



Escuela Politécnica Nacional
Facultad de Ingeniería en Sistemas

Programación II

Proyecto IIB

Docente: Ing. Maldonado Daniel

Integrantes:

Collaguazo Shirley

Chuquer Jorman

Pazmiño Loreley

GR3-SW

9 de septiembre del 2022

PROYECTO SEGUNDO BIMESTRE

1. Objetivos

1.1. Objetivo General

Implementar un código que permita que el usuario pueda ingresar sus datos y seguidamente pueda hacer compras, mediante el uso de interfaz GUI, sockets y conceptos aprendidos en horas de clase para emitir una factura de supermercado que presente cálculos correctos de los valores ingresados y los datos del usuario

1.2. Objetivos específicos

- Implementar una conexión entre dos pc (cliente – servidor), mediante sockets en el código, para enviar y receptar un archivo txt.
- Implementar interfaces gráficas, mediante Gui.form para generar interfaces de usuario y factura en la que permita ingresar datos y realizar compras.
- Realizar la validación de datos, mediante el uso de excepciones para evitar que el programe salte errores.

2. Introducción

La programación orientada a objetos es algo que se ha estado haciendo énfasis en todo el semestre de programación II, se ha planteado un problema para que los estudiantes lo solucionen con los conocimientos adquiridos a lo largo de todo el semestre, e implementar de manera correcta una solución, con el uso de GUI, con métodos conocidos, uso de escritura de archivos, objetos serializables, y sockets de manera general.

3. Marco teórico

La API de Java para desarrollo de GUI

La interfaz de usuario es la parte del programa que permite al usuario interactuar con él. La API de Java proporciona una biblioteca de clases para el desarrollo de Interfaces gráficas de usuario (en realidad son dos) [1]. La biblioteca proporciona un

conjunto de herramientas para la construcción de interfaces gráficas que tienen una apariencia y se comportan de forma semejante en todas las plataformas en las que se ejecuten. La estructura básica de la biblioteca gira en torno a componentes y contenedores. Los contenedores contienen componentes y son componentes a su vez, de forma que los eventos pueden tratarse tanto en contenedores como en componentes. La API está constituida por clases, interfaces y derivaciones. AWT y Swing

SOCKETS

Los sockets son puntos finales de enlaces de comunicaciones entre procesos. Los procesos los tratan como descriptores de ficheros, de forma que se pueden intercambiar datos con otros procesos transmitiendo y recibiendo a través de sockets [2].

El tipo de sockets describe la forma en la que se transfiere información a través de ese socket.

Sockets Stream (TCP, Transport Control Protocol)

Son un servicio orientado a conexión donde los datos se transfieren sin encuadrarlos en registros o bloques. Si se rompe la conexión entre los procesos, éstos serán informados.

El protocolo de comunicaciones con streams es un protocolo orientado a conexión, ya que para establecer una comunicación utilizando el protocolo TCP, hay que establecer en primer lugar una conexión entre un par de sockets. Mientras uno de los sockets atiende peticiones de conexión (servidor), el otro solicita una conexión (cliente). Una vez que los dos sockets estén conectados, se pueden utilizar para transmitir datos en ambas direcciones [3].

Sockets Datagrama (UDP, User Datagram Protocol)

Son un servicio de transporte sin conexión. Son más eficientes que TCP, pero no está garantizada la fiabilidad. Los datos se envían y reciben en paquetes, cuya entrega no

está garantizada. Los paquetes pueden ser duplicados, perdidos o llegar en un orden diferente al que se envió.

El protocolo de comunicaciones con datagramas es un protocolo sin conexión, es decir, cada vez que se envíen datagramas es necesario enviar el descriptor del socket local y la dirección del socket que debe recibir el datagrama. Como se puede ver, hay que enviar datos adicionales cada vez que se realice una comunicación.

KeyListener

Es una interfaz que se ocupa de los cambios en el estado de las teclas de nuestro teclado. Como sugiere el nombre de la interfaz, escucha las claves y actúa en consecuencia [4].

4. Análisis De Resultado

En este programa se presentará la compra de productos, y la emisión de la factura respectiva.

UML del proyecto:



Explicación de los sockets:

Para los sockets se hace uso del protocolo TCP mencionado, para que en este caso el Cliente envíe los productos, primero se hace la escritura en un archivo de texto y este se guarda en el disco duro de otra computadora, para establecer correctamente una conexión Cliente/Servidor es necesario conocer el host del Servidor y seguido el puerto, se escogió el puerto 12500, con la Ipv4 del servidor, ingresando por comando de Windows el comando "ipconfig", de esta manera se establece la ip, para que funcione en otras pc's, tendría que cambiarse este host

al de la computadora a la que se desea enviar el archivo de texto, primero se inicia el Servidor, luego el cliente, para así hacer una conexión exitosa [5].

```
Datos cargados exitosamente  
Socket is closed
```

Luego de la apertura y cerrada de los sockets, se abre la interfaz que permitirá al usuario dejarle hacer compras de unos productos anteriormente creados en la clase Productos, y de igual manera pasados por el archivo de texto, estos se leen en el main y carga esos datos para poder ser usados.

El código muestra una interfaz gráfica para que los datos del usuario sean ingresados por teclado, validando que los datos proporcionados sean correctos tales como nombre, correo, teléfono y cedula.


Ya validado los datos con el botón “Ir a Carrito”, se mostrará la interfaz gráfica de la factura en la cual al lado izquierdo se mostrará los productos disponibles en bodega para comprar.

La interfaz de factura permitirá ingresar el código del producto y la cantidad que se desea adquirir

Funcionamiento de la aplicación:

Al iniciar el proyecto se mostrará una interfaz gráfica para que el usuario ingrese sus datos personales estos los utilizaremos en la factura, el usuario tiene que ingresar todos los datos pedidos caso contrario no se le permitirá avanzar, se utiliza excepciones para que la información ingresada sea validada.

Ventana Cliente

 **Supermarket CCP**

Usuario

Correo Electrónico

Número de Celular

Cédula/RUC

Message



 Ingresa todos los campos

Figura 1 Interfaz gráfica con los datos incompletos



Ventana Cliente

 **Supermarket CCP**

Usuario

Correo Electrónico

Número de Celular

Cédula/RUC

Figura 2 Interfaz gráfica con los datos ingresados correctamente

Si se ingresan los datos correctamente, se da clic en el botón Ir al carrito y se mostrará una nueva interfaz gráfica en la cual se tiene que mirar los productos y cantidad disponible en la parte izquierda de la interfaz y según esta información llenar por consola lo pedido en el JTextField del Código del producto y JTextField Cantidad a comprar esta información depende de lo que el usuario desea comprar, seguido se da clic en agregar y este producto se agregará a la tabla de la derecha de la interfaz gráfica, además si se desea eliminar cierta cantidad del producto o todo el producto se tiene que ingresar los valores correspondientes y dar clic en el botón Eliminar

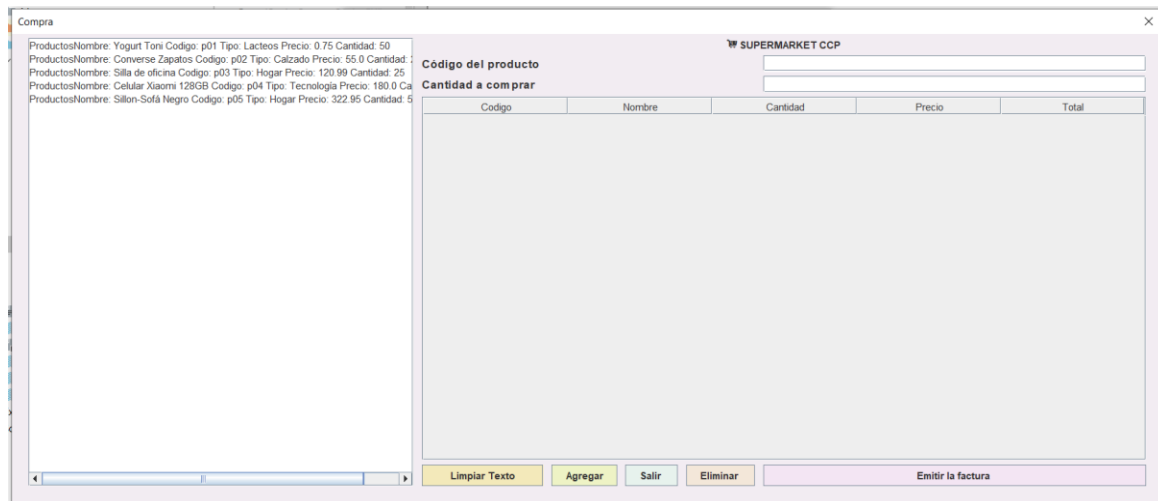


Figura 3 Interfaz gráfica del carrito de compra

Se ha ingresado una cantidad no disponible de productos por lo cual se muestra un mensaje pidiendo que se ingrese correctamente la información, lo mismo sucederá si se desea eliminar una cantidad mayor de producto agregado al carrito y si se elimina un producto que no se agrego

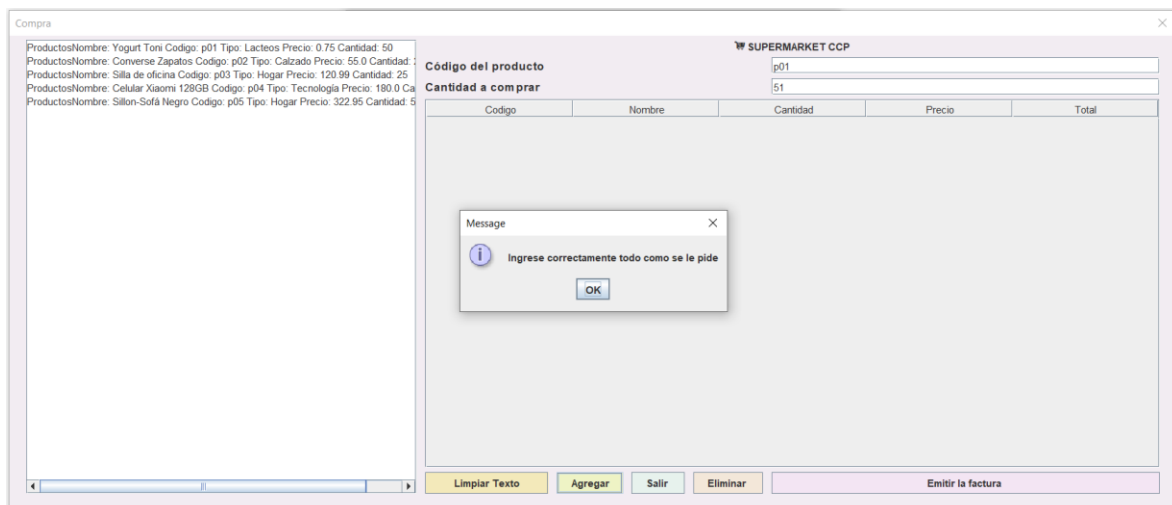


Figura 4 Interfaz gráfica queriendo ingresar una mayor cantidad de productos disponibles

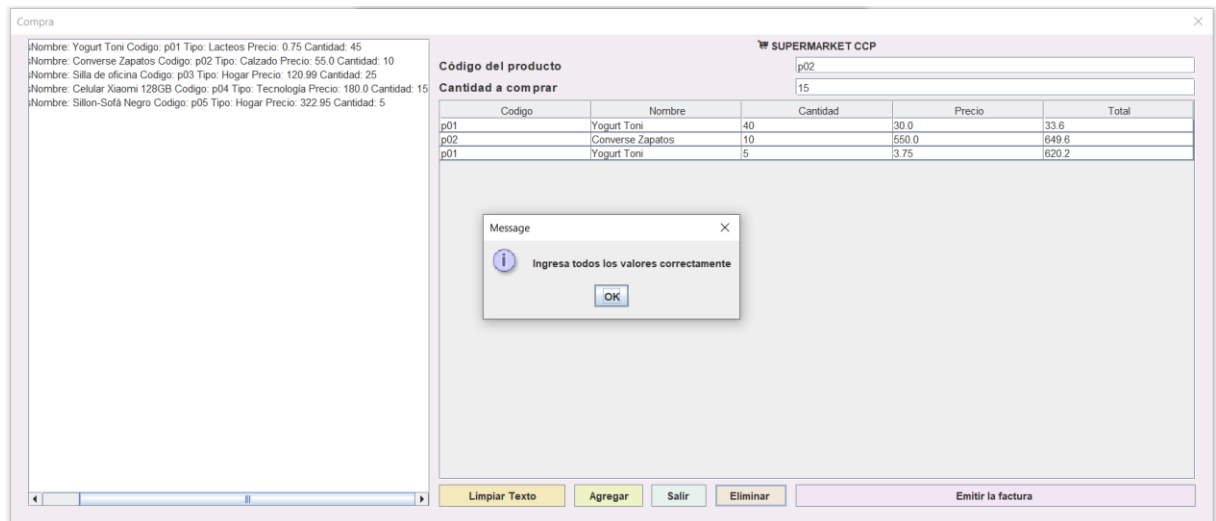


Figura 5 Interfaz gráfica queriendo eliminar una mayor cantidad de productos del que se agrego al carrito

Hemos agregado productos al carrito y podemos ver que efectivamente la cantidad de productos disponible ha disminuido

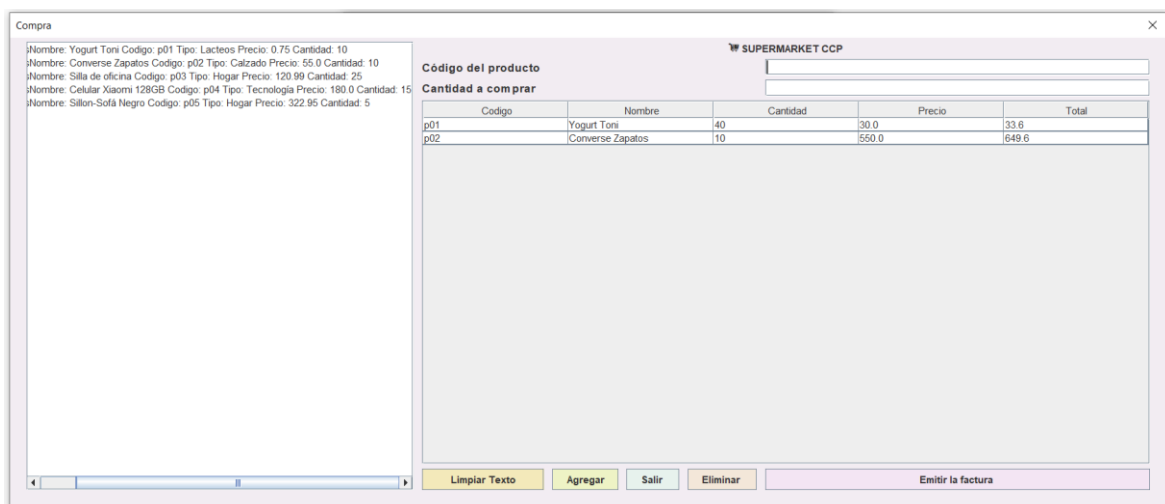


Figura 6 Interfaz gráfica haciendo uso del botón agregar

Hemos eliminado productos del carrito y podemos ver que efectivamente la cantidad de productos disponible ha aumentado

Compra

Nombre: Yogurt Toni Codigo: p01 Tipo: Lacteos Precio: 0.75 Cantidad: 45
Nombre: Converse Zapatos Codigo: p02 Tipo: Calzado Precio: 55.0 Cantidad: 10
Nombre: Silla de oficina Codigo: p03 Tipo: Hogar Precio: 120.99 Cantidad: 25
Nombre: Celular Xiaomi 128GB Codigo: p04 Tipo: Tecnologia Precio: 180.0 Cantidad: 15
Nombre: Sillon-Sofa Negro Codigo: p05 Tipo: Hogar Precio: 322.95 Cantidad: 5

SUPERMARKET CCP

Código del producto

Cantidad a comprar

Codigo	Nombre	Cantidad	Precio	Total
p01	Yogurt Toni	40	30.0	33.6
p02	Converse Zapatos	10	550.0	649.6
p01	Yogurt Toni	5	3.75	620.2

Limpiar Texto Agregar Salir Eliminar Emitir la factura

Figura 7 Interfaz gráfica haciendo uso del botón eliminar

Si ingresamos información errónea damos clic en Limpiar Texto y se borrará la misma como se muestra en la figura

SUPERMARKET CCP

Código del producto

Cantidad a comprar

Codigo	Nombre	Cantidad	Precio	Total
p01	Yogurt Toni	40	30.0	33.6
p02	Converse Zapatos	10	550.0	649.6
p01	Yogurt Toni	5	3.75	620.2

Limpiar Texto Agregar Salir Eliminar Emitir la factura

SUPERMARKET CCP

Código del producto

Cantidad a comprar

Codigo	Nombre	Cantidad	Precio	Total
p01	Yogurt Toni	40	30.0	33.6
p02	Converse Zapatos	10	550.0	649.6
p01	Yogurt Toni	5	3.75	620.2

Limpiar Texto Agregar Salir Eliminar Emitir la factura

Figura 8 Interfaz gráfica haciendo uso del botón Limpiar Texto

Observemos que si no ingresamos ningún producto no se podrá emitir una factura como se muestra en la figura

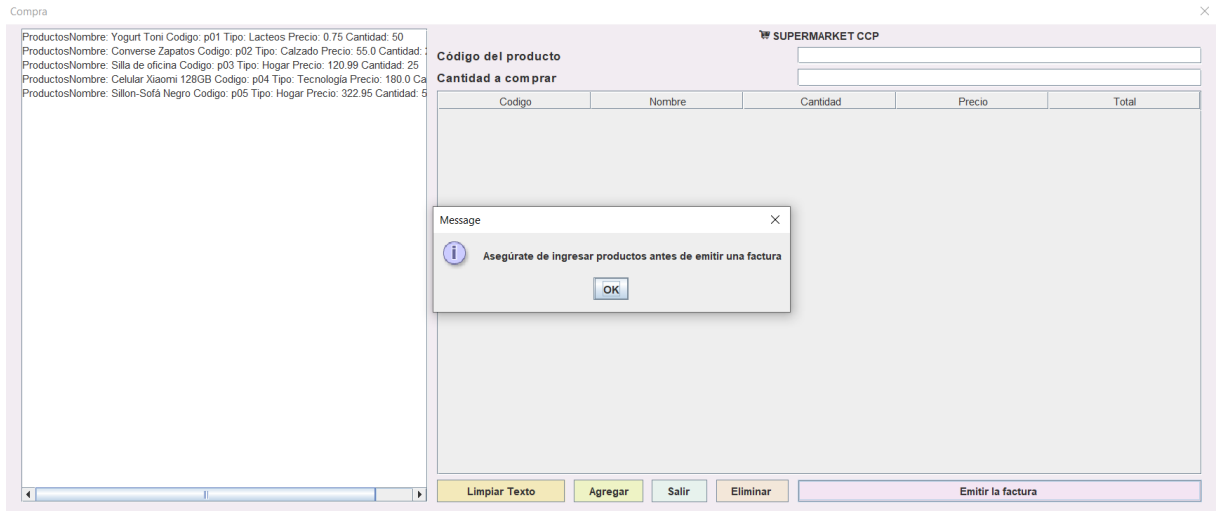


Figura 9 Interfaz gráfica haciendo uso del botón emitir la factura sin agregar ningún producto

```

=====FACTURA=====

SuperMarket CCP
Escuela Politécnica Nacional
0982390699

Nombre: Pedro Gonzáles Cédula: 1308800729 Correo: pedrito@gmail.com Número de teléfono: 0992342942
Nombre: Yogurt Toni Codigo: p01 Tipo: Lacteos Precio: 0.75 Cantidad: 4
Nombre: Silla de oficina Codigo: p03 Tipo: Hogar Precio: 120.99 Cantidad: 2
Nombre: Celular Xiaomi 128GB Codigo: p04 Tipo: Tecnología Precio: 180.0 Cantidad: 1
Subtotal: 424.98
IVA 12%: 50.99759999999999
Total: 475.9776
¡Gracias Por su Compra!
  
```

Figura 10: Creación de un archivo de texto que muestra la factura

Finalmente, ya ingresado solo lo que deseamos comprar al carrito damos clic en Emitir la factura y se podra observar que se creó un archivo.txt en el escritorio su contenido será una factura con los datos, los productos y el total a pagar, gracias a los métodos toString del polimorfismo de Java.

Respecto a la documentación:

Después de realizado el proyecto, se procedió a hacer la documentación con la herramienta de generar Javadoc que proporciona IntelliJIdea, de esta manera se escribió todo lo posible de los métodos, variables implicadas, y objetos existentes en el proyecto, aquí se observa parte del proyecto documentado:

Package CarritoCompra

package CarritoCompra

Classes	
Class	Description
AppCarrito	Clase AppCarrito, la clase Main principal que ayudará a que se ejecute la totalidad del programa, se compone de la clase ServidorProductos
Bodega	Clase Bodega
CarritoGUI	Clase CarritoGUI
ClienteMandarArchivo	Clase ClienteMandarArchivos
Data	Clase Data se crean los ArrayList de productos y mientras se va agregando los productos se creará el ArrayList de productosVendidos Se crea un nuevo usuarioTienda, bodegaTienda y tiendaGeneral
Producto	Clase Producto
SerialWrite	Clase SerialWrite, en donde se crean objetos de todas las otras clases y se interactúan con ellas Esta clase se creó de inicio en el proyecto simplemente para escribir en un archivo de texto los objetos de la clase Producto, de tal manera que luego de su escritura, no se volvió a usar.
ServidorProductos	Clase ServidorProductos
Tienda	Clase Tienda
Usuario	Clase Tienda
UsuarioGUI	Clase UsuarioGui

Figura 11: Menú principal del Javadoc generado del proyecto

Observe la clase más general productos, para hacer una interpretación de todo el proyecto y su documentación:

Package CarritoCompra

Class Producto

`java.lang.Object`
`CarritoCompra.Producto`

All Implemented Interfaces:

`Serializable`

```
public class Producto
extends Object
implements Serializable
```

Clase Producto

Since:

1 de septiembre de 2022

Version:

1.0.0.0

Author:

Grupo Programacion II GR3SW

See Also:

[Serialized Form](#)

Modifier and Type	Field	Description
private String [Ⓔ]	codigoProducto	Campo que almacena en un formato String el valor de codigoProducto
private int	existenciasProducto	Campo que almacena en un formato String el valor de existenciasProducto
private String [Ⓔ]	nombreProducto	Campo que almacena en un formato String el valor de nombreProducto
private double	precioUnitario	Campo que almacena en un formato String el valor de precioUnitario
private String [Ⓔ]	tipoProducto	Campo que almacena en un formato String el valor de tipoProducto

Constructors	
Constructor	Description
Producto(String [Ⓔ] nombreProducto, String [Ⓔ] codigoProducto, String [Ⓔ] tipoProducto, double precioUnitario, int cantidadProducto)	Constructor que inicia los valores para el objeto Producto

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
int	getCantidadProducto()	Función que permite obtener el valor de existenciasProducto
String [Ⓔ]	getCodigoProducto()	Función que permite obtener el valor de codigoProducto
String [Ⓔ]	getNombreProducto()	Función que permite obtener el valor de nombreProducto
double	getPrecioUnitario()	Función que permite obtener el valor de precioUnitario
String [Ⓔ]	getTipoProducto()	Función que permite obtener el valor de tipoProduct
String [Ⓔ] []	prodToArray(int cant, double total)	
void	setCodigoProducto(String [Ⓔ] codigoProducto)	Función que permite modificar el valor de codigoProducto
void	setExistenciasProducto(int existenciasProducto)	Función que permite modificar el valor de existenciasProducto
void	setNombreProducto(String [Ⓔ] nombreProducto)	Función que permite modificar el valor de nombreProducto
void	setPrecioUnitario(double precioUnitario)	Función que permite modificar el valor de precioUnitario
void	setTipoProducto(String [Ⓔ] tipoProducto)	Función que permite modificar el valor de tipoProducto
String [Ⓔ]	toString()	Imprime el tipo de objeto específico de Producto
Methods inherited from class java.lang.Object [Ⓔ]		
clone [Ⓔ] , equals [Ⓔ] , finalize [Ⓔ] , getClass [Ⓔ] , hashCode [Ⓔ] , notify [Ⓔ] , notifyAll [Ⓔ] , wait [Ⓔ] , wait [Ⓔ] , wait [Ⓔ]		

Figura 12, 13, 14: Clase Producto del proyecto

5. Conclusiones

- Se logro comunicar dos pc (cliente-servidor) mediante el uso de sockets en el código, primero ejecutando la clase AppCarrito y en la otra computadora la clase Cliente para que logren tener conexión y verificamos con una

impresión por consola que contiene el siguiente mensaje “Datos cargados exitosamente” en el proyecto final.

- Se ha implementado dos interfaces gráficas la primera para que el usuario ingrese sus datos y la segunda para que pueda interactuar con el carrito agregando y eliminando productos del carrito, además se puede generar una factura, cabe recalcar el uso de GUI.form ayuda a que el entorno gráfico sea más agradable a la vista.
- Se realizó validaciones en todo el proyecto para validar los datos personales del usuario con KeyListener y también con Excepciones para evitar que se ingrese una información errónea al momento de agregar o eliminar productos, con la finalidad de que se muestren mensajes en la pantalla y que el usuario lo vuelva intentar evitando que la ejecución pare si no se desea.

6. Recomendaciones

- Se recomienda validar la información del usuario cuando ya esté terminado el proyecto para evitar pérdida y dirigirnos rápidamente a la Interfaz de AppCarrito, o hacer el uso del método main para solo ejecutar lo que aún nos falta terminar o corregir.
- Se recomienda usar Excepciones para que el programa no se detenga abruptamente debido a errores que puedan romper el código.
- Se recomienda probar cada método o función finalizada para comprobar que esta funcione correctamente.

7. Bibliografía

- [C. Cervigón, «Interfaces Gráficas de Usuario - POO - Tema 6,» [En línea]. Available:
1 <https://www.fdi.ucm.es/profesor/jpavon/poo/Tema6resumido.pdf>. [Último acceso: 31 Agosto 2022].
]
- [ITLP MX, «Tutorial de Java, Cap 9 Socket,» [En línea]. Available:
2 <http://www.itlp.edu.mx/web/java/Tutorial%20de%20Java/Cap9/socket.html#:~:text=Los%20sockets%20son%20puntos%20finales,recibiendo%20a%20trav%C3%A9s%20de%20sockets>. [Último acceso: 31 Agosto 2022].
]

[GeeksForGeeks, «Socket Programming in Java,» 08 Noviembre 2021. [En línea]. Available:
3 <https://www.geeksforgeeks.org/socket-programming-in-java/>. [Último acceso: 30 Agosto 2022].
]

[DelftStack, «Utilice KeyListener en Java,» 13 Octubre 2021. [En línea]. Available:
4 <https://www.delftstack.com/es/howto/java/java-key-listener/>. [Último acceso: 31 Agosto 2022].
]

[R. Paszniuk, «Transferencia de fichero por socket en Java,» 19 Septiembre 2013. [En línea]. Available:
5 <https://www.programacion.com.py/escritorio/java-escritorio/transferencia-de-fichero-por-socket->
] en-
java#:~:text=El%20servidor%20guardar%C3%A1%20el%20fichero%20recibido%20en%20el,leer%20el
%20fichero%20y%20enviarlo%20por%20el%20socket.. [Último acceso: 31 Agosto 2022].