...  | 1946   | 50       |
| Pasta dental triple 3 un. . | 1923   | 50       |
| Jabón líquido Dermo C...    | 516    | 50       |
| Jabón líquido original 70.  | 2069   | 50       |
| Pasta dental Xtra White..   | 1704   | 50       |
| Jabón líquido cremoso ...   | 619    | 50       |
| Jabón barra original 3 u... | 2106   | 50       |
| Jabón Dermo Care Hy...      | 516    | 50       |
| Jabón líquido avena doy.    | 619    | 50       |
| Toallas higiénicas Noso.    | 963    | 50       |

Volver

Añadir carrito





Menu Agregar

### Ingresar Producto

Ingresar nombre:

Ingresar codigo:

Ingresar categoria:

Ingresar precio:

Ingresar cantidad:

**SIA2.4 Se debe hacer un menú para el Sistema donde ofrezca las funcionalidades de: 1) Edición o modificación del elemento y 2) Eliminación del elemento. Esto para la 2ª colección de objetos (colección anidada) del SIA1.5**

Las opciones de modificación y eliminación de elementos de la segunda colección se encuentran en la parte de Empleado. Dentro de empleado se despliega un menú con cuatro opciones y en una de ellas aparece Listar/ Eliminar/ Modificar si entramos a esa opción veremos todos los productos del supermercado y si seleccionamos uno tenemos la opción de eliminarlo o de modificarlo.

**1.- Eliminar Producto:** Si deseamos eliminar algún producto es importante destacar que esta acción será permanente, lo que significa que el producto se eliminará del sistema y del archivo CSV asociado.

**2.- Modificar Producto:** Desde la lista, los empleados pueden seleccionar un producto para editar. Las opciones de modificación incluyen:

- Cambiar el nombre del producto.
- Actualizar la categoría.
- Ajustar el precio.
- Modificar la cantidad disponible.
- Editar todos los aspectos del producto simultáneamente.

**SIA2.5 Se deben incluir al menos 1 funcionalidad propia que sean de utilidad para el negocio (distintas de la inserción, edición, eliminación y reportes).**

**Específicamente: - Subconjunto filtrado por criterio: considera la selección de un**

**subconjunto de objetos basado en un criterio específico, involucrando 1 o más colecciones. Por ejemplo, selección de los alumnos con nota final entre 4,0 y 7,0 de entre todos los cursos; o seleccionar a todos los pasajeros que tengan asiento impar de entre todos los buses de la compañía.**

Para el negocio hemos implementado una funcionalidad de búsqueda por criterio, que permite filtrar los productos según las características que queremos. Esta funcionalidad es bastante útil tanto para los empleados como para los clientes debido a que ayuda a tener una mejor gestión de los productos.

Los criterios de búsqueda son: nombre, categoría, precio, código y stock.

Para poder buscar por los criterios solo debemos poner en la barra de búsqueda lo que queramos buscar. Por ejemplo, si quiero buscar productos de la categoría Lácteos.

The screenshot shows a window titled 'Listar Productos'. Inside, there is a search bar with the text 'lacteos' and a 'Buscar' button. Below the search bar is a table with the following data:

| Codigo        | Nombre          | Categoria | Cantidad | Precio |
|---------------|-----------------|-----------|----------|--------|
| 4330886858... | Leche enter...  | lacteos   | 50       | 283    |
| 3818548610... | Mantequilla ... | lacteos   | 50       | 1456   |
| 9596684879... | Leche en Po...  | lacteos   | 50       | 4049   |
| 3682021066... | Leche semi...   | lacteos   | 50       | 283    |
| 2304226281... | Yoghurt bati... | lacteos   | 50       | 76     |
| 4955201472... | Leche descr...  | lacteos   | 50       | 283    |
| 6939488929... | Mantequilla ... | lacteos   | 50       | 819    |
| 8993743275... | Leche enter...  | lacteos   | 50       | 313    |
| 3766010038... | Huevos extr...  | lacteos   | 50       | 1389   |
| 6045252749... | Huevos gra...   | lacteos   | 50       | 826    |
| 2855002703... | Mantequilla ... | lacteos   | 50       | 753    |
| 8816188884... | Jalea framb...  | lacteos   | 50       | 81     |

At the bottom of the window, there are three buttons: 'Volver', 'Eliminar', and 'Modificar'.

O si quiero buscar productos con un cierto precio o código.

**Lista Productos**

Buscar: 1456

| Codigo       | Nombre        | Categoria      | Cantidad | Precio |
|--------------|---------------|----------------|----------|--------|
| 381854861082 | Mantequill... | lacteos        | 50       | 1456   |
| 438166145631 | Postre del... | postres        | 50       | 165    |
| 137497922166 | Mantequill... | mantequilla... | 50       | 1456   |
| 696231456671 | Spaghetti ... | pastas y s...  | 50       | 572    |

Volver Eliminar Modificar

La única desventaja de este tipo de búsqueda es que encuentra todos los productos que tenga algo en común con lo que se buscó. Es decir, si yo busco “1456” me encuentra los productos con ese precio, pero también me muestra los productos que tienen en su código ese número. (Esto solo ocurre cuando se busca en el apartado de empleado)

La búsqueda para los clientes se presenta de forma simplificada, permitiendo a los usuarios encontrar rápidamente los productos que desean sin complicaciones adicionales.

**Comprar producto**

Buscar: has

Carrito Usuario

| Nombre                      | Precio | Cantidad |
|-----------------------------|--------|----------|
| Palta hass malla 1 kg       | 1330   | 50       |
| Palta hass extra malla 1 .. | 1563   | 50       |

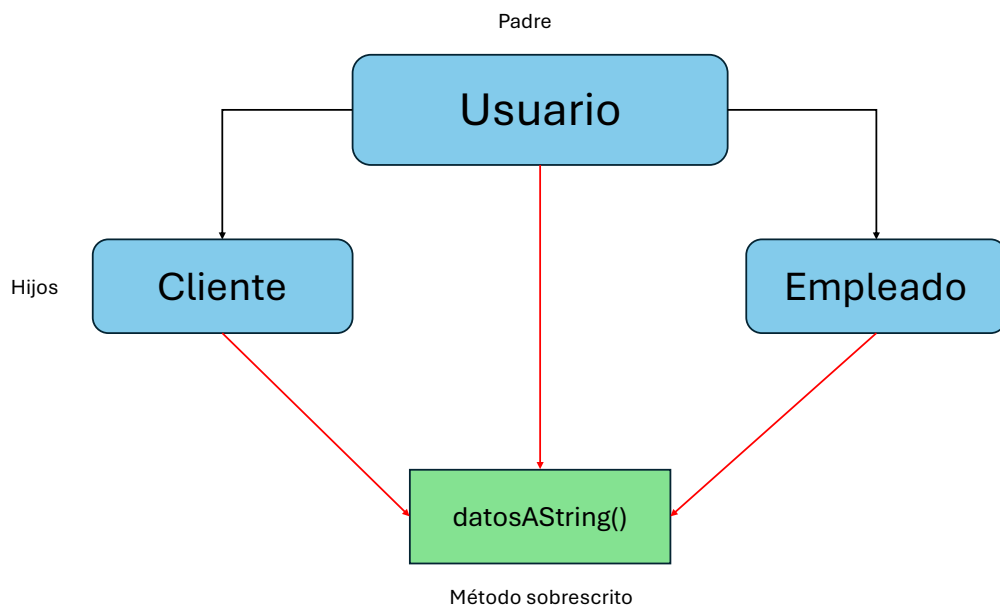
Volver Añadir carrito

**SIA2.6 El código fuente debe estar bien modularizado de acuerdo a lo descrito en el informe además de seguir las buenas prácticas de documentación interna y legibilidad.**

Las practicas empleadas en el código son las de mantener un buen orden y documentar todos los métodos del programa, también tratar de no dejar líneas tan

largas para la correcta visualización del código, menos en la clase controlador, ya que no quedaba legible si se recortaban las líneas. Otras de las practicas realizadas a la hora de codificar es que dejar todo ordenado en el siguiente orden: Declaración de clase, constructores, métodos, métodos setter/getter... todo con la finalidad de que cada clase quede estructurada en ese orden para una mejor comprensión del código. Otra de las practicas aplicadas es el nombramiento de variables significativas para mejor comprensión del código. Además, se ocupa líneas vacías para poder diferenciar mejor cada método, constructor, etc. Dentro del programa. El formato de documentación para el código es: Al principio de cada clase se documenta lo que esta realiza y se extiende de otra clase, etc... También en medio del código se va documentando arriba de cada declaración lo que hace los distintos métodos y su función, por otra parte, también se señala los que son constructores y métodos setter/getter

### SIA2.7 Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos.



#### Clase Padre: Usuario

La clase **Usuario** es la clase base que representa a un usuario en el sistema. Sus atributos principales son:

- **nombre:** String que almacena el nombre del usuario.
- **rut:** String que almacena el rut del usuario.
- **correo:** String que representa la dirección de correo electrónico del usuario.

El método principal de esta clase es **datosAString()**, que devuelve una representación en forma de cadena de los datos del usuario. Este método será sobrescrito en las clases hijas.

### **Clase Hijo: Cliente**

La clase Cliente es uno de los hijos de Usuario. Sus atributos son:

- **compras:** ArrayList que almacena las compras del cliente.
- **historialCompras:** ArrayList que almacena todas las compras que ha hecho el cliente.
- **totalComprasCantidad:** Entero el total de las compras realizadas.

Los métodos que tiene esta clase son: **datosAString** (método sobrescrito), **guardarCompras**, **totalComprasPrecio**, **añadirAHistorial**, **eliminarCompra**, **vaciCarrito**, **buscarCompra**, **comprasAString**.

### **Clase Hijo: Empleado**

La clase Empleado es el otro hijo de la clase Usuario. Su atributo es:

- **productoSinStock:** ArrayList que almacena los productos que no tienen stock.

Los métodos que tiene esta clase son: **datosAString** (método sobrescrito), **vaciProductosSinStock** y **agregarProductosSinStock**.

## **SIA2.8 Implementar el manejo de excepciones capturando errores potenciales específicos mediante Try-catch.**

La sección donde se implementó el Try-catch fue en nuestro controlador. Esto nos permite gestionar de manera efectiva los errores que puedan surgir al procesar las entradas del usuario. Por lo que el objetivo de nuestro Try-catch es mostrar un mensaje cada vez que la entrada no es válida.

## **SIA2.9 Crear 2 clases que extiendan de una Excepción y que se utilicen en el programa.**

Algunas de las clases que extienden de Excepción son **RutException** y **MailException**, estas excepciones se utilizan al principio del programa debido a que el usuario debe de ingresar su rut y su correo. Por lo que estas excepciones se encargan de que los datos sean válidos.

En el caso del rut, se valida el largo del rut ingresado y se verifica que se haya colocado un guion “-” antes del dígito verificador.

Para el correo lo que se válida más que nada es que tenga la terminación “@gmail.com”, debido a que ese contario como un correo válido.

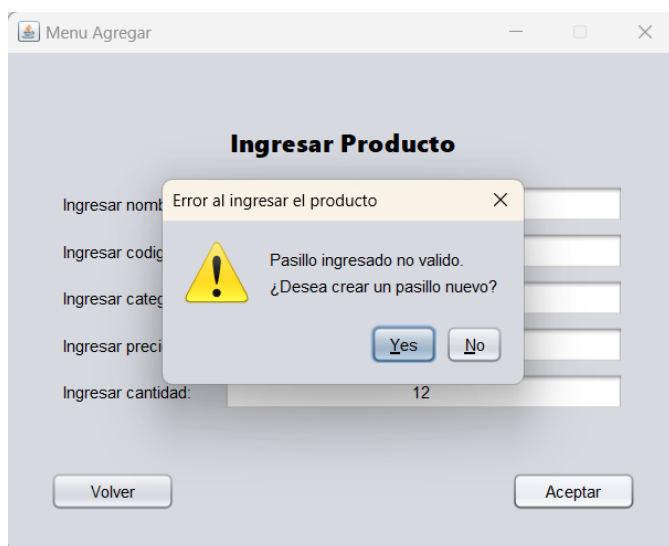
### **SIA2.10 Se debe generar un reporte en archivo txt que considere mostrar datos de la colección de objetos (ej: csv).**

El reporte se almacena dentro de la carpeta de recursos, donde lo que se reporta son las ventas hechas por los clientes, todo esto se implementa después que el cliente dentro de la interfaz del carrito le da al botón de comprar carrito. Dentro de este se almacena el nombre, cantidad y precio de los distintos productos que compra el cliente. Todo esto a través de un método específico para ir guardando cada producto dentro del archivo .txt, el cual se llama reportar(), el cual recibe la lista de los productos comprados por el cliente y los escribe dentro de este archivo. Para su visualización será necesario ingresar a la carpeta de recursos, donde estará junto al .csv de los productos del supermercado.

### **Extras**

### **SIA3.1 (opcional) Implementar las funcionalidades de agregar, eliminar o modificar en elementos de la 1ra colección o nivel.**

Para la primera colección se implementó la funcionalidad de agregar. Cuando se ingresa un producto y la categoría no es igual a ninguna que se encuentra en el supermercado, se muestra una ventana la cual le da la opción al usuario de crear un nuevo pasillo con esa categoría.



Una vez el usuario presiona si, se creara un nuevo pasillo en el supermercado que tiene como “nombre” la categoría ingresada, este luego se ingresa al mapa que contiene supermercado se revisa que en realidad no exista, de este ser el caso, se

agrega tanto al mapa como a una lista de pasillos, luego se ingresa el nuevo producto a este nuevo pasillo.

### SIA3.2 (opcional) Implementar la funcionalidad de buscar elemento en 1 o más niveles.

La funcionalidad de buscar elementos en 1 o más niveles está implementada en nuestra aplicación en la parte de empleado. Esta funcionalidad se encuentra en la barra de búsqueda, debido a que, si ingresamos por ejemplo una categoría y apretamos buscar, mostrara todos los productos en esa categoría, pero también si ingresamos un número o algunos caracteres, mostrarán los elementos que tengan similitudes con lo buscado.

The screenshot shows a window titled 'Listar Productos' with a search bar containing 'fas' and a 'Buscar' button. Below the search bar is a table with 5 columns: 'Codigo', 'Nombre', 'Categoria', 'Cantidad', and 'Precio'. The table contains 4 rows of data. At the bottom of the window are three buttons: 'Volver', 'Eliminar', and 'Modificar'.

| Codigo       | Nombre         | Categoria       | Cantidad | Precio |
|--------------|----------------|-----------------|----------|--------|
| 820191030752 | Corazones d... | conservas       | 50       | 999    |
| 182822687518 | Corazones d... | conservas       | 50       | 999    |
| 992755185712 | Corazones d... | conservas       | 50       | 999    |
| 557082484294 | Crema de Al... | instantaneos .. | 50       | 249    |

The screenshot shows a window titled 'Listar Productos' with a search bar containing 'lacteos' and a 'Buscar' button. Below the search bar is a table with 5 columns: 'Codigo', 'Nombre', 'Categoria', 'Cantidad', and 'Precio'. The table contains 15 rows of data. At the bottom of the window are three buttons: 'Volver', 'Eliminar', and 'Modificar'.

| Codigo        | Nombre          | Categoria | Cantidad | Precio |
|---------------|-----------------|-----------|----------|--------|
| 4330886858... | Leche enter...  | lacteos   | 50       | 283    |
| 3818548610... | Mantequilla ... | lacteos   | 50       | 1456   |
| 9596684879... | Leche en Po...  | lacteos   | 50       | 4049   |
| 3682021066... | Leche semi...   | lacteos   | 50       | 283    |
| 2304226281... | Yoghurt bati... | lacteos   | 50       | 76     |
| 4955201472... | Leche descr...  | lacteos   | 50       | 283    |
| 6939488929... | Mantequilla ... | lacteos   | 50       | 819    |
| 8993743275... | Leche enter...  | lacteos   | 50       | 313    |
| 3766010038... | Huevos extr...  | lacteos   | 50       | 1389   |
| 6045252749... | Huevos gra...   | lacteos   | 50       | 826    |
| 2855002703... | Mantequilla ... | lacteos   | 50       | 753    |
| 8816188884... | Jalea framb...  | lacteos   | 50       | 81     |

En el código, para la búsqueda de productos o pasillos en general se utilizaron diversos métodos, pero para la búsqueda en el Empleado se realizó de una manera un poco distinta debido a que se utilizó un filtro, esto debido a que en la ventana se tiene

una tabla, así que lo que se ingrese a la barra de búsqueda sirve como filtro para la tabla, y así logra mostrar solo los elementos que tengan las similitudes con lo buscado.

### **SIA3.7 (opcional) Considerar la implementación del patrón Modelo-Vista-Controlador (MVC) en la arquitectura del sistema.**

El patrón MVC (Modelo-Vista-Controlador) es un patrón de arquitectura de software utilizado principalmente para organizar y estructurar aplicaciones que separan la lógica de negocio, la lógica de presentación y el manejo de las interacciones del usuario. En el contexto del proyecto de gestión de productos en un supermercado (con las clases Supermercado, Pasillo y Producto), implementar el patrón MVC ayudaría a:

- Separar la lógica de negocio de la lógica de presentación y la interacción del usuario. Esto hace que el código sea más modular, fácil de mantener y escalar.
- Facilitar el cambio de la interfaz de usuario: Si, por ejemplo, se decide cambiar la forma en que se presenta el reporte de ventas, los cambios se realizarían solo en la Vista, sin modificar la lógica central de la aplicación.
- Gestionar las interacciones del usuario de manera más clara: El Controlador maneja la interacción con el usuario, como vender productos o generar reportes. Esto centraliza el manejo de eventos y hace que la lógica de la aplicación sea más clara.

Un ejemplo de las clases vista en el programa son VentanaAgregar, VentanaCarrito, VentanaCliente, entre otras, que representan las ventanas gráficas e interfaces del proyecto. Estas clases gestionan la interacción con el usuario. Por otro lado, las clases modelo incluyen Supermercado, CsvfileReader, Pasillo, y otras que se encargan de la lógica del negocio, procesamiento de datos y la interacción con bases de datos o archivos. Estas clases encapsulan la funcionalidad central del sistema. Finalmente, la clase Controlador actúa como intermediaria entre las vistas y los modelos, gestionando las acciones que se deben llevar a cabo según la interacción del usuario. Es responsable de coordinar la lógica de negocio y actualizar las vistas adecuadamente.