Manual Técnico de Proyecto de Programación II

José Pablo Gutiérrez Medina

Facultad de Ingenierías Universidad Latina

BIS04: Programación II

Johan Figueroa Guevara

22 de agosto de 2022

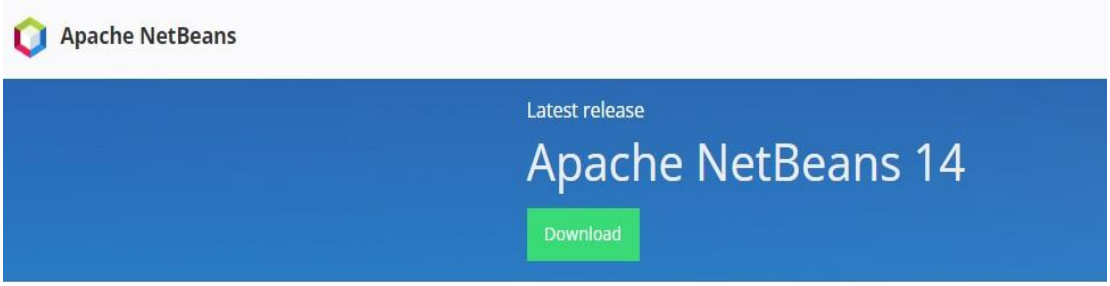
Índice

Introducción

El presente manual técnico tiene como propósito informar al lector sobre los requerimientos básicos que necesita el programa para poder ser ejecutado de manera correcta. También se tiene como objetivo que el lector pueda tener un vistazo a la perspectiva del proyecto desde el punto de vista del creador del programa como tal, puesto que cabe resaltar que para la resolución del problema que se plantea de manera escrita en el proyecto, hay muchas maneras de las cuales se pueden llegar a solución como tal del mismo.

Requerimientos Básicos para la Ejecución del Programa

Descargar e instalar Apache NetBeans.



Descargar e instalar Xampp.

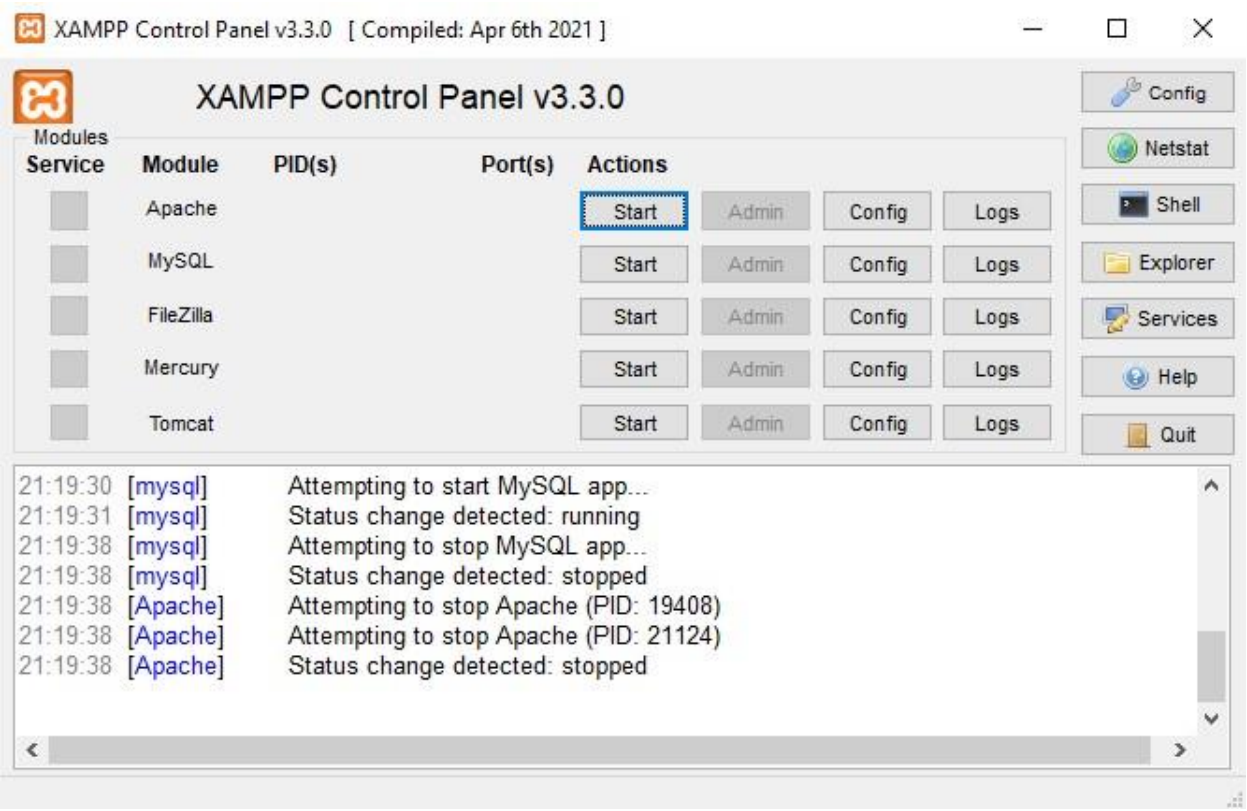
Descargar e instalar My SQL Connector.

mysql-connector-java-5.1.46.jar

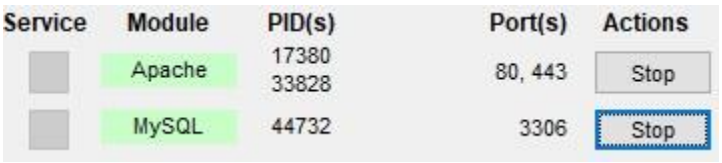
Guía Básica

Una vez instalados los programas básicos para poder ejecutar el programa, lo siguiente es saber cómo configurarlos.

Para empezar, Xampp debe ser configurado para que MySQL y NetBeans funcionen de manera conjunta, por lo tanto al haber instalado el Xampp, lo primero que debemos al llegar a su pantalla principal como vemos en la imagen a continuación, es lo siguiente:

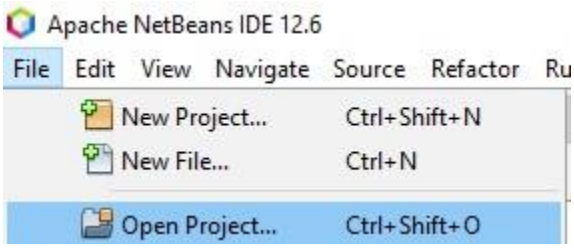


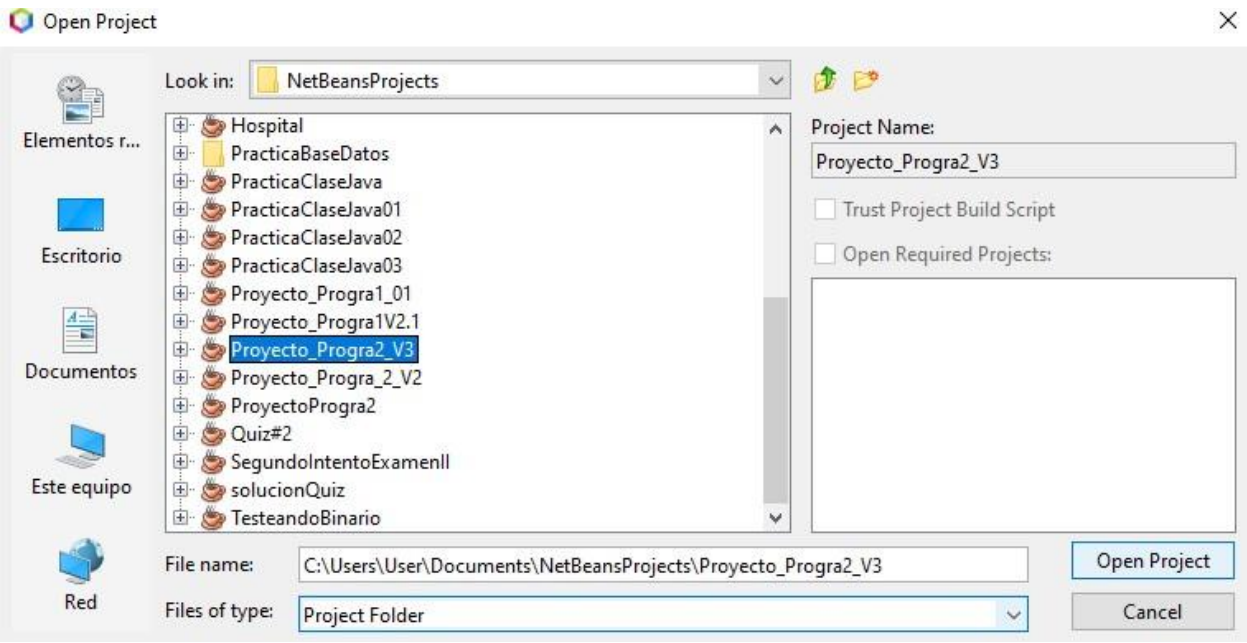
Una vez que hemos llegado acá, lo que se debe hacer es presionar Start tanto en la fila que corresponde a Apache y MySQL. Una vez hecho eso, la columna de Module se pondrá en color verde indicando que estamos listos para el siguiente paso.



Ahora podemos ingresar a NetBeans puesto que el programa ya está configurado para extraer información desde la base de datos en phpMyAdmin, sin embargo en caso de no tener la base de datos en la computadora en la que se va a ejecutar el programa, deberíamos de crearla primero o sino también exportarla por medio de un scritp. Sin embargo en el presente manual no se explicará a detalle.

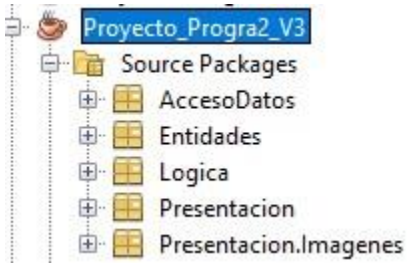
Una vez descomprimido el archivo que contiene el proyecto debemos abrir NetBeans y abrir el proyecto llamado Proyecto_Progra2_V3.



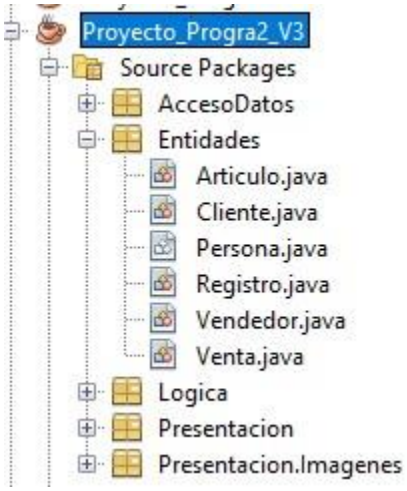


Vista General de Entidades, Paquetes y Clases

Una vez abierto el programa, se podrá notar que todas las clases que corresponden a la definición del proyecto como tal y otras clases que el creador consideró necesarias para el óptimo funcionamiento del mismo, están divididas por paquetes para poder de esa manera dividir las diferentes capas que conforman el programa y además para llevar un orden adecuado del mismo.



Para poder entender el problema que plantea el proyecto debemos ir de manera ordenada, para ello lo primero que debemos hacer es entonces definir las entidades que compondrán el programa.



Entidades, Artículo

Una de las entidades que componen la Venta. Cuenta con tres atributos, dos constructores y sus respectivos encapsuladores.

```java
package Entidades;

public class Articulo {

    private String marca;
    private String descripcion;
    private Double precio;

    public Articulo() {
        this.marca = "";
        this.descripcion = "";
        this.precio = 0.0;
    }

    public Articulo(String marca, String descripcion, Double precio) {
        this.marca = marca;
        this.descripcion = descripcion;
        this.precio = precio;
    }

    public String getMarca() {...3 lines }

    public void setMarca(String marca) {...3 lines }

    public String getDescripcion() {...3 lines }

    public void setDescripcion(String descripcion) {...3 lines }

    public Double getPrecio() {...3 lines }

    public void setPrecio(Double precio) {...3 lines }

}
```

Entidades, Persona

Entidad de la cual se supone que no deberíamos crear instancias, es por esta razón que el creador del proyecto la definió como abstracta. En dado caso que se necesite instanciar un objeto Persona, se puede hacer de manera tal que se comporte como una de sus clases hijas sin que haya ningún conflicto. Cuenta con dos atributos, dos constructores y sus respectivos encapsuladores.

```
package Entidades;

public abstract class Persona {

    private String cedula;
    private String nombre;

    public Persona() {
        this.cedula = "";
        this.nombre = "";
    }

    public Persona(String cedula, String nombre) {
        this.cedula = cedula;
        this.nombre = nombre;
    }

    public String getCedula() {...3 lines }

    public void setCedula(String cedula) {...3 lines }

    public String getNombre() {...3 lines }

    public void setNombre(String nombre) {...3 lines }

}
```

Entidades, Cliente

Una de las clases hijas de Persona y a su vez es parte de la Venta. Cuenta con un atributo propio y con dos atributos heredados de Persona, dos constructores y sus respectivos encapsuladores.

```
package Entidades;

public class Cliente extends Persona{

    private String codigoCliente;

    public Cliente (){
        this.codigoCliente = "";
    }

    public Cliente(String cedula, String nombre, String codigoCliente) {
        super(cedula, nombre);
        this.codigoCliente = codigoCliente;
    }

    public String getCodigoCliente() {...3 lines }

    public void setCodigoCliente(String codigoCliente) {...3 lines }


}
```

Entidades, Vendedor

Otra clase heredada de Persona, también parte de la Venta. Cuenta con un atributo propio y con dos atributos heredados de Persona, dos constructores y sus respectivos encapsuladores.

```
package Entidades;

public class Vendedor extends Persona {

    private String codigoVendedor;

    public Vendedor() {
        this.codigoVendedor = "";
    }

    public Vendedor(String cedula, String nombre, String codigoVendedor) {
        super(cedula, nombre);
        this.codigoVendedor = codigoVendedor;
    }

    public String getCodigoVendedor() {...3 lines }

    public void setCodigoVendedor(String codigoVendedor) {...3 lines }

}
```

Entidades, Venta

Entidad de la cual se basa toda la definición del proyecto y por tanto una de las más importante, al tratarse de una tienda ficticia tratando de hacer ventas como tal. Cuenta con tres atributos que ya conocimos anteriormente que son: Cliente, Vendedor y Artículo. Al igual que dos constructores y sus respectivos encapsuladores.

```
package Entidades;

public class Venta {

    private Cliente cliente;
    private Vendedor vendedor;
    private Articulo articulo;

    public Venta() {
        this.cliente = new Cliente();
        this.vendedor = new Vendedor();
        this.articulo = new Articulo();
    }

    public Venta(Cliente cliente, Vendedor vendedor, Articulo articulo) {
        this.cliente = cliente;
        this.vendedor = vendedor;
        this.articulo = articulo;
    }

    public Cliente getCliente() {...3 lines }

    public void setCliente(Cliente cliente) {...3 lines }

    public Vendedor getVendedor() {...3 lines }

    public void setVendedor(Vendedor vendedor) {...3 lines }

    public Articulo getArticulo() {...3 lines }

    public void setArticulo(Articulo articulo) {...3 lines }

}
```

Entidades, Registro

Esta entidad es necesaria porque es la que se comunicará con las demás capas. Registro será lo que se enviará en muchos métodos por parámetros o lo que devolverán los métodos para que pueda existir la comunicación entre los distintos niveles de capas.

Cuenta con un único atributo que será Venta, dos constructores y sus respectivos encapsuladores.

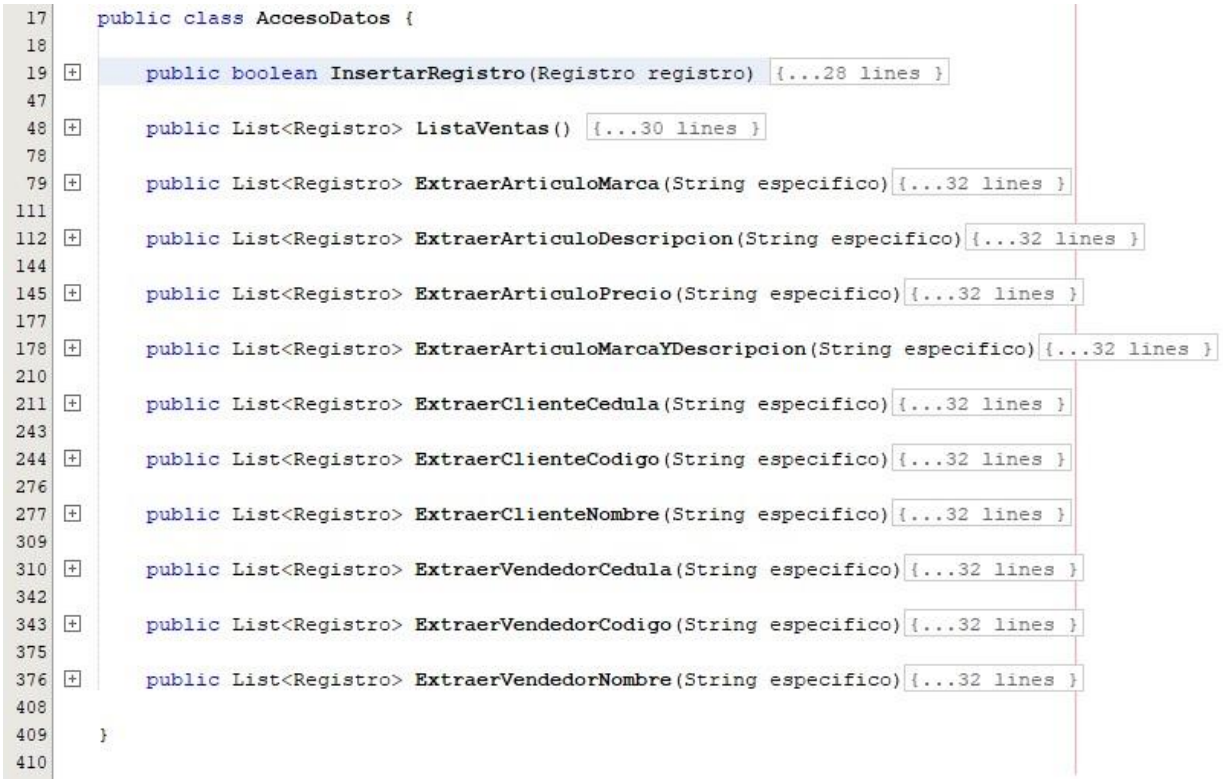
```
package Entidades;

public class Registro {

    private Venta venta;

    public Registro() {
        this.venta = new Venta();
    }

    public Registro(Venta venta) {
        this.venta = venta;
    }

    public Venta getVenta() {...3 lines }

    public void setVenta(Venta venta) {...3 lines }

}
```

Ahora que se ha visto cada una de las Entidades que conforman el programa, lo siguiente será conocer las distintas capas que conforman el proyecto, empezando desde la más baja hasta llegar a la capa más cercana al usuario que sería la capa de Presentación.

Este manual técnico no entrará en detalle con cada uno de los métodos que conforman las capas que veremos a continuación dado que contienen muchas líneas de código, por lo tanto, se dará un breve repaso de su funcionamiento ya que se explicará con lujo de detalle su funcionamiento el día que el creador del programa haga su exposición en clase.

Acceso a Datos

Esta clase es la que se encarga de comunicarse con la base de datos, por lo tanto cuenta con métodos que crean instancias de las entidades que se conocieron anteriormente para poder instanciar un Registro de manera que este pueda ser enviado a la capa superior que en este caso es Lógica y a la base de datos. Cabe destacar que esta clase solamente podrá comunicarse con la capa Lógica y con la base de datos, nunca deberá comunicarse con Presentación.



En la imagen podemos observar algunos de los métodos que componen la capa de AccesoDatos.

Lógica

La capa de Lógica es la encargada de comunicarse con la de Presentación y con Acceso Datos. También como se podrá ver en una imagen cuenta con una clase llamada Estadística que es la encargada de realizar los cálculos necesarios para el promedio y suma total de ventas.

```
1   package Logica;
2
3   import AccesoDatos.AccesoDatos;
4   import Entidades.Registro;
5   import java.util.List;
6   import javax.swing.JOptionPane;
7
8   public class Logica {
9
10      public boolean Validar(Registro registro) {...9 lines }
19
20      public boolean Numerico(String string) {...8 lines }
28
29      public List<Registro> ListaVentas() {...5 lines }
34
35      public List<Registro> ExtraerArticuloMarca(String especifico) {...4 lines }
39
40      public List<Registro> ExtraerArticuloDescripcion(String especifico) {...4 lines }
44
45      public List<Registro> ExtraerArticuloPrecio(String especifico) {...4 lines }
49
50      public List<Registro> ExtraerCedulaCliente(String especifico) {...4 lines }
54
55      public List<Registro> ExtraerCodigoCliente(String especifico) {...4 lines }
59
60      public List<Registro> ExtraerNombreCliente(String especifico) {...4 lines }
64
65      public List<Registro> ExtraerCedulaVendedor(String especifico) {...4 lines }
69
70      public List<Registro> ExtraerCodigoVendedor(String especifico) {...4 lines }
74
75      public List<Registro> ExtraerNombreVendedor(String especifico) {...4 lines }
79
80  }
```

```
package Logica;

import Entidades.Registro;
import java.util.List;
import javax.swing.JOptionPane;

public class Estadistica {

    public Double PromedioVentas(List<Registro> lista) {...15 lines }

    public Double SumaTotal(List<Registro> lista) {...11 lines }

}
```

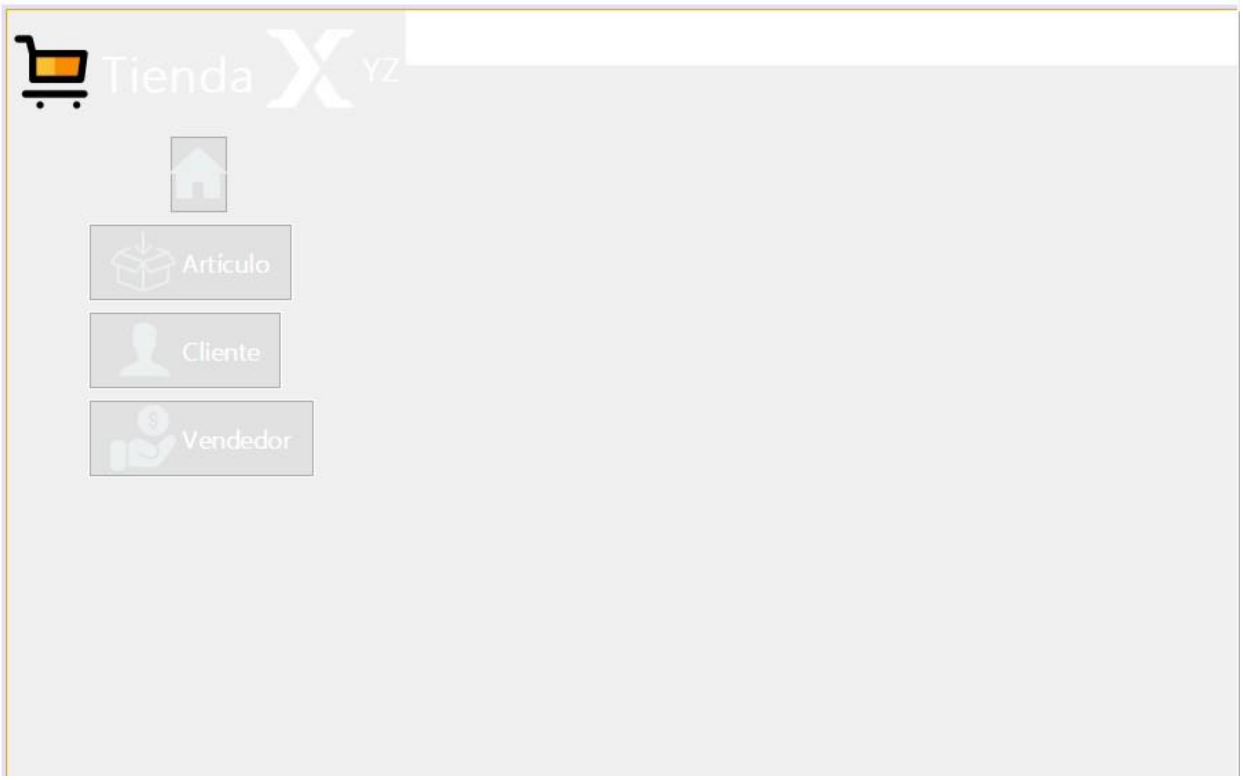
Presentación

La capa de Presentación cuenta con varios Frames que el creador consideró necesarios para darle un mayor orden al programa. Además al ser la parte que interactúa con el usuario fue muy importante dividir cada uno de esos Frames para que la interfaz del usuario fuera más amigable y más agradable para el usuario. También al ser una clase tan grande y que contiene tantas líneas de código se tratará de exponer al lector a esta capa con la mayor brevedad, sin embargo, como se mencionó en la capa de Acceso Datos se verá con mucho más detalle y más explicado el día de la exposición en clase.

Métodos de la capa Presentación:

```
public class Presentacion extends javax.swing.JFrame {

    public Presentacion() {...14 lines }

    class JPanelBlue extends JPanel {...15 lines }

    public void Transparencia(){...14 lines }

    @SuppressWarnings("unchecked")
    Generated Code

    private void btnHomeActionPerformed(java.awt.event.ActionEvent evt) {...10 lines }

    private void btnArticuloActionPerformed(java.awt.event.ActionEvent evt) {...11 lines }

    private void btnClienteActionPerformed(java.awt.event.ActionEvent evt) {...10 lines }

    private void btnVendedorActionPerformed(java.awt.event.ActionEvent evt) {...11 lines }

    public static void main(String args[]) {...31 lines }

    // Variables declaration - do not modify
    private javax.swing.JButton btnArticulo;
    private javax.swing.JButton btnCliente;
    private javax.swing.JButton btnHome;
    private javax.swing.JButton btnVendedor;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel pnlContenedor;
    // End of variables declaration
}
```

Diseño de Presentación :



Home

El Frame Home es básicamente un JPanel que se encuentra en Presentación, es por eso que es prácticamente la primera vista que ve el usuario y con la que va a interactuar primero para poder realizar una venta y ver el listado de ventas encontrado en la base de datos.

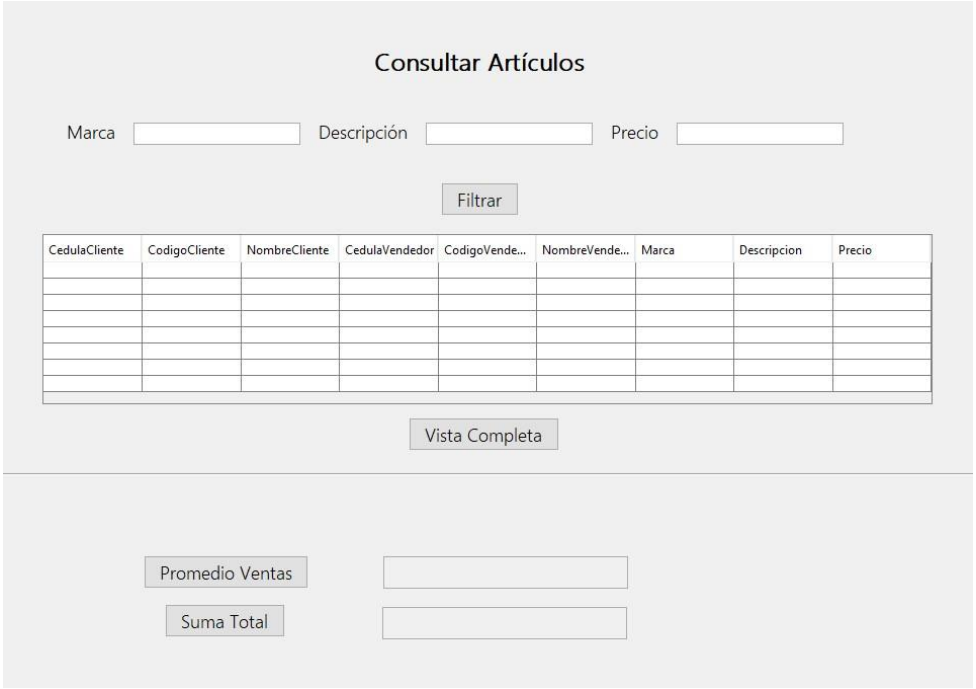
```
public class Home extends javax.swing.JPanel {

    public Home() {...3 lines }

    class JPanelGray extends JPanel {...15 lines }

    private TableModel CompletarTableModel(){...27 lines }

    private void RefrescarTabla() {...4 lines }

    @SuppressWarnings("unchecked")
    Generated Code

    private void txtCedulaClienteActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

    private void txtCedulaVendedorActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

    private void btnGuardarVentaActionPerformed(java.awt.event.ActionEvent evt) {...38 lines }

    private void btnActualizarTablaActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }
```

Diseño de Home.

Artículo

El Frame Artículo (no confundir con la entidad Artículo) es básicamente el que se encarga de filtrar los artículos de la base de datos por Marca, Descripción o Precio. Además es acá donde se podrá realizar el cálculo de promedio y de suma total de artículos.

```
public class ArticuloFrame extends javax.swing.JPanel {

    public ArticuloFrame() {...5 lines }

    class JPanelGray extends JPanel {...15 lines }

    private TableModel CompletarTableModel() {...27 lines }

    private TableModel MostrarListaMarca() {...27 lines }

    private TableModel MostrarListaDescripcion() {...27 lines }

    private TableModel MostrarListaPrecio() {...27 lines }

    private void RefrescarTabla() {...4 lines }

    private void ListaMarca() {...4 lines }

    private void ListaDescripcion() {...4 lines }

    private void ListaPrecio() {...4 lines }

    @SuppressWarnings("unchecked")
    Generated Code

    private void btnPromedioVentasActionPerformed(java.awt.event.ActionEvent evt) {...5 lines }

    private void btnSumaTotalActionPerformed(java.awt.event.ActionEvent evt) {...5 lines }

    private void btnVistaCompletaActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

    private void btnFiltrarActionPerformed(java.awt.event.ActionEvent evt) {...18 lines }
```



Cliente

El Frame Cliente es el que se encarga de filtrar Clientes por cédula, código o nombre.

```
public class ClienteFrame extends javax.swing.JPanel {

    public ClienteFrame() {...4 lines }

    class JPanelGray extends JPanel {...15 lines }

    private TableModel CompletarTableModel() {...27 lines }

    private TableModel MostrarListaCedula() {...27 lines }

    private TableModel MostrarListaCodigo() {...27 lines }

    private TableModel MostrarListaNombre() {...27 lines }

    private void RefrescarTabla() {...4 lines }

    private void ListaCedula() {...4 lines }

    private void ListaCodigo() {...4 lines }

    private void ListaNombre() {...4 lines }
```

Vendedor

El Frame Vendedor al igual que el de cliente se encarga de filtrar Vendedores por cédula, código o nombre.

```
public class VendedorFrame extends javax.swing.JPanel {

    public VendedorFrame() {...4 lines }

    class JPanelGray extends JPanel {...15 lines }

    private TableModel CompletarTableModel() {...27 lines }

    private TableModel MostrarListaCedula() {...27 lines }

    private TableModel MostrarListaCodigo() {...27 lines }

    private TableModel MostrarListaNombre() {...27 lines }

    private void RefrescarTabla() {...4 lines }

    private void ListaCedula() {...4 lines }

    private void ListaCodigo() {...4 lines }

    private void ListaNombre() {...4 lines }
```