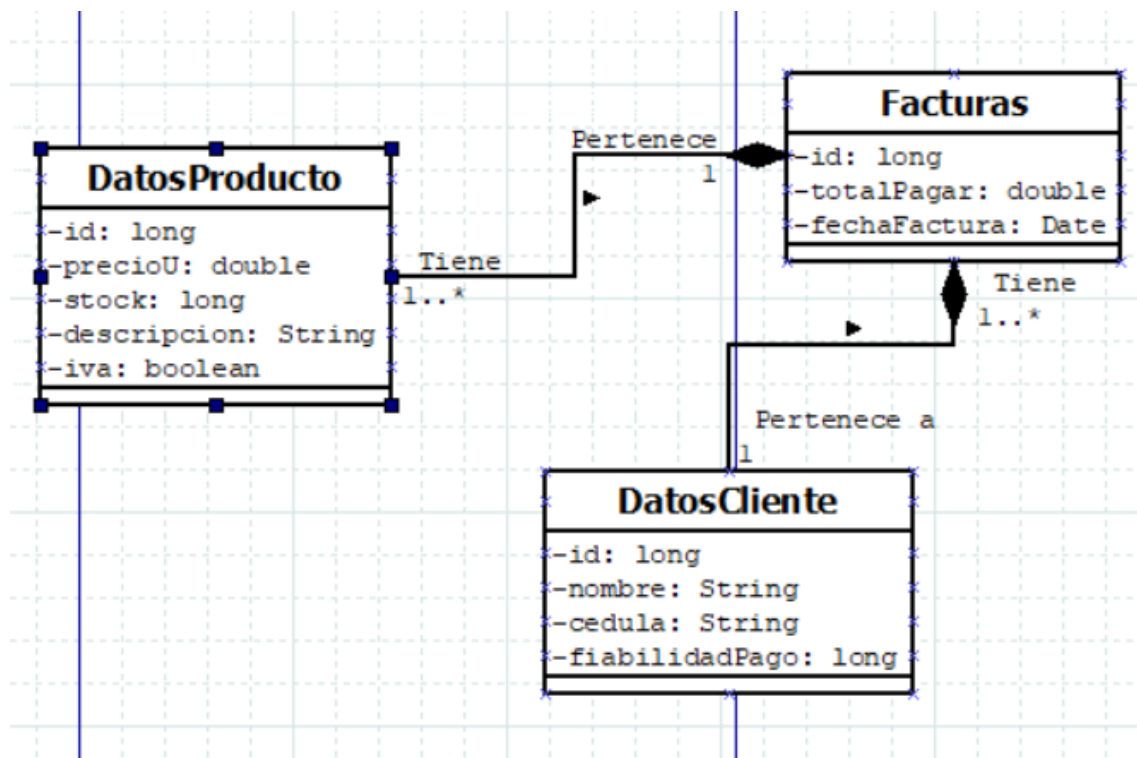


Proyecto Primer Inteciclo

Andrés Alba

El enunciado lo que nos dice que existen 3 clases las cuales son Datos del cliente (DatosCliente), datos del producto (datosProducto) y la clase factura, la relación que existe entre estas 3 clases es que para crear una factura mismo necesitaremos de la clase DatosCliente y datosProducto por lo que nos producen 2 composiciones unidireccionales, ya que una composición significa que esta clase no puede existir si la otra aún no existe, así que en nuestro programa necesitamos 2 objetos ya creados, 1 que será la instancia de la clase DatosCliente y otra de datosProducto.



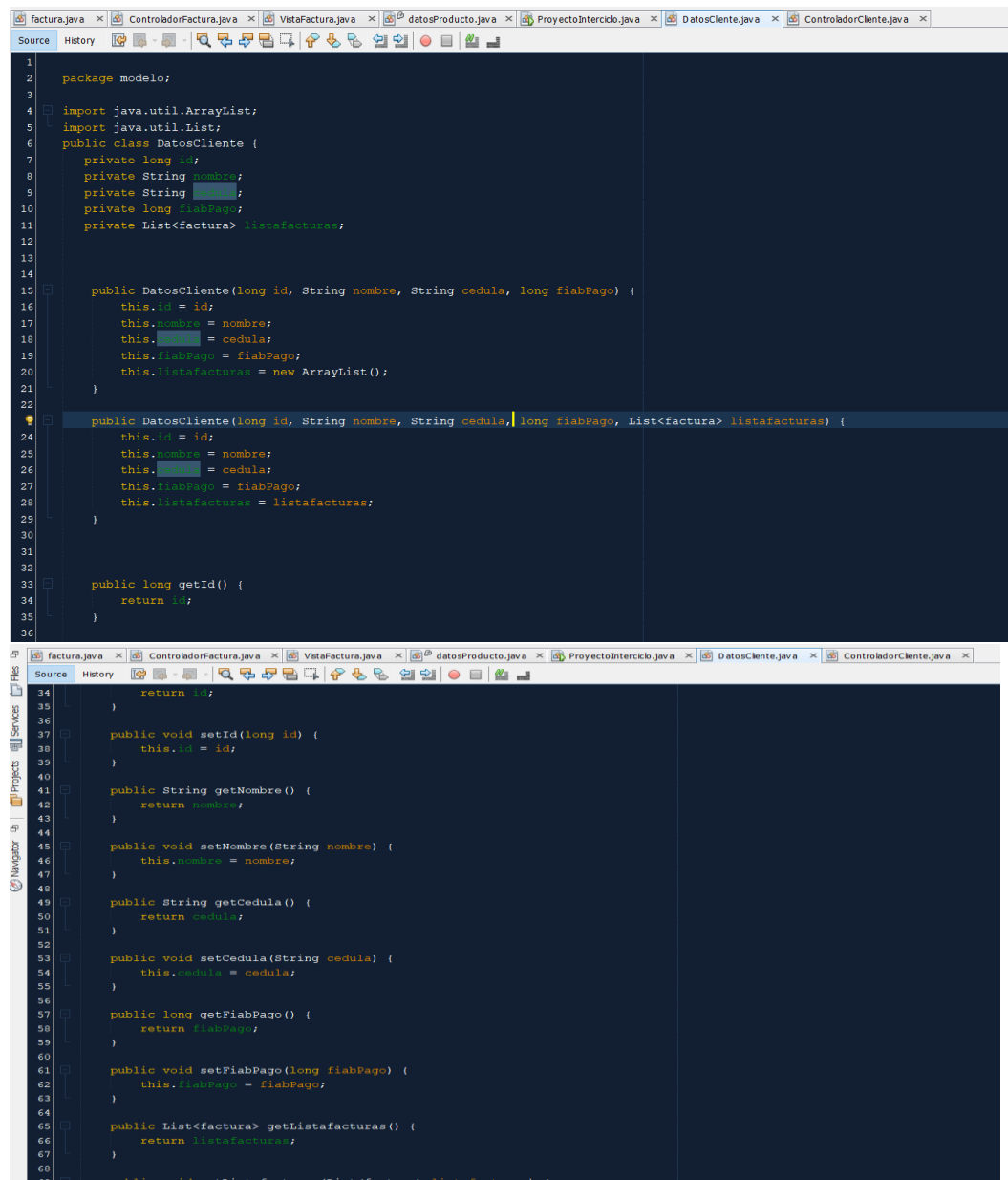
- Modelos

Para todos los modelos se siguió la misma estructura

1. Crear las variables privadas
2. Solo en factura, Crearemos las variables con los nombres de las clases que necesitaremos, donde guardaremos los datos que necesitamos para nuestra factura, los datos de los productos y los datos del cliente.
3. Crear los constructores necesarios para todas las variables, y las listas y los objetos.
4. Crearemos los getters y setters de todas las variables para hacer el encapsulamiento y que solo se pueda entrar a los datos mediante los métodos.

5. Sobrescribiremos el método ToString, para mostrar las variables que queremos mostrar al usuario.

- **Ejemplo de mi clase DatosCliente del modelo.**



```
1 package modelo;
2
3
4 import java.util.ArrayList;
5 import java.util.List;
6 public class DatosCliente {
7     private long id;
8     private String nombre;
9     private String cedula;
10    private long fiabPago;
11    private List<factura> listaFacturas;
12
13
14
15    public DatosCliente(long id, String nombre, String cedula, long fiabPago) {
16        this.id = id;
17        this.nombre = nombre;
18        this.cedula = cedula;
19        this.fiabPago = fiabPago;
20        this.listaFacturas = new ArrayList();
21    }
22
23
24    public DatosCliente(long id, String nombre, String cedula, long fiabPago, List<factura> listaFacturas) {
25        this.id = id;
26        this.nombre = nombre;
27        this.cedula = cedula;
28        this.fiabPago = fiabPago;
29        this.listaFacturas = listaFacturas;
30    }
31
32
33    public long getId() {
34        return id;
35    }
36
37
38    public void setId(long id) {
39        this.id = id;
40    }
41
42    public String getNombre() {
43        return nombre;
44    }
45
46    public void setNombre(String nombre) {
47        this.nombre = nombre;
48    }
49
50    public String getCedula() {
51        return cedula;
52    }
53
54    public void setCedula(String cedula) {
55        this.cedula = cedula;
56    }
57
58    public long getFiabPago() {
59        return fiabPago;
60    }
61
62    public void setFiabPago(long fiabPago) {
63        this.fiabPago = fiabPago;
64    }
65
66    public List<factura> getListafacturas() {
67        return listaFacturas;
68    }
69
70    public void setListafacturas(List<factura> listaFacturas) {
```

```

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }

    public long getFiabPago() {
        return fiabPago;
    }

    public void setFiabPago(long fiabPago) {
        this.fiabPago = fiabPago;
    }

    public List<factura> getListafacturas() {
        return listafacturas;
    }

    public void setListafacturas(List<factura> listafacturas) {
        this.listafacturas = listafacturas;
    }

    @Override
    public String toString() {
        return "DatosCliente{" + "id=" + id + ", nombre=" + nombre + ", cedula=" + cedula + ", fiabPago=" + fiabPago + " $ " + '}';
    }
}

```

- Controladores

Los controladores son la parte vital del programa pues aquí va el CRUD (Create, Read, Update y Delete), y pues en los controladores necesitaremos crear 2 variables nuevas: la primera será una lista donde guardaremos los datos de las variables del modelo por lo que deberemos crear una importación del paquete modelo y la clase que necesitamos, y la segunda variable será un objeto de la clase del modelo, esto nos servirá para cuando trabajemos con la vista seleccionar correctamente la clase a la que queremos seleccionar.

- La estructura de los controladores

1. Importamos de la clase util, a ArrayList y List.
2. Importamos el modelo de la clase con la que trabajaremos.
3. Creamos las variables privadas que ya mencioné.
4. Creamos un Controlador vacío donde inicializamos las variables.
5. Creamos los getters y setters para poder acceder a estas variables mediante los métodos.
6. Creamos el método Crear () donde necesitaremos todas las variables que pidan necesite la clase y luego con esos datos obtenidos por el usuario instanciaremos un nuevo objeto en la clase modelo y el método retornara el objeto creado añadido en la lista de la clase que creamos en el controlador.
7. Creamos el método Buscar (), en este método necesitaremos una forma de reconocer a nuestro objeto así que cada clase tendrá algo que la represente de forma única, en este método recibiremos un valor a buscar en nuestra lista, así que la recorreremos analizando si es que logramos que el valor ingresado coincide con el de algún objeto. Si lo hace retornaremos el objeto encontrado y sino un null.
8. Crearemos el método Actualizar(), para este método necesitaremos del anterior creado Buscar, para confirmar si es que existe el objeto al que

buscaremos, al método actualizar le llegaran los nuevos datos, si es que el objeto si existe pues se usaran los métodos Set del modelo para actualizar los datos y luego se retornara la lista actualizada a la misma posición, pero si es que no existe el objeto retornaremos un false.

9. Crearemos el método Eliminar() donde en este método también necesitaremos de Buscar para confirmar la existencia del objeto, si es que existe el objeto se usara un método de la clase lista llamado remove, y eliminaremos el objeto, si no existe se regresara un null.

- **Ejemplo de mi clase ControladorProductos de mis controladores.**

```
package controlador;
import java.util.ArrayList;
import java.util.List;
import modelo.datosProducto;

public class ControladorProductos {
    private List<datosProducto> listaProductos;
    private datosProducto selecc;

    public ControladorProductos() {
        listaProductos = new ArrayList();
        selecc=null;
    }

    //CREATE
    public boolean crear(long id, double precioU, long stock, String descripcion, boolean iva){
        datosProducto datosProductos = new datosProducto (id, precioU, stock, descripcion, iva);
        return listaProductos.add(datosProductos);
    }

    //READ
    public datosProducto buscar(String descripcion){
        for (datosProducto datosProductos : listaProductos) {
            if(datosProductos.getDescripcion().equals(descripcion)) {
                return datosProductos;
            }
        }
        return null;
    }

    //UPDATE
    public boolean actualizar(double precioU, long stock, String descripcion, boolean iva){
        datosProducto datosProductos = this.buscar(descripcion);
        if(datosProductos != null) {
            int posicion = listaProductos.indexOf(datosProductos);
            datosProductos.setPrecioU(precioU);
        }
    }
}
```

```

//UPDATE
public boolean actualizar(double precioU, long stock, String descripcion, boolean iva){
    datosProducto datosProductos = this.buscar(descripcion);
    if(datosProductos != null) {
        int posicion = listaProductos.indexOf(datosProductos);
        datosProductos.setPrecioU(precioU);
        datosProductos.setStock(stock);
        datosProductos.setDescripcion(descripcion);
        datosProductos.setIva(iva);
        listaProductos.set(posicion, datosProductos);
        return true;
    }
    return false;
}

//DELETE
public boolean eliminar(String descripcion){
    datosProducto datosProductos = this.buscar(descripcion);
    if(datosProductos != null) {
        return listaProductos.remove(datosProductos);
    }
    return false;
}

//GETTER SETTERS
public List<datosProducto> getListaProductos() {
    return listaProductos;
}

public void setListaProductos(List<datosProducto> listaProductos) {
    this.listaProductos = listaProductos;
}

public datosProducto getSelecc() {
    return selecc;
}

public void setSelecc(datosProducto selecc) {

```

```

        return selecc;
    }

//GETTER SETTERS
public List<datosProducto> getListaProductos() {
    return listaProductos;
}

public void setListaProductos(List<datosProducto> listaProductos) {
    this.listaProductos = listaProductos;
}

public datosProducto getSelecc() {
    return selecc;
}

public void setSelecc(datosProducto selecc) {
    this.selecc = selecc;
}

public void imprimir(){
    for (datosProducto producto : listaProductos) {
        System.out.println(producto);
    }
}

}

```

(PARA LA CLASE FACTURA)

Se pidió un método Borrar el cual no debe de recibir ni retornar datos, por lo que mi solución fue crear 2 método y en este método borrar donde no recibe ni retorna datos, crear una variable volátil donde se pide la id de la factura a buscar y de ahí del método llamar al otro método Eliminar() el cual sigue la misma estructura que el explicado brevemente, donde mostramos si fue o no borrada la factura en el método que no recibe ni retorna datos.

```
//delete
public boolean eliminar(long id){
    factura facturas = this.buscar(id);
    if(facturas != null) {
        facturas.getDatosCliente().getListafacturas().remove(facturas);
        return datos.remove(facturas);
    }
    return false;
}

public void borrar(){
    long id;
    boolean factura;
    System.out.println("Ingrese la id de su factura: ");
    id=leer.nextLong();
    factura = this.eliminar(id);
    System.out.println("Factura borrada "+factura);
}
```

- Vistas

Las vistas son la parte donde interactúa el usuario y estos datos ingresados son los que se van a los controladores y estos los que conectan con el modelo.

Para las vistas existen 2 partes las vistas de las distintas clases y la vista general

-Vista de la Clases

En todas las vistas de la clase se siguió la misma estructura.

1. Importamos la clase Scanner pues la necesitaremos para que el usuario pueda ingresar sus datos.
2. Importamos las clases de los controladores que necesitaremos, en la clase factura al depender de las otras 2 clases pues deberá importar las 2 clases, en cambio las otras vistas solo su propio controlador.
3. Creamos la variable que instanciara el Scanner y creamos las variables que serán instancias de la clase de los controladores, esto nos sirve para usar los métodos de la clase de los controladores.
4. Creamos un constructor sin valores en el cual instanciaremos el Scanner y los Controladores. Excepto en facturas porque aquí

necesitaremos los otros 2 controladores de las otras clases donde los tomaremos como objetos.

5. Se crean los getters y setters de las variables creadas.
6. Crearemos el método menu () donde será la interfaz en la que se desplazara el usuario. En este método llamaremos al resto de métodos que se crearan en esta clase. Y si ingresan una opción no valida pues saldrá de la vista de la clase y volverá a la principal.
7. Creamos la clase crear () donde crearemos las variables volátiles donde se guardará todo lo que necesitemos y estos datos enviaremos mediante la instancia del controlador, aparte en la clase factura se usaran el método de selección () para seleccionar la clase y que se guarde con el objeto factura.
8. Creamos la clase buscar () pediremos la variable que debemos buscar y esta se le mandara al método del controlador.
9. Creamos la clase actualizar () donde pediremos la variable que deseamos buscar y los nuevos datos, pero solo podemos cambiar los datos de las variables que pertenecen a la clase, del resto de clases no podemos modificar los datos desde una vista que no le pertenezca, llamaremos a los métodos del controlador de la vista.
10. En el método eliminar también solo se pedirá la variable que debemos de buscar para eliminar y se le llamara al método del controlador.

- Muestra de la vista factura

```
package vista;
import controlador.ControladorCliente;
import controlador.ControladorProductos;
import controlador.ControladorFactura;
import modelo.factura;
import java.util.Scanner;

public class VistaFactura {

    private ControladorCliente controladorCliente;
    private ControladorProductos controladorProducto;
    private ControladorFactura datos;
    Scanner leer = new Scanner(System.in);

    public VistaFactura(ControladorProductos controladorProducto, ControladorCliente controladorCliente) {
        this.controladorCliente= controladorCliente;
        this.controladorProducto = controladorProducto;
        datos = new ControladorFactura();
    }

    public void menu() {
        int opcion = 0;
        do {
            System.out.println("\nGestión de factura");
            System.out.println("1. Crear");
            System.out.println("2. Actualizar");
            System.out.println("3. Buscar");
            System.out.println("4. Eliminar");
            System.out.println("5. Listar");
            System.out.println("6. Salir");
            opcion = leer.nextInt();
            switch(opcion){
                case 1: crear(); break;
            }
        }
    }
}
```

```

34         System.out.println("6. Salir");
35         opcion = leer.nextInt();
36         switch(opcion){
37             case 1: crear(); break;
38             case 2: actualizar(); break;
39             case 3: buscar(); break;
40             case 4: eliminar(); break;
41             case 5:
42                 System.out.println("Listado de facturas:");
43                 datos.imprimir();
44                 break;
45         }
46     } while (opcion < 6);
47 }
48
49 public void crear(){
50     System.out.println("Ingrese los datos de la factura:");
51
52     System.out.println("total a pagar");
53     double totalPagar = leer.nextDouble();
54     System.out.println("Ingrese el Dia de creacion de la factura");
55     int fechaDia = leer.nextInt();
56     System.out.println("Ingrese el Mes de creacion de la factura");
57     int fechaMes = leer.nextInt();
58     System.out.println("Ingrese el Año de creacion de la factura");
59     int fechaAño = leer.nextInt();
60     boolean resultado;
61     resultado= datos.crear(totalPagar,fechaMes,fechaDia,fechaAño,controladorCliente.getSelecc(),controladorProducto.getSelecc());
62     System.out.println("Producto creado: " + resultado);
63 }
64
65
66 public factura buscar(){
67     System.out.println("Ingrese ID de la factura: ");
68     long id= leer.nextLong();
69     factura factura = datos.buscar(id);
70     System.out.println(factura);

```

```

    }

    public factura buscar(){
        System.out.println("Ingrese ID de la factura: ");
        long id= leer.nextLong();
        factura factura = datos.buscar(id);
        System.out.println(factura);
        return factura;
    }

    public void actualizar(){
        System.out.println("Ingrese los datos: ");
        System.out.println("Id de la factura a buscar");
        long id = leer.nextLong();
        System.out.println("Nuevo Total a pagar");
        double totalPagar = leer.nextDouble();
        System.out.println("Ingrese el nuevo Dia de creacion de la factura");
        int fechaDia = leer.nextInt();
        System.out.println("Ingrese el nuevo Mes de creacion de la factura");
        int fechaMes = leer.nextInt();
        System.out.println("Ingrese el nuevo Año de creacion de la factura");
        int fechaAño = leer.nextInt();
        boolean resultado = datos.actualizar(id,totalPagar,fechaDia,fechaMes,fechaAño);
        System.out.println("Producto actualizado: " + resultado);
    }

    public void eliminar(){
        datos.borrar();
    }

    public void asignarSeleccionado(factura factura){
        datos.setSelecc(factura);
    }

```



```

public void eliminar() {
    datos.borrar();
}

public void asignarSeleccionado(factura factura) {
    datos.setSelecc(factura);
}

public ControladorCliente getControladorCliente() {
    return controladorCliente;
}

public void setControladorCliente(ControladorCliente controladorCliente) {
    this.controladorCliente = controladorCliente;
}

public ControladorProductos getControladorProducto() {
    return controladorProducto;
}

public void setControladorProducto(ControladorProductos controladorProducto) {
    this.controladorProducto = controladorProducto;
}

public ControladorFactura getDatos() {
    return datos;
}

public void setDatos(ControladorFactura datos) {
    this.datos = datos;
}

public Scanner getLeer() {

```

Vista principal

La clase de la vista principal es la primera vista que vera el usuario, aquí podremos movernos a través de las distintas vistas pero la para cuando una clase depende de otra la vista se bloquea hasta que exista un objeto en las clases que necesita, por lo que la clase factura estará bloqueada hasta que exista DatosCliente y Datos Productos.

Para la estructura de la vista principal:

1. Debemos de importar el Scanner y a los modelos de las clases de las que depende alguna otra, y aparte todas las vistas.
2. Creamos variables privadas que son instancias de las vistas
3. Creamos un constructor vacío donde inicializamos las variables y también el Scanner, pero al inicializar la vista que depende de otras clases también se le envían los getters de los controladores de la instancia de las vistas necesarias.
4. Creamos el método menú donde nos podremos desplazar y avanzar a los métodos del menú de las distintas clases.

5. Para la clase factura que depende de las otras se crea un nuevo método, donde confirmaremos si existen los objetos instanciados y si existen las que se necesita, se envía a la vista factura.

-Muestra de la vista principal

```
package vista;

import java.util.Scanner;
import modelo.DatosCliente;
import modelo.datosProducto;
import vista.VistaCliente;
import vista.VistaFactura;
import vista.VistaProductos;

public class VistaPrincipal {
    private VistaCliente clienteVista;
    private VistaProductos productoVista;
    private VistaFactura facturaVista;
    Scanner leer = new Scanner(System.in);

    public VistaPrincipal() {
        clienteVista = new VistaCliente();
        productoVista = new VistaProductos();
        facturaVista = new VistaFactura(productoVista.getProductosControlador(), clienteVista.getClienteControlador());
    }

    public void menu() {
        int opcion = 0;
        do {
            System.out.println("\nSeleccione una opción");
            System.out.println("1. Datos Cliente");
            System.out.println("2. Datos Producto");
            System.out.println("3. Factura");
            System.out.println("4. Salir");
            opcion = leer.nextInt();

            switch(opcion){
                case 1: clienteVista.menu(); break;
            }
        } while (opcion <= 4);
    }

    public void factura() {
        System.out.println("Seleccione el cliente para gestionar sus facturas");
        DatosCliente cliente = clienteVista.buscar();
        if(cliente != null){
            clienteVista.asignarSeleccionado(cliente);
        } else {
            System.out.println("No existe el cliente");
            this.menu();
        }
        System.out.println("Seleccione el id de la lista de productos a gestionar");
        datosProducto producto = productoVista.buscar();
        if(cliente != null){
            productoVista.asignarSeleccionado(producto);
        } else {
            System.out.println("No existen los productos");
            this.menu();
        }
        facturaVista.menu();
    }
}
```

Consideraciones

- Clase Factura: Se me complico mucho usar una variable date por el formato, si no se ingresaba correctamente por lo que el programa colapsaba, así que use una variable para el día ,fecha y año para conseguir la fecha.

Vista de la interfaz en Consola

-Menú Principal

```
Seleccione una opción
1. Datos Cliente
2. Datos Producto
3. Factura
4. Salir
```

-Menú Datos Cliente

```
Gestión de Clientes
1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir
1
Ingrese los siguientes datos:
Id:
12
Nombre:
Nicolas
Cedula:
0107555445
valor de fiabilidad de pago:
450
Cliente creado: true
```

```
Gestión de Clientes
1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir
5
```

```
5. Listar
6. Salir
5
Listado de clientes:
DatosCliente{id=1, nombre=Andres, cedula=0107300469, fiabPago=500 $ }
DatosCliente{id=12, nombre=Nicolas, cedula=0107555445, fiabPago=450 $ }
```

-Actualización

```
Gestión de Clientes
1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir
2
Actualizar
Ingrese la Cedula a buscar:
0107555445
Ingrese nuevo Nombre:
Carlos
Ingrese nuevo Valor de fiabilidad de pago:
458
Cliente actualizado: true

Gestión de Clientes
1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir
5
Listado de clientes:
DatosCliente{id=1, nombre=Andres, cedula=0107300469, fiabPago=500 $ }
DatosCliente{id=12, nombre=Carlos, cedula=0107555445, fiabPago=458 $ }
```

-Eliminar

```
3. Salir
4
Eliminar Cliente
Cedula:
0107555445
Cliente eliminado: true

Gestión de Clientes
1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir
5
Listado de clientes:
DatosCliente{id=1, nombre=Andres, cedula=0107300469, fiabPago=500 $ }
```

-Menú de productos (Funciona igual que el menú de Cliente)

Gestión de Productos

1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar/Imprimir
6. Salir

1

Ingrese los siguientes datos:

Id:

22

PrecioU:

580

stock:

3

descripcion del producto:

LAPTOP

Producto con IVA (True/False):

FALSE

Producto creado: true

Gestión de Productos

1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar/Imprimir
6. Salir

5

FALSE

Producto creado: true

Gestión de Productos

1. Crear
 2. Actualizar
 3. Buscar
 4. Eliminar
 5. Listar/Imprimir
 6. Salir
- 5

Listado de Productos:

datosProducto{id=1, precioU=125.0\$, stock=12unidades , descripcion=DELL, iva=true, SI TIENE IVA SU VALOR CON IVA ES=140.0\$}

datosProducto{id=22, precioU=580.0\$, stock=3 unidades , descripcion=LAPTOP, iva=false, NO TIENE IVA SU VALOR SIN IVA ES=580.0\$}

-Menú Principal Seleccionando factura

Seleccione una opción

1. Datos Cliente
 2. Datos Producto
 3. Factura
 4. Salir
- 3

Seleccione el cliente para gestionar sus facturas

Buscar Cedula

Cedula:

0107300469

DatosCliente{id=1, nombre=Andres, cedula=0107300469, fiabPago=500 \$ }

Seleccione el id de la lista de productos a gestionar

Buscar Descripcion

Descripcion:

LAPTOP

datosProducto{id=22, precioU=580.0\$, stock=3 unidades , descripcion=LAPTOP, iva=false, NO TIENE

-Menú Factura

Gestión de factura

1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir

1

Ingrese los datos de la factura:

total a pagar

580

Ingrese el Dia de creacion de la factura

08

Ingrese el Mes de creacion de la factura

06

Ingrese el Año de creacion de la factura

2021

Producto creado: true

Gestión de factura

1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir

5

Listado de facturas:

factura{id=1, totalPagar=580.0, fecha de la factura=6/8/2021 DatosCliente=DatosCl

te=DatosCliente{id=1, nombre=Andres, cedula=0107300469, fiabPago=500 \$ }, DatosProducto=datosProducto{i

, DatosProducto=datosProducto{id=22, precioU=580.0\$, stock=3 unidades , descripcion=LAPTOP, iva=false, NO TIENE IVA SU VALOR SIN

Sigue guardando los datos, aunque se salga del menú.

Gestión de factura

1. Crear
2. Actualizar
3. Buscar
4. Eliminar
5. Listar
6. Salir

6

Seleccione una opción

1. Datos Cliente
2. Datos Producto
3. Factura
4. Salir

3

Seleccione el cliente para gestionar sus facturas

Buscar Cedula

Cedula:

0107300469

DatosCliente{id=1, nombre=Andres, cedula=0107300469, fiabPago=500 \$ }

Seleccione el id de la lista de productos a gestionar

Buscar Descripcion

Descripcion:

LAPTOP

datosProducto{id=22, precioU=580.0\$, stock=3 unidades , descripcion=LAPTOP, iva=f

~ ~ ~ ~ ~ ~ ~ ~ ~ ~

Descripcion:

LAPTOP

datosProducto{id=22, precioU=580.0\$, stock=3 unidades , descripcion=LAPTOP, iva=false, NO TIENE IVA SU VA

Gestión de factura

1. Crear
 2. Actualizar
 3. Buscar
 4. Eliminar
 5. Listar
 6. Salir
- 5

Listado de facturas:

factura{id=1, totalPagar=580.0, fecha de la factura=6/8/2021 DatosCliente=DatosCliente{id=1, nombre=Andres,

Gestión de factura

1. Crear