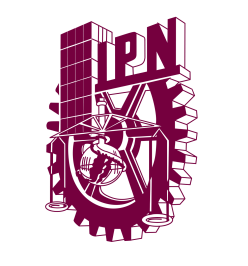
***Instituto Politécnico Nacional***

***Escuela Superior de Cómputo***

*Aplicaciones para Comunicaciones de Red*

***Practica 2: Carrito de Compras***

***Integrantes:***

* *Ornelas García Luis Ángel*
* *Sampayo Hernández Mauro*

***Grupo: 3CM5***

***Profesor:*** *Moreno Cervantes Axel Ernesto*

**Introducción:**

Los carritos de compra son una herramienta importante para la gran mayoría de las empresas, porque les permite incrementar su alcance en el mercado global dadas sus características (difusión, alcance, disponibilidad, etc.). Las principales características que deben tener los carritos de compras de e-comerce son: exhibición del catálogo en línea y dinámico para mostrar en tiempo real la disponibilidad de productos, inclusión dinámica de la información del catálogo. Interfaz de usuario de uso intuitivo y con un diseño que permita mostrar desde 1 hasta cientos de productos de forma ordenada, así como también soporte para la realización de pago electrónico.

**Desarrollo:**

* **Artículo**

La clase artículo servirá para poder definir cada uno de los artículos obtenidos desde una Base de Datos en forma de un objeto para posteriormente ser usados en la aplicación cliente-servidor del Carrito de Compras.

Cada uno de los objetos tendrá los siguientes atributos:

* Una cadena de texto para almacenar el ID del producto.
* Una cadena de texto para almacenar el nombre del producto.
* Una cadena de texto para almacenar la descripción del producto, en caso de que esta sea null, se le pondrá por defecto la cadena “Descripción no disponible :c".
* Un flotante para el precio unitario del producto.
* Una cadena de texto para almacenar la promoción del producto en formato nxm (ej. 2x1). En caso de ser null, el Cliente pondrá la leyenda “N/A” en el apartado de promoción del Catalogo y el Carrito de Compras.
* Una cadena de texto para almacenar el nombre del archivo de la imagen que estará asociada al producto.
* Un entero para el número de existencias con las que cuenta el producto
* **Servidor**

Para la parte del Servidor, se utilizaron las clases Conexión y Operaciones (las cuales implementan una pequeña Base de Datos) para definir los artículos del lado del servidor. Tambien se decidió por implementar una clase llamada “Ticket”, además de la principal. La clase principal inicializa un objeto de tipo “Ticket”, la cual se encargará de llevar a cabo el envío de la lista de los Artículos al Cliente, el cuál se encargará de mostrarlos en un catálogo, y a su vez, recibir la lista de artículos del carrito una vez el Cliente haya terminado una operación de compra, para así generarle un ticket. El servidor solo da servicio en el puerto 9090.

**Clase Ticket:**

La clase “Ticket” es de tipo JFrame y se encargará de inicializar 5 funciones en el siguiente orden: **“EnviarArticulos”, “RecibirListaCompras” “iniciarComponentes”** y **“ActualizarBaseDatos**”. Las últimas 3 se hallan dentro de un ciclo infinito para que de esta manera el cliente pueda generar más de un ticket sin que el Servidor se cierre, aunque con la desventaja de que este se deba reiniciar si es que se desea abrir el catálogo de nuevo, ya que de momento no logramos implementar una mejor solución para este problema.

La función “**EnviarArticulos”** Toma de nuestra base de datos el ID, NOMBRE, DESCRIPCION, PRECIO, PROMOCION, EXISTENCIAS e IMAGEN de nuestro producto, y por medio de un socket de servidor se encargará de enviar todos los objetos de tipo “Articulo” que se encuentran en la base de datos definida”.

La función **“RecibirListaCompras”** utilizará el mismo socket que se usó previamente en **“EnviarArticulos”,** pero en esta ocasión, una vez el cliente realice una operación de compra, este recibirá la lista de productos del carrito, así como también un hashmap que contendrá el número de unidades compradas por el cliente, y los cuales serán guardadas de una lista dinámica llamada “listacarrito” y un hashmap llamado “numproductos” respectivamente.

La función **“iniciarComponentes”** tendrá dos funciones: **“DefinirEncabezado”** y **“Generar Ticket”.**

**“DefinirEncabezado”** realizará el encabezado del ticket, el cual contendrá el título “Ticket de Compra”, la fecha y hora de la realización la compra, y los títulos de los atributos que se mostrarán de cada producto en el ticket.

**“Generar Ticket”** toma cada uno de los objetos tipo “Articulo” enlistados en el carrito, para así obtener y mostrar el nombre de dicho artículo y su precio unitario, así como también obtener la cantidad de unidades solicitadas del hashmap “numproductos”.

Tras esto, se realiza la obtención del precio a pagar por cada producto. Si el producto no tiene promoción, simplemente se realiza la multiplicación del precio unitario con la cantidad de unidades solicitada. En caso de que el producto tenga promoción, se ejecutará la función **“ObtenerPromocion”** el cual tomará el dato del atributo “promoción” del producto, cuyo tipo de dato es String, y tiene el formato “nxm”, para así obtener n y m, y convertir ambos en enteros. Posteriormente aplicamos esta ecuación:

Donde:

* numprod: Numero de productos solicitados por el cliente
* $: Precio unitario del producto
* div= Cociente de
* res= Restante de

Finalmente, se realiza la obtención del subtotal, IVA (16%) y total a pagar del producto al igual que los encabezados para indicar que tipo de cifra se está representando en el ticket de compra.

El Subtotal se obtiene mediante la suma de todos precios totales a pagar de cada producto, el IVA mediante la multiplicación del Subtotal y 0.16, y el total sumando el subtotal y el IVA

Por último, se tiene la función **“ActualizarBaseDatos**” Usa la función update de la clase Operaciones para modificar el numero de EXISTENCIA del articulo con el comando SQLite UPDATE.

* **Cliente**

En esta parte se realizará toda la lógica para mostrar los artículos recibidos desde el Servidor, así como también la lógica tras las operaciones para agregar, eliminar o modificar productos en el carrito de compras.

Para la parte del Cliente, se decidió por implementar 3 clases (Solicitud, Catalogo y Carrito) además de la principal. La clase principal inicializa un objeto de tipo “Catalogo”, la cual se encargará de llevar a cabo toda la lógica para desplegar el catálogo del carrito de compras, así como también inicializar los objetos de tipo “Solicitud” y “Carrito” cuando sean requeridos.

**Clase Solicitud:**

La clase “Solicitud”, es de tipo JDialog, y esta se inicializará una vez la clase “Catalogo” se inicialice también y ejecute la función **“RecibirArticulos”.** Esta desplegará una ventana que le pedirá al usuario que digite la dirección del servidor al cual desea conectarse para así obtener el catálogo, para posteriormente presionar el botón Solicitar. Si el cliente digita una dirección incorrecta habrá un error.

**Clase Catalogo:**

Una vez el usuario haya hecho la solicitud, se obtendrá los datos escritos por el usuario, y se utilizarán para realizar la conexión con el servidor, y recibir los artículos que se mostrarán en el artículo desde este último, así como también el número de artículos recibidos. Los artículos recibidos se guardarán en un arreglo dinámico de objetos de tipo “Articulo”, llamado “artículos”.

Tras obtener los artículos desde el servidor, se procederá a ejecutar la función **“iniciarComponentes”**, la cual tendrá tres funciones que se ejecutaran en el siguiente orden:

* **“DefinirEncabezado”**, que se encargará de definir un encabezado en la parte superior de la página que servirá para indicar al usuario que dato en específico del artículo se muestra en el catálogo.
* **“DefinirBotones”**, la cual definirá los botones “Comprar” y “Ver Carrito” cuyas funciones se explicarán más adelante, y
* **“AñadirContenido”** que se explicara más adelante.

La función “**AñadirContenido”** se encargará de llenar el contenido del catálogo, poniendo una imagen alusiva de cada uno de los productos disponibles, seguido por el nombre del artículo, su descripción, su precio, si es que tiene promoción o no, y en caso de tenerla se mostrara cual es, y una lista desplegable que contiene una numeración desde el 0 hasta el n, donde n es el número de existencias del producto en cuestión.

Al costado derecho de cada artículo, se mostrará el botón “Agregar”, definido por un arreglo de botones, para que de esta manera cada artículo tenga el proceso del botón, asociado a cada uno de los productos por separado.

Al presionar “Agregar”, se tomará el valor que se tenga seleccionado en ese momento en la lista desplegable del producto(en caso de ser 0 no se hará nada), y restara dicho valor, con el del que se tenga asociado en el atributo de “existencias” del producto en ese momento, para después sustituir dicho atributo con el del resultado de dicha resta. Este resultado se guardará también en un Hashmap llamado “existencia”, que funcionará como un auxiliar posteriormente. Procedemos a guardar el artículo en un HashMap llamado “listacarrito” que como su nombre indica servirá para guardar los productos en el carrito, así como también guardaremos el número de unidades seleccionadas por el usuario en un HashMap llamado “numproductos”. En caso de que el artículo ya haya sido añadido previamente a la lista del carrito únicamente se actualizara el valor de numero de unidades seleccionadas por el usuario. Finalmente se procederá a actualizar la lista desplegable para que solo despliegue el número de existencias restantes tras esta operación.

El botón “Agregar” está configurado para deshabilitarse una vez el numero de existencias del producto llegue a 0.

Una vez el usuario haya terminado de realizar su elección de artículos que desea comprar, este podrá utilizar cualquiera de los dos botones definidos previamente en la función **“DefinirBotones”.**

Si el usuario selecciona el botón “Comprar”, el programa procederá a realizar una nueva conexión con el servidor para que este reciba los artículos contenidos en la lista del carrito, así como también la cantidad de unidades que desea de cada uno de los productos, y así se le pueda realizar un ticket de compra. Posterior a esto, se realizará el vaciado de los hashmaps pertenecientes a la lista del carrito y al número de unidades de cada artículo solicitado por el cliente para que así se pueda realizar una nueva operación de compra desde 0.

Por otro lado, si se selecciona el botón “Ver Carrito”, se creará una instancia del objeto “Carrito”, y se le mandaran los hashmaps pertenecientes a la lista del carrito, las existencias de los productos y el número de unidades pedidas por el cliente de cada producto. Cabe resaltar que la clase “Carrito” cuenta con sus propios hashmaps equivalentes a los hashmaps que le son enviados desde “Catalogo”, y a los cuales se les igualará con los hashmaps que recibe desde esta última; además, cuando afectamos uno de los valores de los hashmaps de la clase “Carrito”, estos también afectaran los contenidos en la clase “Catalogo”.

**Clase Carrito:**

La clase “Carrito” es de tipo JDialog. Como ya se mencionó anteriormente, tiene hashmaps a los cuales se les igualará con los hashmaps que recibe desde **“Cataogo”**. Tras haberse realizado esto, se ejecutará la función **“iniciarComponentes”**, la cual, al igual que su equivalente en “Catalogo”, tendrá las funciones “DefinirEncabezado”, “DefinirBotones” y **“AñadirContenido”**, las cuales se ejecutarán en el mismo orden y realizaran funciones con la misma lógica que en “Contenido”, pero con algunas diferencias en el contenido que mostraran.

En este caso, el contenido de **“DefinirEncabezado”** y **“AñadirContenido”** se adapta para mostrar una imagen alusiva de cada uno de los productos de la lista del carrito, seguidos del nombre del producto, la cantidad de unidades del producto y el descuento del producto en caso de tenerlo. También se incluirá el precio total a pagar por cada producto, el cual se obtiene mediante la multiplicación de la cantidad de productos solicitada y el precio unitario del producto si es que este no tiene promoción.

En caso de que el producto tenga promoción, se ejecutará la función **“ObtenerPromocion”** el cual tomará el dato del atributo “promoción” del artículo, cuyo tipo de dato es String, y tiene el formato “nxm”, para así obtener n y m, y convertir ambos en enteros. Posteriormente aplicamos esta ecuación:

Donde:

* numprod: Numero de productos solicitados por el cliente
* $: Precio unitario del producto
* div= Cociente de
* res= Restante de

Adicionalmente dentro de la función **“AñadirContenido”** se tienen 3 arreglos de botones, uno el cual se ubica en el costado derecho de cada artículo en el carrito, así como el botón “Agregar” en “Catalogo” el cual tiene una “X”(el cual llamaremos Eliminar), y dos a ambos lados del apartado en donde se muestra la cantidad de productos, los cuales tienen los caracteres “>”(el cual llamaremos Añadir) y “<”(el cual llamaremos Quitar).

El botón “Eliminar” tiene la función de eliminar de la lista un producto en su totalidad, es decir, quita todas las unidades del producto solicitadas por el cliente, y las adiciona nuevamente dentro de las existencias del producto en cuestión, así como también dicho producto del hashmap del carrito de compras. El botón también se encargará de volver a cargar todo el contenido desde 0 para volver a imprimir todos los elementos de la lista del carrito, con la obvia excepción del producto que hemos removido, al ya no estar más en la lista del carrito. En caso de que se eliminen todos los productos del carrito, este mostrara la leyenda “No hay artículos en el carrito”.

El botón “Añadir” se encarga de aumentar una unidad a la cantidad de unidades del producto solicitado por el cliente de la lista del carrito, siempre y cuando este sea menor o igual al número de existencias del producto. Cuando la cantidad llega al límite máximo de unidades, es decir, las existencias con las que el producto cuenta, este se deshabilitará, y no volverá a habilitarse a no ser que el cliente disminuya en 1 la cantidad de unidades por medio del botón “Quitar” explicado más adelante.

Cada que sea presionado el botón “Añadir” al hashmap auxiliar de “existencia” se le restara 1, y al hashmap que guarda el número de unidades del producto solicitadas por el cliente se le sumara 1. Al atributo “existencias” del artículo en cuestión también se le restará 1. En caso de que el articulo cuente con una promoción, se llamará a la función **“ObtenerPromocion”** y se realizara el mismo proceso anteriormente descrito en la función **“AñadirContenido”**. Asimismo, el apartado que muestra la cantidad de unidades y el precio total se actualizarán.

El botón “Quitar” realiza prácticamente lo mismo que el botón “Añadir” solo que disminuyendo una unidad a la cantidad de unidades del producto, es decir, que en este caso al hashmap “existencia” y al atributo de existencias del producto se les sumara 1, y al hashmap de “numproductos” se le restara 1.

A diferencia del botón “Añadir”, una vez el botón “Quitar” llegue al límite mínimo de unidades que puede tener el producto, que en todos los casos será 0, este botón no se deshabilitará, sino que se eliminará totalmente de la lista del carrito similar a lo que se realiza al presionar el botón “Eliminar”.

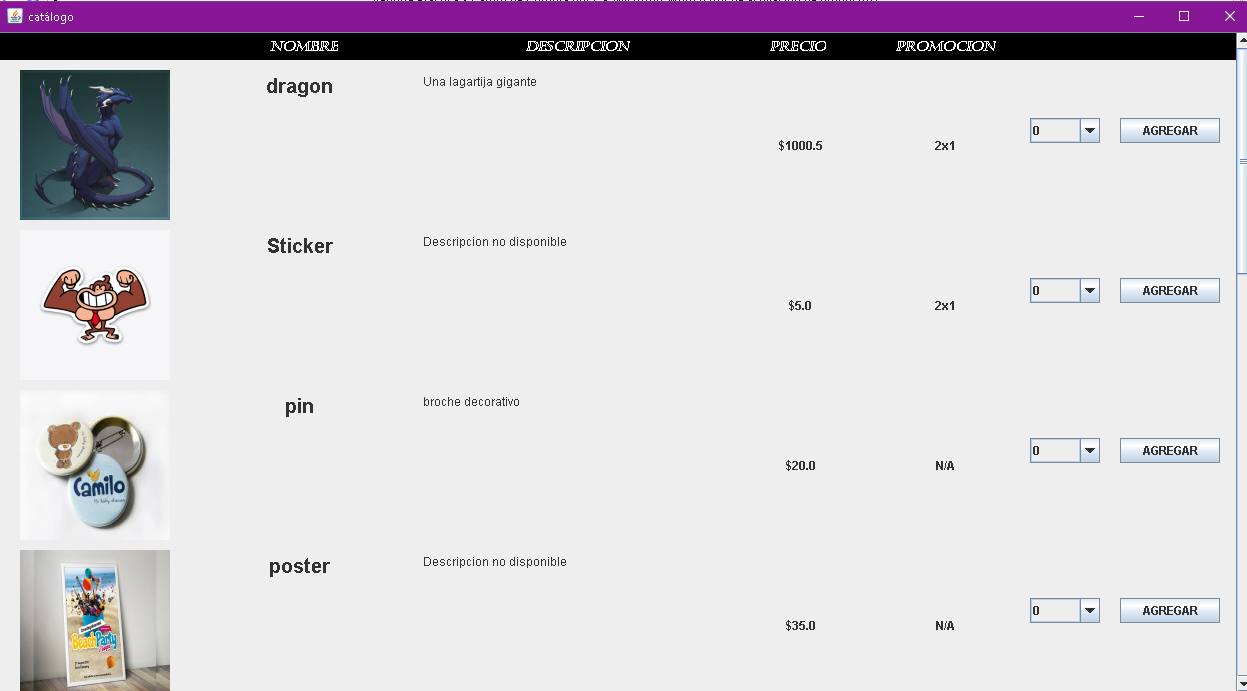
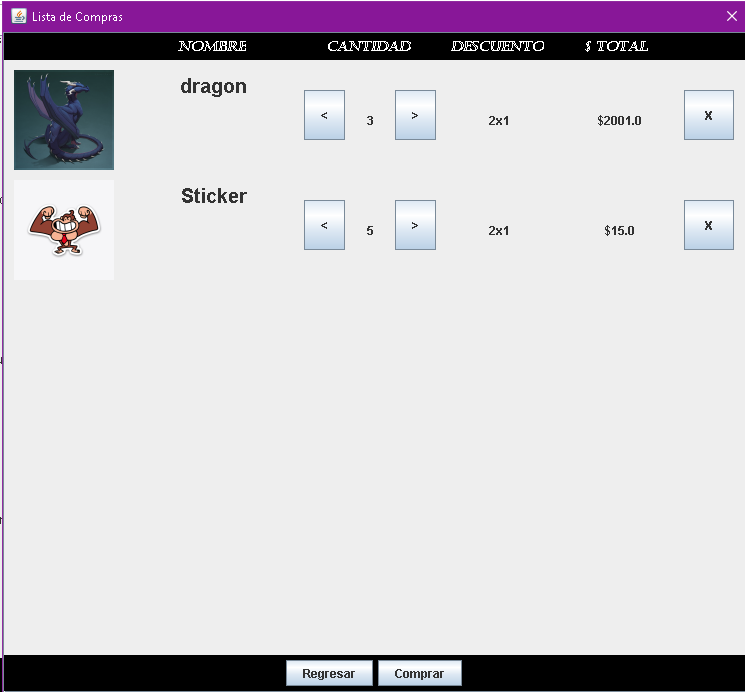
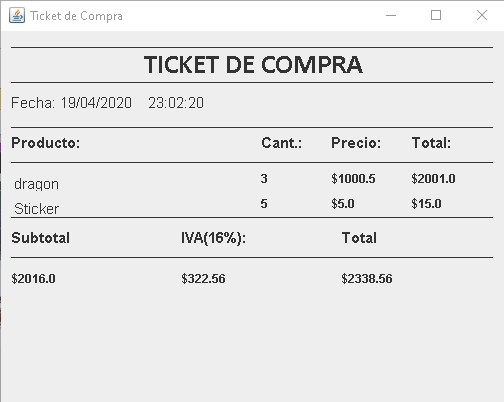
Finalmente, en la función **“DefinirBotones”** se definirán dos botones en la parte inferior del JFrame, al igual que lo que sucede con esta misma función en “Catalogo”, con la diferencia de que en este caso los botones serán “Comprar” y “Regresar”.

El botón “Comprar”, realizará exactamente lo mismo que este mismo botón, pero en la clase “Catalogo”; y “Regresar” simplemente cerrará el carrito de compras, permitiéndonos regresar a la clase “Catalogo” y seguir realizando operaciones sobre esté.

Cabe aclarar que, una vez cerremos el carrito de compras, las modificaciones sobre los productos que hayamos realizado sobre este, también se manifestarán una vez hallamos regresado al catálogo y viceversa.

**Pruebas:**



**Conclusión:**

Los servicios de carritos de compra en línea hoy en día resultan ser una opción muy viable y sencilla para que las personas hallen y obtengan productos sin tener la necesidad de salir a la calle a conseguirlos, en especial en estos días que, debido a la situación actual, las salidas a la calle son muchas más limitadas.

**Bibliografía:**

* Oracle Help Center. 2020. *JDK 14 Documentation - Home*. [online] Available at: <https://docs.oracle.com/en/java/javase/14/>
* Docs.oracle.com. 2020. *Overview (Java Platform SE 7)*. [online] Available at: <https://docs.oracle.com/javase/7/docs/api/overview-summary.html>