Manual Técnico

1. Método Main

Se inicializa la ventana de Inicio de Sesión

```
public class SancarlistaShop {

    public static void main(String[] args) {

   Módulo_Autenticación M1=new Módulo_Autenticación("Inicio de Sesión");
     M1.setVisible(true);//Se hace visible la ventana de sesión

    }
}
```

2. AbrirCerrar

Valida los datos de inicio de sesión del administrador

```
public void AbrirCerrrar(String Opcion, String letra, String letra2) {

    switch (Opcion) {
        case "a" -> {
            System.out.println("-----------------------------------------------------------");
            System.out.println("El valor del código es:" + letra + "y contraseña:" + letra2);
            System.out.println("-----------------------------------------------------------");

            Controlador llave = new Controlador();
            boolean respuesta2 = llave.Iniciar(letra, letra2); //Función que verifica si la lla
            System.out.println("El valor de respueta2:" + respuesta2);
            System.out.println("Se ejecuto el método");

            if (p == null) {
                p = new Módulo_Administración("Módulo_Administración");
            }
            p.setVisible(respuesta2);
            Módulo_Autenticación.visibilidad.setVisible(!respuesta2);

        }
```

3. **Iniciar**

   Valida los datos de Inicio de sesión para los vendedores

```
public boolean Iniciar(String codigo, String contraseña) {

    if (codigo.equals("admin") && contraseña.equals("IPC1<C>")) {
        System.out.println("Código y Contraseña Correctos, se encontro en el sistema");
        return true;
    }
    System.out.println("Código ó Contraseña Incorrectos, no se encontro en el sistema");
                        JOptionPane.showMessageDialog(null, "No se encontro admin ");

    return false;
}

///Método para volver en cualquier página a Inciar Seció
public void Casa(JFrame a) {
    Módulo_Autenticación.visibilidad.setVisible(true);
    a.setVisible(false);
}
```

4. **Casa**

   Permite al sistema regresar al la ventana de inicio de sesión

```
///Método para volver en cualquier página a Inciar Seció
public void Casa(JFrame a) {
    Módulo_Autenticación.visibilidad.setVisible(true);
    a.setVisible(false);
}
```

5. **CargarArchivo**

Función que permite cargar los archivos csv

```
public void cargarArchivo(File csv) throws FileNotFoundException {
    FileReader fr = null;
    BufferedReader br = null;

    try {
        fr = new FileReader(csv);
        br = new BufferedReader(fr);

        //Contar cuantas lineas tiene el archivo
        int numLineas = 0;

        while (br.readLine() != null) {
            numLineas++;
            AA = numLineas;
        }
        // Reiniciar el lector para volver al inicio del archivo
        fr.close();
        br.close();

        fr = new FileReader(csv);
        br = new BufferedReader(fr);
        NN = 100;

        //Leer y guardar datos
        String linea;
        int i = 0;//la clave
        Productos V6 = new Productos();

        hh = V6.PrimeroCrear(i);
        BB = hh;
```

```
            System.out.println("Archivo cargado correctamente con " + NN + " registros.");

            Productos a = new Productos();
            a.CrearTablaF();
            a.PrimeroCargar(numLineas);
            //a.NoIgual();
            a.ReyenarTablaCargar(NN);

        } catch (Exception ex) {
            System.out.println("Error al leer el archivo: " + ex.getMessage());
        } finally {//Cerrar archivo
            try {
                if (fr != null) {
                    fr.close();
                }
                if (br != null) {
                    br.close();
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }

        }
    }
```

6. EditorBoton

Es parte de la interfaz para la edición del boton en la tabla

```
public EditorBoton(JTable table) {
    super(new JCheckBox()); // solo se usa para cumplir con DefaultCellEditor
    this.table = table;
    editarBoton = new JButton();
    editarBoton.setFont(new Font("StialHati-Regular", Font.CENTER_BASELINE, 10));
    //editarBoton.setBackground(new Color(255, 255, 255));
    editarBoton.setForeground(new Color(0, 102, 204));
    editarBoton.setOpaque(true);

    editarBoton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            fireEditingStopped(); //Permite no queda en modo edición
        }
    });
}
```

Funciones de Controlador para todos los módulos

7. setcodigo

Guarda el codigo y realiza las operaciones según sea el modulo, para luego guardarlo en la matriz correspondiente

```
    public void setCodigo(String codigo) {
        this.codigo = codigo;

        //Validaciones
        switch (codigo) {
            case "" -> {
                System.out.println("El código no puede ser vacio");
                break;

            }
    default -> {
        //Se agrega a la matriz
        UsuarioP[contadorP][0] = codigo;
        contadorP++;
        System.out.println(codigo);

        for (int i = 0; i < (contadorP - 1); i++) {
            if (Objects.equals(UsuarioP[i][0], UsuarioP[contadorP - 1][0])) {
                UsuarioP[contadorP - 1][0] = null;
                contadorP = contadorP - 1;
                System.out.println("El código ya existe");
                break;
            } else {
                System.out.println("Codigo guardado en matriz: " + UsuarioP[contadorP - 1][0] + "Comprobación: " + i);
                //break;
            }
        }
    }
}
System.out.println("ContadorP: " + contadorP);
```

8. setNombre

Guarda el nombre y realiza las operaciones según sea el modulo, para luego guardarlo en la matriz correspondiente

```
public void setNombre(String nombre) {
    this.nombre = nombre;
    //Validaciones
    switch (nombre) {
        case "" -> {
            System.out.println("El nombre no puede ser vacio");
            break;

        }
        default -> {
            //Se agrega a la matriz
            UsuarioP[contador2P][1] = nombre;
            contador2P++;
            System.out.println(UsuarioP[contador2P - 1][1]);
        }
    }
    System.out.println("Contador2P: " + contador2P);
}
```

9. setCategoria

Guarda la categoría y realiza las operaciones según sea el modulo, para luego guardarlo en la matriz correspondiente

```
public String setCategoria(String categoria) {
    this.categoria = categoria;
    //Validaciones
    switch (categoria) {
        case "Tecnología", "tecnología" -> {
            UsuarioP[contador3P][2] = "Tecnología";
            contador3P++;
            System.out.println(UsuarioP[contador3P - 1][2]);

        }
        case "Alimento", "alimento" -> {
            //Se agrega a la matriz
            UsuarioP[contador3P][2] = "Alimento";
            contador3P++;
            System.out.println(UsuarioP[contador3P - 1][2]);
        }
        case "General", "general" -> {
            //Se agrega a la matriz
            UsuarioP[contador3P][2] = "General";
            contador3P++;
            System.out.println(UsuarioP[contador3P - 1][2]);
        }
    }
```

10. setAtributo

Guarda el atributo y realiza las operaciones según sea el modulo, para luego guardarlo en la matriz correspondiente (en muchos módulos se sobreescribe)

```
public void setAtributo(String atributo) {//Mensaje que se va ha mostrar
    this.atributo = atributo;
    switch (atributo) {
        case "" -> {
            System.out.println("La contraseña no puede estar vacia");
            break;

        }
        default -> {
            //Se agrega a la matriz
            UsuarioP[contador4P][3] = atributo;
            contador4P++;
            System.out.println(UsuarioP[contador4P - 1][3]);
        }
    }
    System.out.println("Contador4P: " + contador4P);
}
```

11. setConfirmadas

Guarda el dato numérico de la tabla y realiza las operaciones según sea el modulo, para luego guardarlo en la matriz correspondiente (en muchos módulos se sobreescribe)

```
public void setConfirmados(String confirmados) {
    this.confirmados = confirmados;
    String con = String.valueOf(confirmados);
    System.out.println("La categoría es: " + con);

    switch (con) {
        case "Tecnología", "tecnología" -> {
            UsuarioP[contador5P][4] = "Ver detalle T";
            contador5P++;
            System.out.println(UsuarioP[contador5P - 1][2]);
```

```
            case "Alimento", "alimento" -> {
                //Se agrega a la matriz
                UsuarioP[contador5P][4] = "Ver detalle A ";
                contador5P++;
                System.out.println(UsuarioP[contador5P - 1][2]);
            }
            case "General", "general" -> {
                //Se agrega a la matriz
                UsuarioP[contador5P][4] = "Ver detalle G";
                contador5P++;
                System.out.println(UsuarioP[contador5P - 1][2]);
            }
        }
```

12. Crear Tabla

Crea la tabla con su respectivo cambio en sus atributos

```
public void CrearTablaF() { //Crear tabla con formato
    //Validar que la tabla no exista

    if (tms == null) {
        System.out.println("La tabla no existe, creando tabla");
        DefaultTableModel MD = new DefaultTableModel(new String[]{"Código", "Nombre", "Categoría", "Acciones"}, 101);

        Módulo_Administración.tabla2.setModel(MD);
        Módulo_Administración.tabla2.getColumnModel().getColumn(3).setCellRenderer(new GestionBoton());
        Módulo_Administración.tabla2.getColumnModel().getColumn(3).setCellEditor(new EditorBoton(Módulo_Administración.tabla2));
        tms = Módulo_Administración.tabla2.getModel();
    } else {
        System.out.println("La tabla ya existe, se continua para rellenar");
    }
}
```