



UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE  
SEDE SANTO DOMINGO DE LOS TSÁCHILAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN



PERIODO : 2024 51 mayo– septiembre 2024  
ASIGNATURA : Sistemas Operativos  
TEMA : Examen U3  
ESTUDIANTE : Jiménez Juan  
NIVEL-PARALELO - NRC: NRC 15310  
DOCENTE : Ing. Javier Cevallos  
FECHA DE ENTREGA : 31/08/2024

SANTO DOMINGO – ECUADOR

<b>1. Introducción</b>	<b>3</b>
<b>2. Objetivos</b>	<b>3</b>
Objetivo General:	3
Objetivos Específicos:	3
<b>3. Desarrollo / Marco Teórico/ Práctica</b>	<b>4</b>
Marco teórico:	4
Marco Practico:	5
<b>4. Conclusiones</b>	<b>29</b>
<b>5. Recomendaciones</b>	<b>29</b>
<b>6. Bibliografía/ Referencias</b>	<b>30</b>
<b>7. Anexos:</b>	<b>30</b>

## **1. Introducción**

En el presente informe se revisará la elaboración de la actividad para evaluación del examen de la 3ra unidad de la materia de Sistemas Operativos (SO), donde se realizó una interfaz gráfica para la ejecución de scripts mediante el uso de botones dentro del mismo programa esto desarrollado tanto para el SO windows y distribuciones de linux (Debian).

Un terminal es un dispositivo de hardware, ya sea de naturaleza electromecánica o electrónica, que se puede usar tanto para ingresar como para transcribir información. Esas tareas se pueden llevar a cabo bien desde un ordenador o echando mano de un sistema informático.  
(Desarrollo, 2022)

## **2. Objetivos**

### ***Objetivo General:***

- Realizar un programa que cree una interfaz gráfica con botones que ejecuten scripts tanto como windows y en Debian.

### ***Objetivos Específicos:***

- Realizar una interfaz gráfica con el lenguaje de programación java para realizar una ventana que ejecute mediante botones los scripts.
- Realizar la interfaz gráfica uno para el SO windows y otra interfaz gráfica para Debian que realicen la misma función.

### **3. Desarrollo / Marco Teórico/ Práctica**

#### ***Marco teórico:***

#### ***KERNEL:***

El componente central de un sistema operativo, también conocido como núcleo (kernel), sirve como intermediario entre el hardware y el software, lo que permite que interactúen entre sí para que el sistema funcione correctamente. Para comprender su funcionamiento y maximizar su eficiencia y seguridad, es esencial estudiar sus capacidades de hardware soportadas y la programación a nivel de sistema.

#### ***Características y componentes de Kernel***

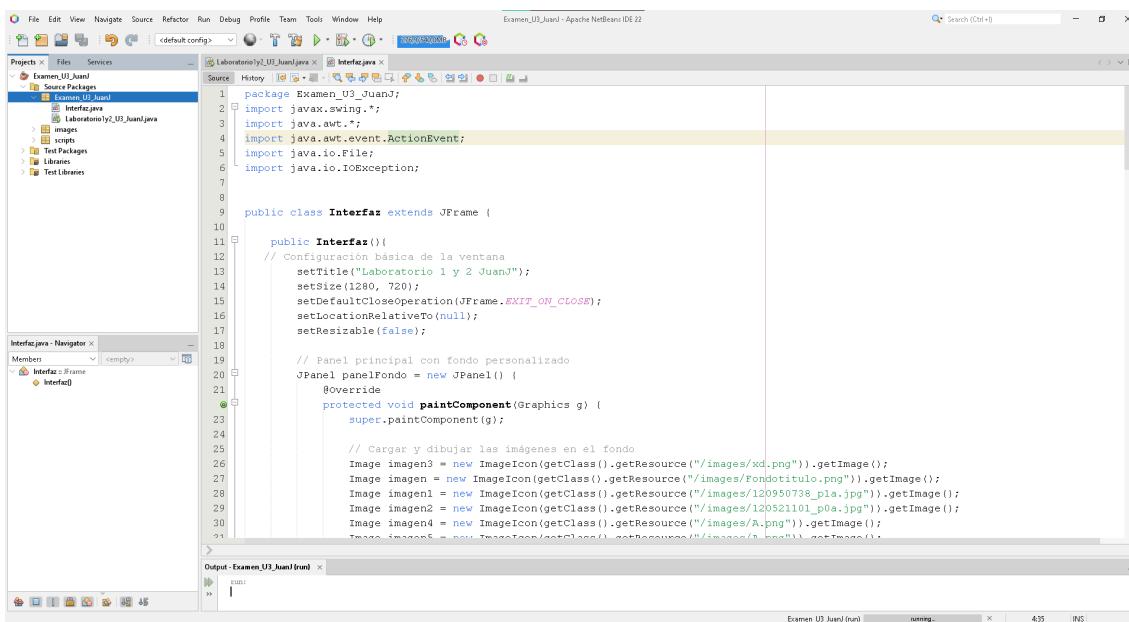
- El Kernel Realiza una variedad de funciones que son esenciales para el funcionamiento del sistema operativo:
- Gestión de procesos: El Kernel controla la ejecución de los procesos del sistema, asigna tiempo de CPU, controla la concurrencia y coordina la comunicación entre procesos.
- Gestión de memoria: es responsable de asignar y liberar memoria a los procesos, garantizando un uso efectivo de los recursos y evitando conflictos de acceso a la memoria.
- Gestión de E/S (Entrada/Salida): Mantiene la comunicación fluida y sincronizada entre los dispositivos de E/S y el resto del sistema.
- Sistema de Archivos: Permite la lectura, escritura y manipulación de datos almacenados en disco mediante el acceso a sistemas de archivos.

## **Capacidades de Hardware Soportadas:**

El kernel de un sistema operativo especifica los tipos de hardware que puede administrar y cómo interactuar con ellos. Por ejemplo, los sistemas operativos modernos deben poder reconocer y usar una amplia gama de tarjetas de red, controladores de dispositivos, dispositivos de almacenamiento y otros componentes de hardware. “Investigar las características y funcionalidades del kernel de varios sistemas operativos, incluida su arquitectura y mecanismos de comunicación con el hardware, es esencial para comprender cómo estas capacidades afectan la compatibilidad y el rendimiento del sistema.”"Silberschatz, Galvin y Gagne (año desconocido),.

### **Marco Practico:**

#### **Programación en java código Windows:**



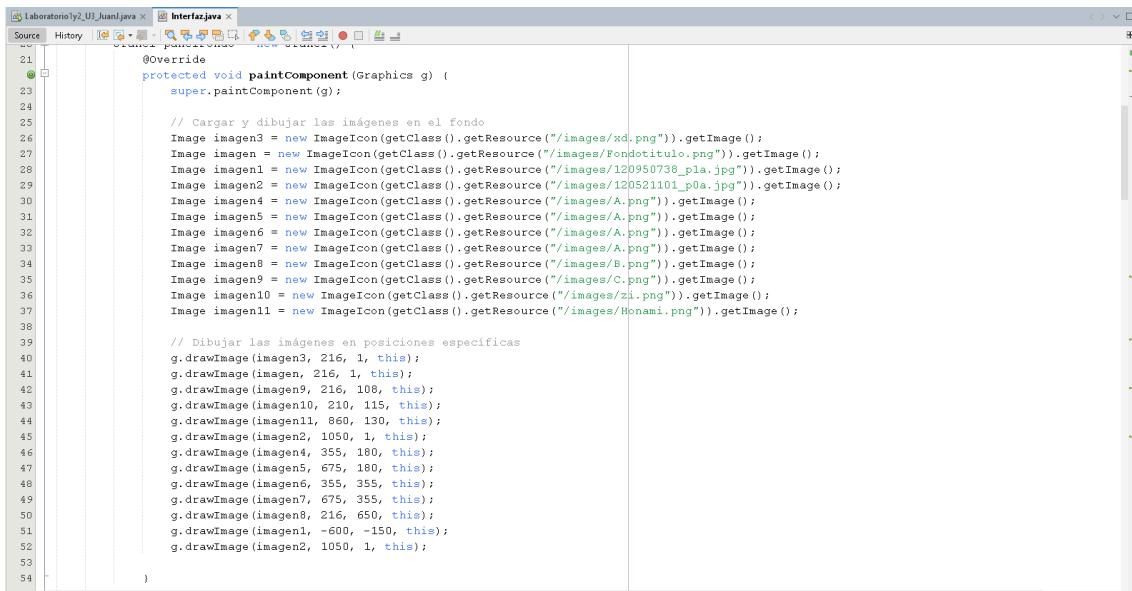
The screenshot shows the Apache NetBeans IDE interface with the following details:

- Project Explorer:** Shows the project "Examen\_U3\_JuanJ" with a package named "Examen\_U3\_JuanJ" containing files "Interfaz.java" and "Laboratorio1\_U3\_Juan.java".
- Code Editor:** Displays the content of "Interfaz.java". The code defines a class "Interfaz" that extends "JFrame". It includes methods for window configuration, a custom panel, and painting logic. It also loads several images from the "images" directory.
- Navigator:** Shows the class "Interfaz" under the "Members" section.
- Output:** Shows the output for "Examen\_U3\_JuanJ" including build logs and run results.

En la imagen se presencia la parte inicial de la clase “Interfaz” que es la encargada de mostrar la ventana con la interfaz gráfica, este empieza con las librerías que son llamadas para el funcionamiento de la interfaz como lo son “la awt” para lo gráfico y la “Swing” parte de JFrame, la de “File” que sirve para la detección de archivos externos de java y la última de

“Exception” que se encarga de la excepciones que pueden dar al momento de ejecutar el programa este no se cuelgue al dar con algún error.

En la imagen también se ve el inicio de la clase interfaz que extiende JFrame que es la principal encargada de las interfaces, siendo esta de tipo pública, dentro se declara la función public Interfaz que es la función principal de la clase donde también se declaran las variables principales de la ventana como lo son el título, el tamaño, sí puede ser configurable, entre otros.



```
21     @Override
22     protected void paintComponent(Graphics g) {
23         super.paintComponent(g);
24
25         // Cargar y dibujar las imágenes en el fondo
26         Image imagen3 = new ImageIcon(getClass().getResource("/images/xd.png")).getImage();
27         Image imagen = new ImageIcon(getClass().getResource("/images/Fondotitulo.png")).getImage();
28         Image imagen1 = new ImageIcon(getClass().getResource("/images/120950738_pla.jpg")).getImage();
29         Image imagen2 = new ImageIcon(getClass().getResource("/images/120521101_p0a.jpg")).getImage();
30         Image imagen4 = new ImageIcon(getClass().getResource("/images/A.png")).getImage();
31         Image imagen5 = new ImageIcon(getClass().getResource("/images/A.png")).getImage();
32         Image imagen6 = new ImageIcon(getClass().getResource("/images/A.png")).getImage();
33         Image imagen7 = new ImageIcon(getClass().getResource("/images/A.png")).getImage();
34         Image imagen8 = new ImageIcon(getClass().getResource("/images/B.png")).getImage();
35         Image imagen9 = new ImageIcon(getClass().getResource("/images/C.png")).getImage();
36         Image imagen10 = new ImageIcon(getClass().getResource("/images/z1.png")).getImage();
37         Image imagen11 = new ImageIcon(getClass().getResource("/images/Honami.png")).getImage();
38
39         // Dibujar las imágenes en posiciones específicas
40         g.drawImage(imagen3, 216, 1, this);
41         g.drawImage(imagen, 216, 1, this);
42         g.drawImage(imagen9, 216, 108, this);
43         g.drawImage(imagen10, 210, 115, this);
44         g.drawImage(imagen11, 860, 130, this);
45         g.drawImage(imagen2, 1050, 1, this);
46         g.drawImage(imagen4, 355, 180, this);
47         g.drawImage(imagen5, 675, 180, this);
48         g.drawImage(imagen6, 355, 355, this);
49         g.drawImage(imagen7, 675, 355, this);
50         g.drawImage(imagen8, 216, 650, this);
51         g.drawImage(imagen1, -600, -150, this);
52         g.drawImage(imagen2, 1050, 1, this);
53     }
54 }
```

Se llama a una librería denominada “JPanel” que se encarga de crear un panel donde se van a colocar las imágenes de fondo, en la imagen se ve cada una de la declaración de las imágenes y las que llaman además de agregar esas al Panel de las imágenes de fondo para que no interfieran con los elementos externos al fondo, aquí también se coloca las posiciones de cada una de las imágenes, entre otros.

```

    panelFondo.setLayout(null);

    // Crear y agregar los 4 botones
    ImageIcon icono = new ImageIcon(getClass().getResource("/images/Almacen-de-datos1.png")); // Cambia la ruta por la tuya
    JButton boton1 = new JButton(icono);
    boton1.setBounds(359, 225, 242, 100);
    ImageIcon icono2 = new ImageIcon(getClass().getResource("/images/segu.png"));
    JButton boton2 = new JButton(icono2);
    boton2.setBounds(679, 225, 242, 100);
    ImageIcon icono3 = new ImageIcon(getClass().getResource("/images/discos.png"));
    JButton boton3 = new JButton(icono3);
    boton3.setBounds(359, 400, 242, 100);
    ImageIcon icono4 = new ImageIcon(getClass().getResource("/images/docker.png"));
    JButton boton4 = new JButton(icono4);
    boton4.setBounds(679, 400, 242, 100);

    // Dandole funciones a los botones
    //boton 1 = boton para gestion de Almacenamiento
    boton1.addActionListener((ActionEvent e) -> {
        try {
            File archivo = new File("C:\\\\Users\\\\CivilJuan\\\\Documents\\\\NetBeansProjects\\\\Laboratorio1y2_U3_JuanJ\\\\Gestion");
            // Verificar si Desktop es compatible con la plataforma
            if (Desktop.isDesktopSupported()) {
                Desktop desktop = Desktop.getDesktop();

                // Verificar si se puede abrir el archivo
                if (archivo.exists()) {
                    desktop.open(archivo); // Abre el archivo con la aplicación predeterminada del sistema
                } else {
                    System.out.println("El archivo no existe.");
                }
            } else {
                System.out.println("Desktop no es compatible en esta plataforma.");
            }
        } catch (IOException j) {
        };
    });

```

En la imagen se aprecia los comandos para declararel panel del imágenes de fondo como null, osea que nos permita mover las imágenes donde queramos ubicarlos en el fondo, además de agregar los botones con el comando JButton y dándoles imagen con la librería ImageIcon que se encarga de leer un archivo de tipo imagen además de darle las posiciones en la que queremos que estén lo botones.

```

    // Dandole funciones a los botones
    //boton 1 = boton para gestion de Almacenamiento
    boton1.addActionListener((ActionEvent e) -> {
        try {
            File archivo = new File("C:\\\\Users\\\\CivilJuan\\\\Documents\\\\NetBeansProjects\\\\Laboratorio1y2_U3_JuanJ\\\\Gestion");
            // Verificar si Desktop es compatible con la plataforma
            if (Desktop.isDesktopSupported()) {
                Desktop desktop = Desktop.getDesktop();

                // Verificar si se puede abrir el archivo
                if (archivo.exists()) {
                    desktop.open(archivo); // Abre el archivo con la aplicación predeterminada del sistema
                } else {
                    System.out.println("El archivo no existe.");
                }
            } else {
                System.out.println("Desktop no es compatible en esta plataforma.");
            }
        } catch (IOException j) {
        };
    });

```

En la imagen se ve el código para la acción de los botones en este caso el comando “addActionListener” que genera una acción al pulsar el botón, para empezar el botón tiene un

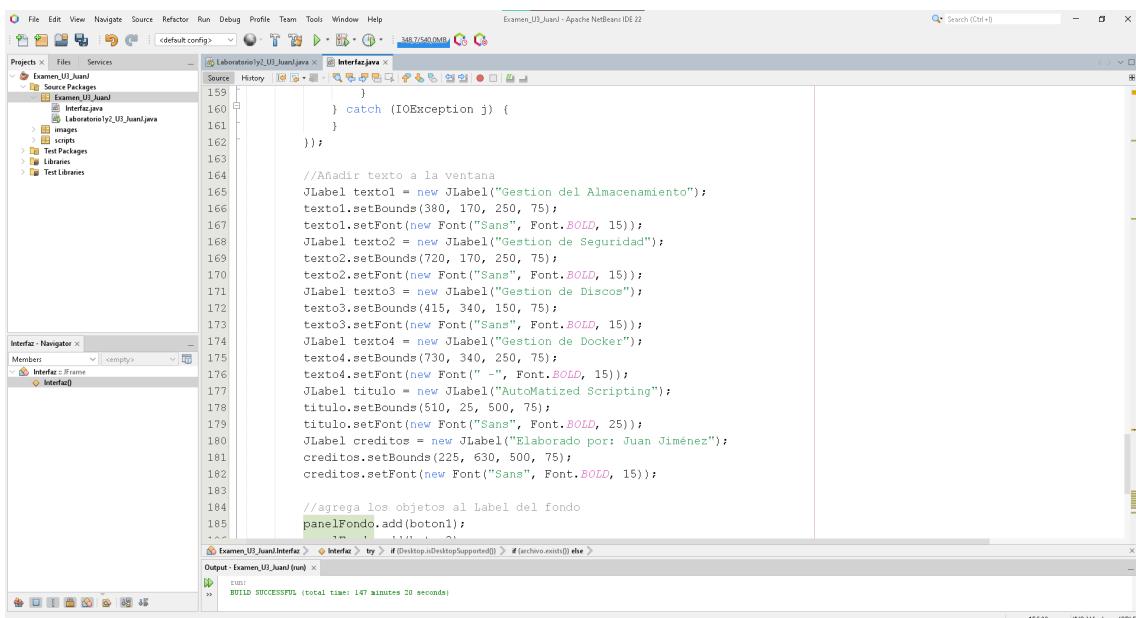
"try" que es para en caso de no funcionar la actividad programada para el botón esté mande un mensaje de error en vez de trabarse el programa.

```

98 //boton 2 = boton para gestion de Seguridad
99 boton2.addActionListener(ActionEvent e) -> {
100     try {
101         File archivo = new File("C:\\Users\\CivilJuan\\Documents\\NetBeansProjects\\Laboratorioly2_U3_JuanJ\\Gestior
102
103
119 //boton 3 = boton para gestion de Discos
120 boton3.addActionListener(ActionEvent e) -> {
121     try {
122         File archivo = new File("C:\\Users\\CivilJuan\\Documents\\NetBeansProjects\\Laboratorioly2_U3_JuanJ\\Gestior
123
124
142 //boton 4 = boton para gestion de Dockers
143 boton4.addActionListener(ActionEvent e) -> {
144     try {
145         File archivo = new File("C:\\Users\\CivilJuan\\Documents\\NetBeansProjects\\Laboratorioly2_U3_JuanJ\\Gestior
146

```

Cabe recalcar que todos los botones realizan la misma acción con el único cambio del cambio de archivo que ejecuta siendo este dependiente del archivo necesario para el botón además de ser estos los únicos que se deben agregar de nuevo la dirección del archivo que se encuentra dentro de la carpeta del proyecto debido a que no logre encontrar la forma de hacer que desde otro paquete este los lea como con las imágenes debido a que al momento de ejecutar la función del botón este se colgaba y congelaba la ventana.



En la imagen se ven el código que se usó para el desarrollo de colocar texto en la pantalla para dar las indicaciones del programa, primero se crea el JLabel y se le da un texto, además de

darles una ubicación, tamaño y largo que puede estar, también se le agregó tipo de letra tamaño y si esta en negrita o normal.

```

File Edit View Navigate Source Refactor Run Debug Profile Tools Window Help
Examen_U3_JuanJ - Apache NetBeans IDE 22
Projects Files Services
Examen_U3_JuanJ
Source Packages
Examen_U3_JuanJ
Source Packages
Interfaz.java
Interfaz.java
178     titulo.setBounds(510, 25, 500, 75);
179     titulo.setFont(new Font("Sans", Font.BOLD, 25));
180     JLabel creditos = new JLabel("Elaborado por: Juan Jiménez");
181     creditos.setBounds(225, 630, 500, 75);
182     creditos.setFont(new Font("Sans", Font.BOLD, 15));
183
184     //agrega los objetos al Label del fondo
185     panelFondo.add(boton1);
186     panelFondo.add(boton2);
187     panelFondo.add(boton3);
188     panelFondo.add(boton4);
189     panelFondo.add(texto1);
190     panelFondo.add(texto2);
191     panelFondo.add(texto3);
192     panelFondo.add(texto4);
193     panelFondo.add(titulo);
194     panelFondo.add(creditos);
195
196     // Agregar el panel al JFrame
197     add(panelFondo);
198 }
199

```

Output - Examen\_U3\_JuanJ (run) > run > BUILD SUCCESSFUL (total time: 147 minutes 20 seconds)

En la parte final de la clase se puede ver como se termina añadiendo todos los botones y JLabels que se crearon a la ventana principal para que estos se agreguen y se logre visualizar en la ventana.

```

File Edit View Navigate Source Refactor Run Debug Profile Tools Window Help
Examen_U3_JuanJ - Apache NetBeans IDE 22
Projects Files Services
Examen_U3_JuanJ
Source Packages
Examen_U3_JuanJ
Source Packages
Examen_U3_JuanJ
Interfaz.java
Interfaz.java
1 package Examen_U3_JuanJ;
2 import javax.swing.*;
3
4 /**
5 * @author CivilJuan
6 */
7 public class Examen_U3_JuanJ {
8
9     public static void main(String[] args) {
10         // TODO code application logic here
11         SwingUtilities.invokeLater(() -> {
12             Interfaz ventana = new Interfaz();
13             ventana.setVisible(true);
14         });
15     }
16 }
17
18
19
20

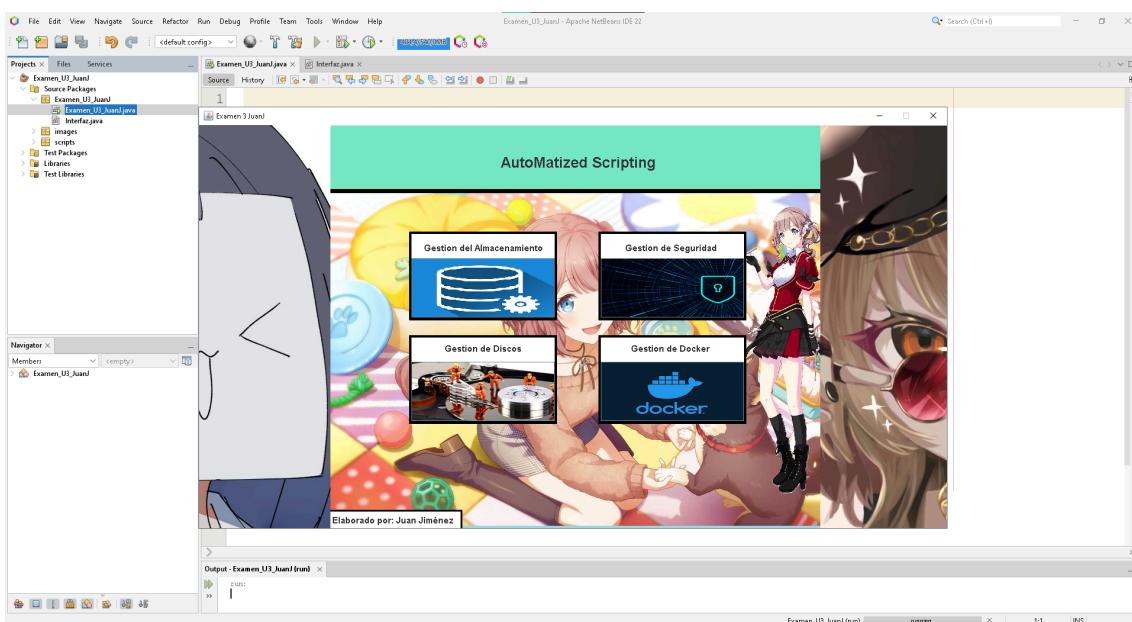
```

Output - Examen\_U3\_JuanJ (run) > run > BUILD SUCCESSFUL (total time: 147 minutes 20 seconds)

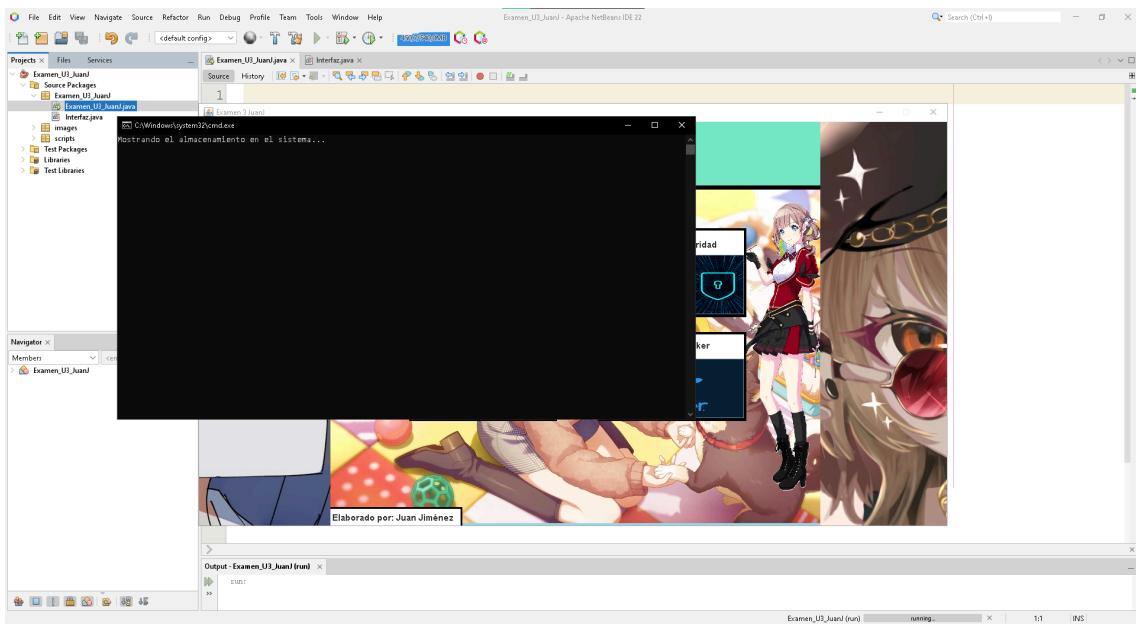
Para finalizar con el código del programa en windows se explicara el código de la clase principal “Examen\_U3\_JuanJ”, este únicamente importa el paquete en el que se encuentra y

importa la librería “Swing” que es para usar las herramientas para la creación de ventanas, seguimos con la declaración de la clase principal para dentro de la misma colocar la función principal o “Main” del programa que se encarga de ejecutar, dentro de este se llama a la librería Swing para usar su código para la creación de ventanas dentro de la función se llama a la clase “Interfaz” que fue explicada anteriormente y se le da una variable de nombre ventana con la cual procede a usar la función “setVisible(true)” para hacer visible la ventana creada.

### **Ejecución de la interfaz Windows:**



Al momento de ejecutar la ventana se mostrará todo lo anteriormente explicado donde las sobre el código, como se ve es una interfaz bastante simple con el título arriba marcado en negrita y separado del resto del fondo con un fondo de color único, siendo los botones lo único distinguible rodeado de un marco negro y un texto diciendo su función en la parte superior siendo las imágenes el botón el cual se clickea.



Al momento de darle click a la imagen de la “Gestión de almacenamiento se nos ejecutara un cmd que realice la función de la Gestión del almacenamiento a continuación se mostrará el script que realiza dicha función

#### *Script Gestión Almacenamiento en Windows:*

```

Tools Window Help Examen_U3_JuanJ - Apache NetBeans 22
Examen_U3_JuanJ - Bloc de notas
Interfaz.java
Examen_U3_JuanJ;
javax.swing.SwingUtilities;
hor CivilJuan
class Examen_U3_JuanJ {
lic static void main(String[]
// TODO code application logic
SwingUtilities.invokeLater()
    Interfaz ventana = new In
    ventana.setVisible(true);
});
echo off
echo Mostrando el almacenamiento en el sistema...
timeout /t 5 /nobreak >nul
echo.
echo Información del disco:
wmic logicaldisk get size,freespace,caption
echo.
echo Contando archivos en la carpeta TMP...
set /a countBefore=0
for %%A in (%TEMP%\*) do set /a countBefore+=1
echo Hay %countBefore% archivos en la carpeta TMP.
echo.
echo Eliminando archivos de la carpeta TMP en 5 segundos...
timeout /t 5 /nobreak >nul
del /q %TEMP%\*.* 2>nul
echo.
echo Contando archivos en la carpeta TMP después de la eliminación...
set /a countAfter=0
for %%A in (%TEMP%\*) do set /a countAfter+=1
echo Ahora hay %countAfter% archivos en la carpeta TMP.
echo.
echo Limpiando la caché del sistema en 5 segundos...
timeout /t 5 /nobreak >nul
echo Limpiando la caché del sistema...
rd /s /q %SystemRoot%\Temp 2>nul
rd /s /q %localAppData%\Microsoft\Windows\INetCache 2>nul
del /q /f %SystemRoot%\Prefetch\*.* 2>nul
echo Caché limpia.
timeout /t 5 /nobreak >nul
pause

```

Se puede apreciar que

se muestra al inicio un comando que sirve para limpiar la pantalla “@echo off” para siguiente mostrar un mensaje que dice que es el inicio del script de gestión de almacenamiento además de

darle un intervalo de 5 segundos entre cada comando para así visualizar la funciones del comando y este no se ejecute todo al mismo tiempo..

**El primer comando** es para mostrar en cmd el almacenamiento del sistema donde mostrará el almacenamiento del equipo dependiendo del número total de unidades de almacenamiento que tenga el usuario en el equipo.

**El segundo comando** es para mostrar el número total de archivos que se encuentran dentro de la Carpeta TMP que es la carpeta de archivos temporales.

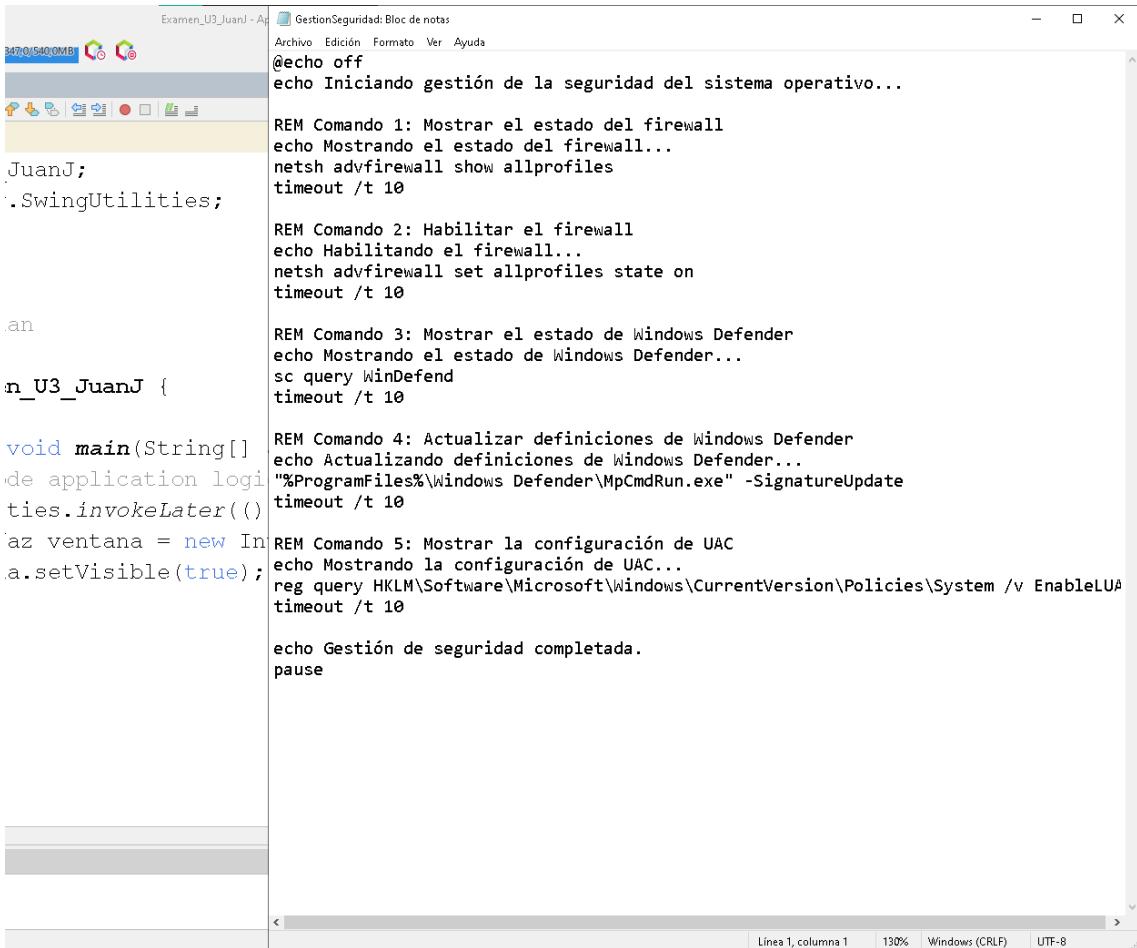
**El tercer comando** es para eliminar los archivos de la carpeta temporal y únicamente los archivos que se encuentren dentro de esa carpeta, cabe recalcar que siempre quedan un aprox de 6 archivos debido a que estos no se pueden borrar de ninguna manera de la carpeta incluso uno haciéndolo de manera manual.

**El cuarto comando** es para volver a mostrar la cantidad total de los archivos que se encuentran dentro de la carpeta TMP para verificar que realmente funcione la actividad y en los casos que he probado este si ha eliminado la mayoría de archivos y dejando los únicos archivos mencionados con anterioridad.

**El quinto comando** es el encargado de eliminar los archivos que se encuentren dentro de la memoria Caché del sistema.

Con eso finaliza el Primero Función de la interfaz realizada para la Gestión de almacenamiento del equipo.

### Script Gestión Seguridad en Windows:



The screenshot shows a Windows Notepad window titled "GestionSeguridad: Bloc de notas". The script contains several commands to manage Windows security, including the firewall and Windows Defender. The code is as follows:

```
Examen_U3_JuanJ - Ap  GestionSeguridad: Bloc de notas
Archivo Edición Formato Ver Ayuda
@echo off
echo Iniciando gestión de la seguridad del sistema operativo...
REM Comando 1: Mostrar el estado del firewall
echo Mostrando el estado del firewall...
netsh advfirewall show allprofiles
timeout /t 10
REM Comando 2: Habilitar el firewall
echo Habilitando el firewall...
netsh advfirewall set allprofiles state on
timeout /t 10
REM Comando 3: Mostrar el estado de Windows Defender
echo Mostrando el estado de Windows Defender...
sc query WinDefend
timeout /t 10
REM Comando 4: Actualizar definiciones de Windows Defender
echo Actualizando definiciones de Windows Defender...
"%ProgramFiles%\Windows Defender\MpCmdRun.exe" -SignatureUpdate
timeout /t 10
REM Comando 5: Mostrar la configuración de UAC
echo Mostrando la configuración de UAC...
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA
timeout /t 10
echo Gestión de seguridad completada.
pause
```

El script encargado de la gestión de seguridad inicia similar al de Gestión de Almacenamiento así mismo mostrando en pantalla que se ejecutara la gestión de seguridad:

**Primer comando**, este mostrará el estado del firewall con el comando “netsh advfirewall show allprofiles” este se encargará de mostrar todos los perfiles que se encuentran dentro del firewall de windows.

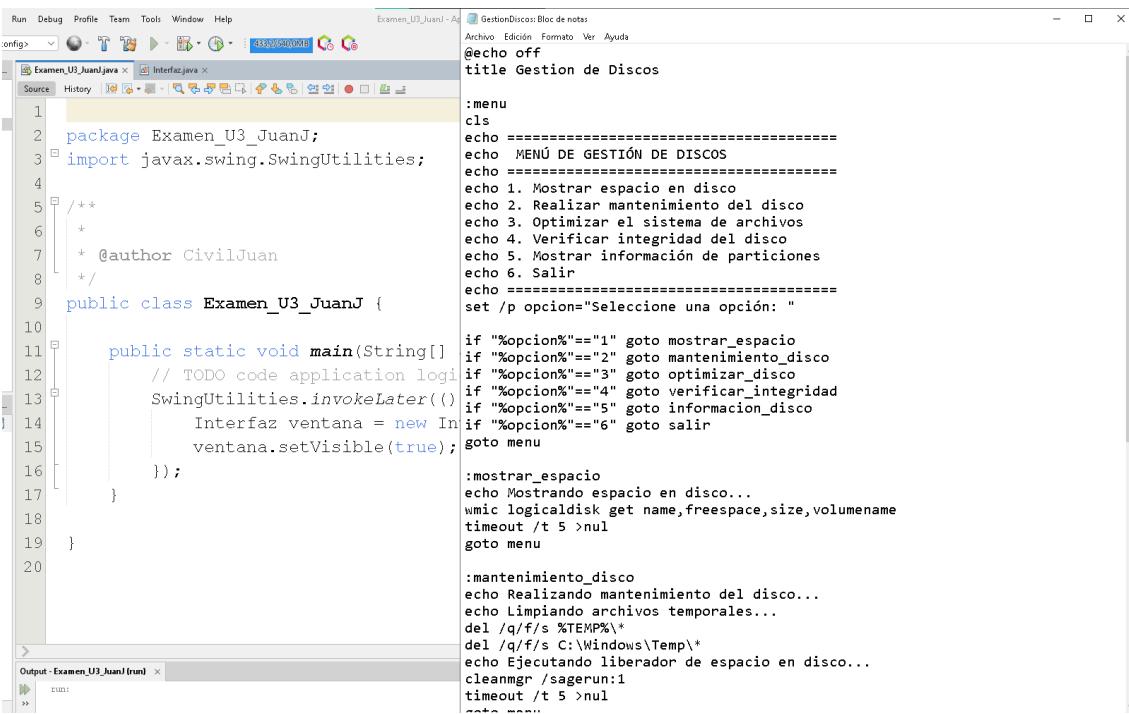
**El segundo comando** se encarga de Habilitar el firewall de windows defender, este siendo con el comando “netsh advfirewall set allprofiles state on” cabe decir que este habilita para todos los perfiles del equipo.

**El tercer comando** se encarga de mostrar el estado de Windows defender, este lo realiza con el comando “sc query WinDefend” este se encarga de verificar si el servicio de windows defender se encuentra operativo y lo mostrará en el cmd ejecutado.

**El cuarto comando** se encarga de Actualizar las definiciones del windows defender esto se refiere a que va a actualizar los parámetros y el servicio de windows defender para asi este se encuentre en la mejor condición, esto se realiza con el comando ““%ProgramFiles%\Windows Defender\MpCmdRun.exe” -SignatureUpdate”.

**El Quinto comando** y ultimo se encarga de mostrar la configuración UAC del equipo, UAC significa Control de Cuentas del Usuario o en ingles User Account Control este se encarga de realizar la gestion del control de las cuentas del equipo, este se realiza con el comando “reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA”.

#### *Script Gestión de discos en Windows:*



The screenshot shows a Java development environment with two tabs open: 'Examen\_U3\_JuanJ.java' and 'Interfaz.java'. The code in 'Examen\_U3\_JuanJ.java' is a simple Swing application. The code in 'Interfaz.java' is a batch script titled 'GestionDiscos' which handles disk management tasks like space, maintenance, optimization, and verification.

```
Examen_U3_JuanJ.java
1 package Examen_U3_JuanJ;
2 import javax.swing.SwingUtilities;
3
4 /**
 * 
 * @author CivilJuan
 */
5 public class Examen_U3_JuanJ {
6
7     public static void main(String[] args) {
8         // TODO code application logic here
9         SwingUtilities.invokeLater(() -> {
10             Interfaz ventana = new Interfaz();
11             ventana.setVisible(true);
12         });
13     }
14 }
15
16
17
18
19 }
```

```
GestionDiscos: Bloc de notas
Archivo Edición Formato Ver Ayuda
@echo off
title Gestion de Discos

:menu
cls
echo =====
echo MENÚ DE GESTIÓN DE DISCOS
echo =====
echo 1. Mostrar espacio en disco
echo 2. Realizar mantenimiento del disco
echo 3. Optimizar el sistema de archivos
echo 4. Verificar integridad del disco
echo 5. Mostrar información de particiones
echo 6. Salir
echo =====
set /p opcion="Seleccione una opción: "

if "%opcion%"=="1" goto mostrar_espacio
if "%opcion%"=="2" goto mantenimiento_disco
if "%opcion%"=="3" goto optimizar_disco
if "%opcion%"=="4" goto verificar_integridad
if "%opcion%"=="5" goto informacion_disco
if "%opcion%"=="6" goto salir
goto menu

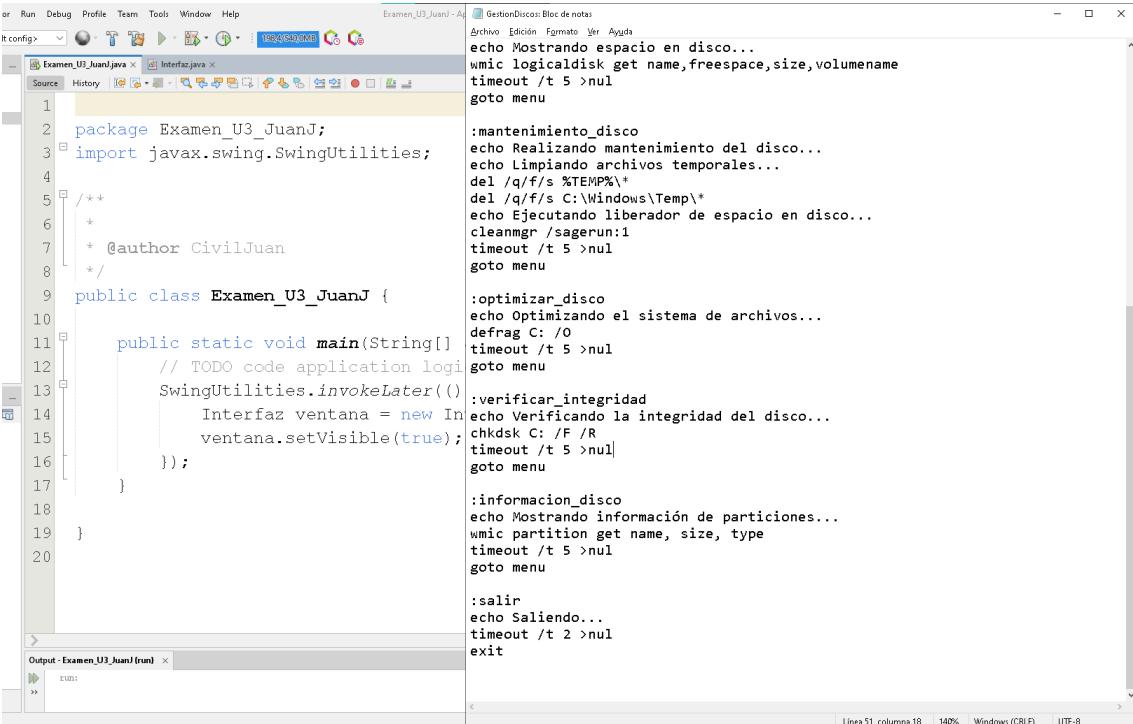
:mostrar_espacio
echo Mostrando espacio en disco...
wmic logicaldisk get name,freespace,size,volumename
timeout /t 5 >nul
goto menu

:mantenimiento_disco
echo Realizando mantenimiento del disco...
echo Limpando archivos temporales...
del /q/f/s %TEMP%\*
del /q/f/s C:\Windows\Temp\*
echo Ejecutando liberador de espacio en disco...
cleanmgr /sagerun:1
timeout /t 5 >nul
goto menu
```

En la imagen se ve el script para realizar la función de Gestión de discos el cual se encarga de manera rápida la gestión de los discos que tiene el equipo como se ve el programa al momento

de ejecutarlo abre un cmd donde nos mostrará un menú de opciones donde nos dejará realizar varias funciones para llevar a cabo la gestión de los discos:

Al momento de ingresar a la opción 1 de la gestión de discos realizará la siguiente función El **primer comando** se encarga de realizar la función de mostrar el espacio de los discos en la ventana de cmd donde este no avanzara a no ser que demos a otro botón del teclado asi devolviendonos al menu principal.



The screenshot shows an IDE interface with two tabs: 'Examen\_U3\_JuanJ.java' and 'Interfaz.java'. The 'Examen\_U3\_JuanJ.java' tab contains the following Java code:

```
1 package Examen_U3_JuanJ;
2 import javax.swing.SwingUtilities;
3
4 /**
5 * 
6 * @author CivilJuan
7 */
8 public class Examen_U3_JuanJ {
9
10    public static void main(String[] args) {
11        // TODO code application logic here
12        SwingUtilities.invokeLater(() ->
13            Interfaz ventana = new Interfaz();
14            ventana.setVisible(true);
15        });
16    }
17
18
19 }
```

To the right of the code, there is a command prompt window titled 'GestionDiscos: Bloc de notas' with the following content:

```
Examen_U3_JuanJ - Agregar archivo
Archivo Edición Formato Ver Ayuda
echo Mostrando espacio en disco...
wmic logicaldisk get name,freespace,size,volumename
timeout /t 5 >nul
goto menu

:mantenimiento_disco
echo Realizando mantenimiento del disco...
echo Limpiando archivos temporales...
del /q/f/s %TEMP%/*
del /q/f/s C:\Windows\Temp\*
echo Ejecutando liberador de espacio en disco...
cleanmgr /sagerun:1
timeout /t 5 >nul
goto menu

:optimizar_disco
echo Optimizando el sistema de archivos...
defrag C: /O
timeout /t 5 >nul
goto menu

:verificar_integridad
echo Verificando la integridad del disco...
chkdsk C: /F /R
timeout /t 5 >nul
goto menu

:informacion_disco
echo Mostrando información de particiones...
wmic partition get name, size, type
timeout /t 5 >nul
goto menu

:salir
echo Saliendo...
timeout /t 2 >nul
exit
```

Al momento de seleccionar el **segundo comando** este realizara la función de mantenimiento del disco que ejecutara la funcion de limpiar los archivos de la carpeta temporal del usuario como del sistema operativo y ejecuta el liberador de espacio del disco de windows este es una ventana que viene con windows que elimina archivos automáticamente que el sistema no vea utiles.

**El tercer comando** o tercera opción realizará la función de optimizar el disco esté siendo mediante la desfragmentación del disco para asi hacer que la información fluya de manera más rápida.

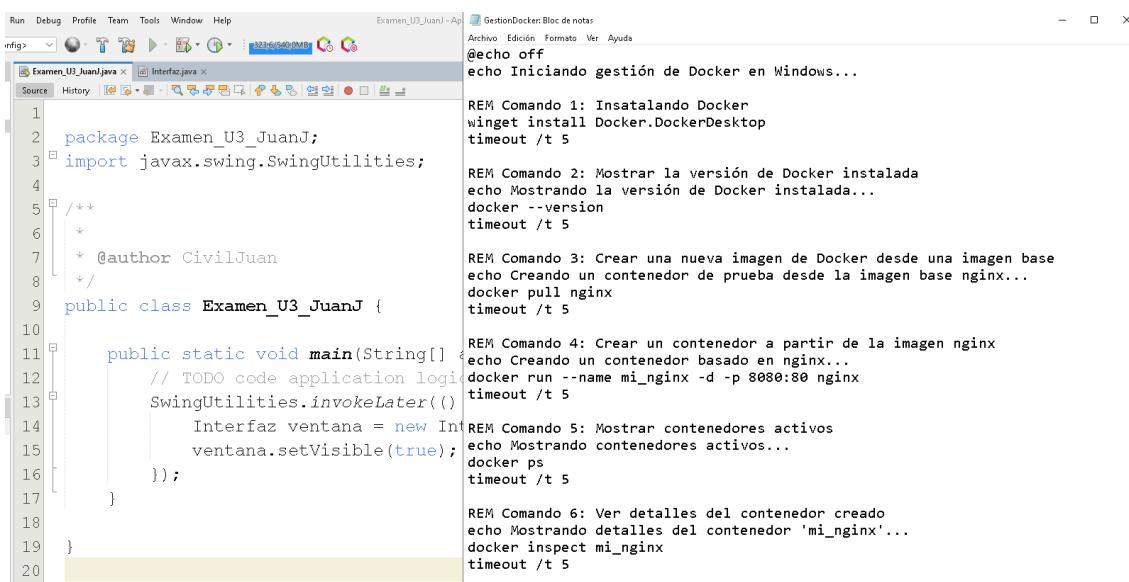
**El cuarto comando** / cuarta opción realizará la función de verificar la integridad del disco osea esto siendo de manera que verifica que el disco se encuentre en óptimo estado para evitar tener

un equipo lento debido a que al tener un disco dañado puede que el equipo tarde más en realizar actividades.

**El quinto comando** / quinta opción esta se encarga de mostrar en cmd la información de las particiones así permitiéndonos ver de manera detallada la información de las particiones que están en el equipo y ver si es necesario de tenerlas o no.

La sexta o última opción es la encargada de darle fin al script o de cerrar el cmd.

*Script Gestión de docker en Windows:*



The screenshot shows a Java development environment with two windows. On the left, the 'Source' tab of a Java file named 'Examen\_U3\_JuanJ.java' is visible, containing Java code. On the right, a terminal window titled 'GestionDocker: Bloc de notas' is open, displaying a batch script with various Docker commands. The script includes comments in Spanish explaining each command's purpose.

```
REM Comando 1: Instalando Docker
winget install Docker.DockerDesktop
timeout /t 5

REM Comando 2: Mostrar la versión de Docker instalada
echo Mostrando la versión de Docker instalada...
docker --version
timeout /t 5

REM Comando 3: Crear una nueva imagen de Docker desde una imagen base
echo Creando un contenedor de prueba desde la imagen base nginx...
docker pull nginx
timeout /t 5

REM Comando 4: Crear un contenedor a partir de la imagen nginx
echo Creando un contenedor basado en nginx...
docker run --name mi_nginx -d -p 8080:80 nginx
timeout /t 5

REM Comando 5: Mostrar contenedores activos
echo Mostrando contenedores activos...
docker ps
timeout /t 5

REM Comando 6: Ver detalles del contenedor creado
echo Mostrando detalles del contenedor 'mi_nginx'...
docker inspect mi_nginx
timeout /t 5
```

El script de gestión de docker se encarga de realizar la gestión del docker en windows para esto toca tener instalado la aplicación docker desktop:

**El primero comando** se encarga de instalar la dicha aplicación debido a que esta es necesaria para la funciones siguientes, este realiza además la función de actualizar la aplicación en caso de haber alguna.

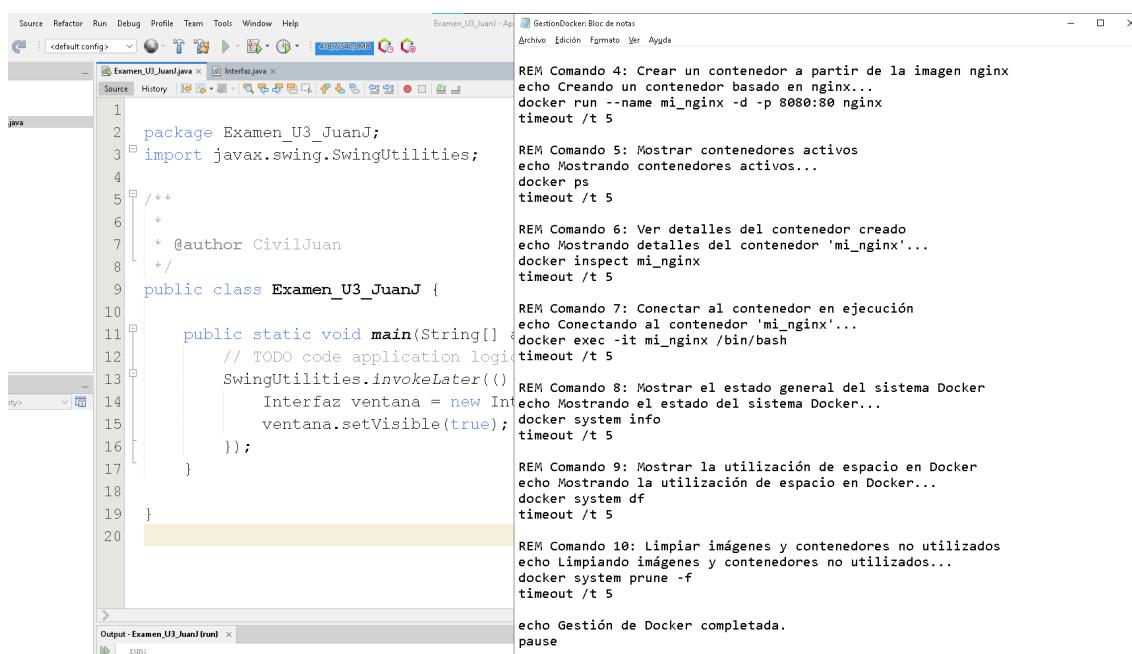
**El segundo comando** se encarga de verificar la version de Docker instalada en el equipo.

**El tercer comando** se encarga de crear una imagen de Docker desde alguna imagen de base del equipo.

**El cuarto comando** se encarga de generar un contenedor (docker) que obtenga una imagen a partir de nginx.

**El quinto comando** se encarga de mostrar los contenedores (dockers) activos, osea que mostrará los contenedores que están dentro del sistema de dockers y que se encuentre operativos.

**El sexto comando** permite ver el contenedor creado que se especifique en este caso mostrará el que se creó recientemente.



The screenshot shows a Java IDE interface with two tabs open: 'Examen\_U3\_JuanJava' and 'Interfazjava'. The code editor contains a Java class named Examen\_U3\_JuanJ with a main method. To the right of the code editor is a terminal window titled 'GestionDocker: Bloc de notas' containing a series of REM comments and corresponding docker commands:

```
REM Comando 4: Crear un contenedor a partir de la imagen nginx
echo Creando un contenedor basado en nginx...
docker run --name mi_nginx -d -p 8080:80 nginx
timeout /t 5

REM Comando 5: Mostrar contenedores activos
echo Mostrando contenedores activos...
docker ps
timeout /t 5

REM Comando 6: Ver detalles del contenedor creado
echo Mostrando detalles del contenedor 'mi_nginx'...
docker inspect mi_nginx
timeout /t 5

REM Comando 7: Conectar al contenedor en ejecución
echo Conectando al contenedor 'mi_nginx'...
docker exec -it mi_nginx /bin/bash
timeout /t 5

REM Comando 8: Mostrar el estado general del sistema Docker
echo Mostrando el estado del sistema Docker...
docker system info
timeout /t 5

REM Comando 9: Mostrar la utilización de espacio en Docker
echo Mostrando la utilización de espacio en Docker...
docker system df
timeout /t 5

REM Comando 10: Limpiar imágenes y contenedores no utilizados
echo Limpiando imágenes y contenedores no utilizados...
docker system prune -f
timeout /t 5

echo Gestión de Docker completada.
pause
```

**El séptimo comando** se encarga de conectar al contenedor (docker) que se encuentre en ejecución o sea el que recién se creó.

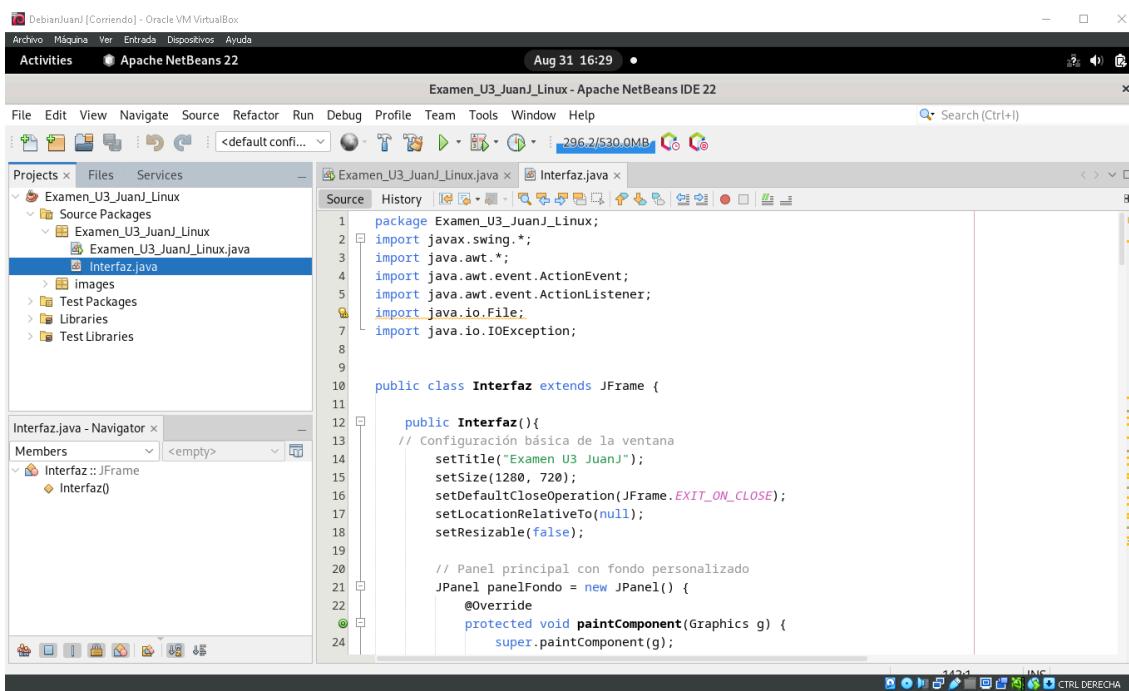
**El Octavo comando** se encarga de mostrar el estado del sistema de docker osea que mostrará información sobre el estado del servicio de docker.

**El noveno comando** se encarga de mostrar el espacio total usado por el servicio de docker como de los dockers que se encuentren en el programa.

**El décimo comando** se encarga de limpiar las imágenes y docker que no se utilicen esto lo que hace es eliminar los dockers que no estén en operación para así optimizar el uso del almacenamiento del equipo.

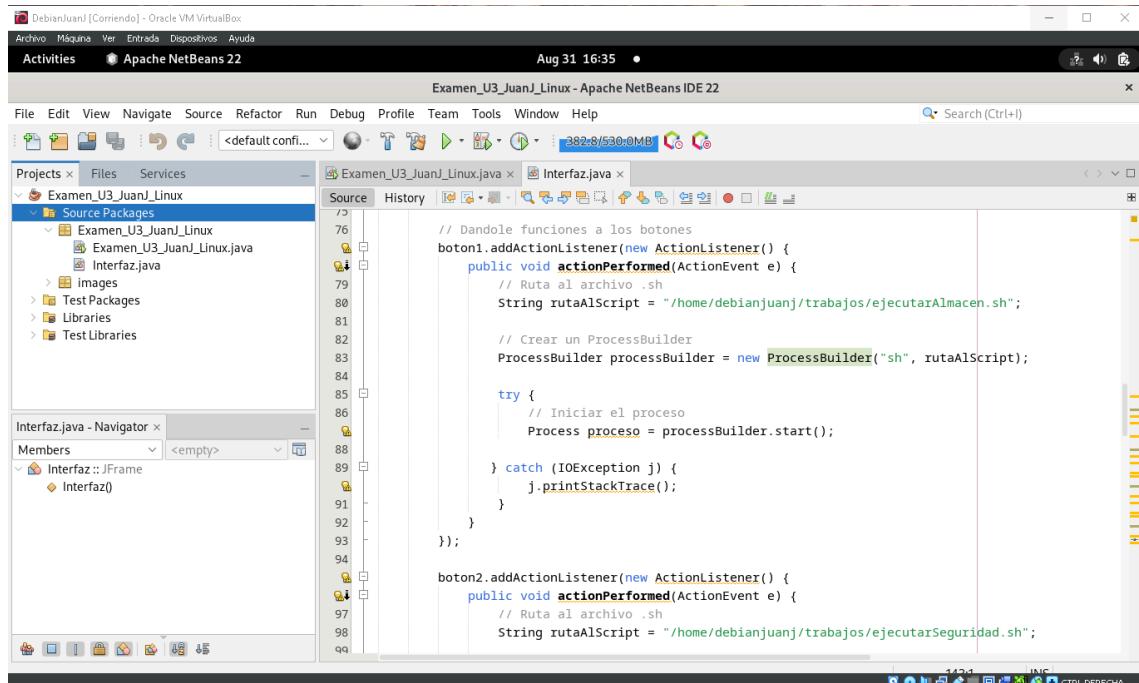
Quiero afirmar que yo pienso que nomas con estos 4 botones serian necesarios para realizar todos los 20 scripts debido a que no hay mucho que para hacerle con los temas aprendidos aparte de estos a no ser que sepáremos cada comando en un script distinto cosa que no veo tan necesario ademas de que al agregar mas botones la interfaz podría llevar a ser bastante confusa y de aspecto poco agradable por lo que decidí nomas trabajar con estos 4 botones que realizan múltiples funciones.

### ***Programación en java código Debian:***



En la parte de linux de la interfaz de los botones que ejecuten los scripts este en su mayoria es muy similar a su parte de Windows así que se prefirió solo explicar los cambios realizados para el funcionamiento del programa en Debian (linux).

## Cambios:



The screenshot shows the Apache NetBeans IDE 22 interface. The title bar reads "Examen\_U3\_JuanJ\_Linux - Apache NetBeans IDE 22". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for file operations like New, Open, Save, and Run. The Projects tab shows a single project "Examen\_U3\_JuanJ\_Linux" with a "Source Packages" node containing files "Examen\_U3\_JuanJ\_Linux.java" and "Interfaz.java". The Source tab displays the Java code for "Interfaz.java". The code uses `ProcessBuilder` to execute shell scripts. The Navigator tab shows the members of the "Interfaz" class.

```
// Dandole funciones a los botones
boton1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Ruta al archivo .sh
        String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarAlmacen.sh";

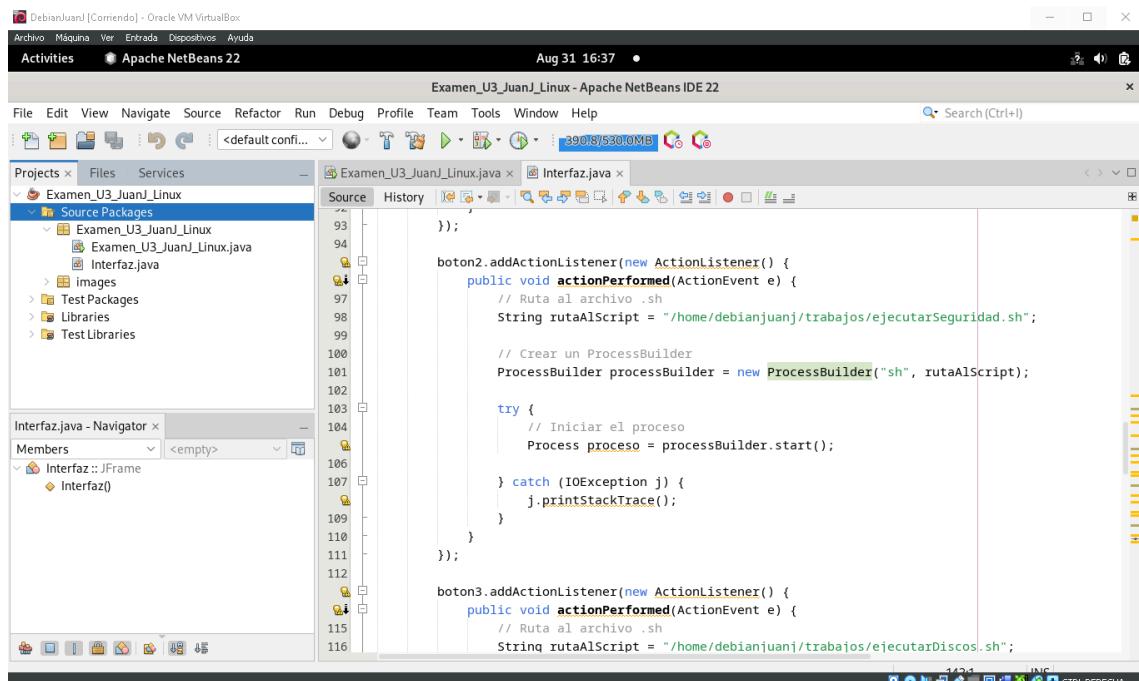
        // Crear un ProcessBuilder
        ProcessBuilder processBuilder = new ProcessBuilder("sh", rutaAlScript);

        try {
            // Iniciar el proceso
            Process proceso = processBuilder.start();

        } catch (IOException j) {
            j.printStackTrace();
        }
    }
});

boton2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Ruta al archivo .sh
        String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarSeguridad.sh";
    }
});
```

El cambio mas notorio está en el apartado de la ejecución del script donde se realizo uso de la librería de “ProcessBuilder” que por su traducción se diría constructor de procesos que se encarga de ejecutar el proceso indicado en este caso ejecuta el script que se le selecciona en la ruta.



This screenshot shows the same Apache NetBeans IDE 22 interface as the previous one, but with additional code in the "Interfaz.java" source editor. The new code adds three more buttons (boton3, boton4, boton5) with their respective action listeners, each executing a different shell script from the "/home/debianjuanj/trabajos/" directory.

```
});
```

```
boton2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Ruta al archivo .sh
        String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarSeguridad.sh";

        // Crear un ProcessBuilder
        ProcessBuilder processBuilder = new ProcessBuilder("sh", rutaAlScript);

        try {
            // Iniciar el proceso
            Process proceso = processBuilder.start();

        } catch (IOException j) {
            j.printStackTrace();
        }
    }
});
```

```
boton3.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Ruta al archivo .sh
        String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarDiscos.sh";
    }
});
```

```
boton4.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Ruta al archivo .sh
        String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarAlmacen.sh";
    }
});
```

```
boton5.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Ruta al archivo .sh
        String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarSeguridad.sh";
    }
});
```

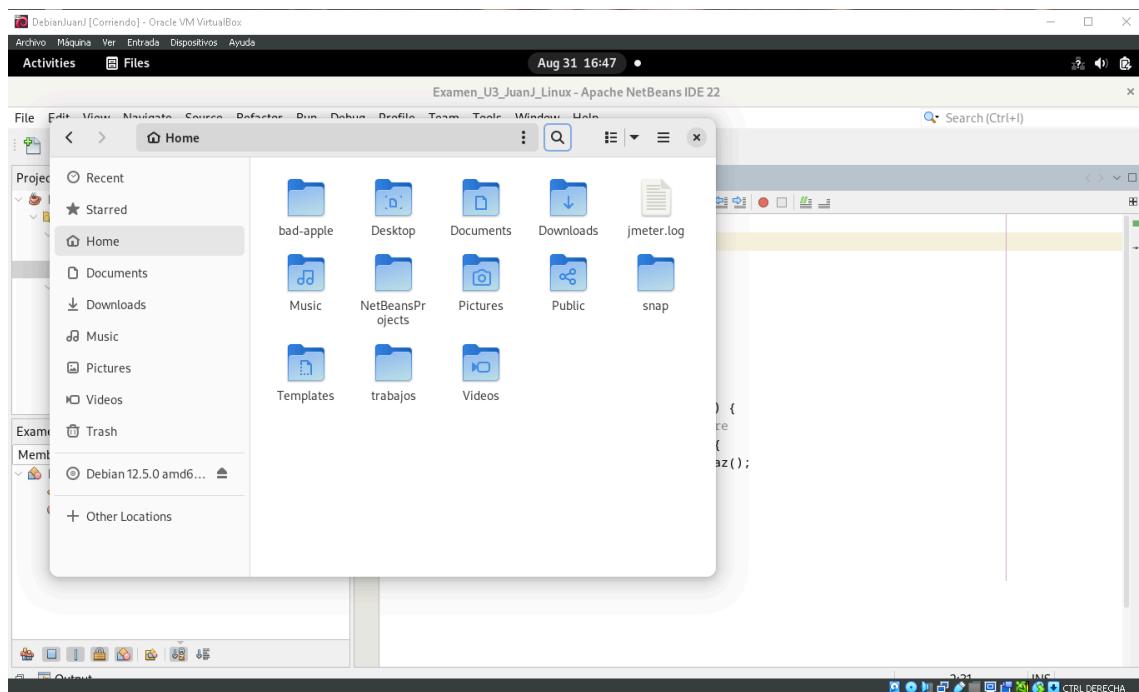
```
110     });
111 });
112 });
113 JButton boton3 = new JButton("Ejecutar Discos");
114 boton3.addActionListener(new ActionListener() {
115     public void actionPerformed(ActionEvent e) {
116         // Ruta al archivo .sh
117         String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarDiscos.sh";
118
119         // Crear un ProcessBuilder
120         ProcessBuilder processBuilder = new ProcessBuilder("sh", rutaAlScript);
121
122         try {
123             // Iniciar el proceso
124             Process proceso = processBuilder.start();
125
126         } catch (IOException j) {
127             j.printStackTrace();
128         }
129     }
130 });
131
132 JButton boton4 = new JButton("Ejecutar Docker");
133 boton4.addActionListener(new ActionListener() {
134     public void actionPerformed(ActionEvent e) {
135         // Ruta al archivo .sh
136         String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarDocker.sh";
137
138         // Crear un ProcessBuilder
139         ProcessBuilder processBuilder = new ProcessBuilder("sh", rutaAlScript);
140
141         try {
142             // Iniciar el proceso
143             Process proceso = processBuilder.start();
144
145         } catch (IOException j) {
146             j.printStackTrace();
147         }
148     }
149
150     //Añadir texto a la ventana
151     JLabel texto1 = new JLabel("Gestion del Almacenamiento");
152     texto1.setBounds(365, 170, 250, 75);
153     texto1.setFont(new Font("Sans", Font.BOLD, 15));
154     JLabel texto2 = new JLabel("Gestion de Seguridad");
```

```
130     JButton boton4 = new JButton("Ejecutar Docker");
131     boton4.addActionListener(new ActionListener() {
132         public void actionPerformed(ActionEvent e) {
133             // Ruta al archivo .sh
134             String rutaAlScript = "/home/debianjuanj/trabajos/ejecutarDocker.sh";
135
136             // Crear un ProcessBuilder
137             ProcessBuilder processBuilder = new ProcessBuilder("sh", rutaAlScript);
138
139             try {
140                 // Iniciar el proceso
141                 Process proceso = processBuilder.start();
142
143             } catch (IOException j) {
144                 j.printStackTrace();
145             }
146         }
147     });
148
149     //Añadir texto a la ventana
150     JLabel texto1 = new JLabel("Gestion del Almacenamiento");
151     texto1.setBounds(365, 170, 250, 75);
152     texto1.setFont(new Font("Sans", Font.BOLD, 15));
153     JLabel texto2 = new JLabel("Gestion de Seguridad");
```

Así mismo se realizó el cambio en los demás botones para que estos ejecuten el script pertinente y logre realizar su debido funcionamiento llamando a cada uno de los scripts necesarios para su debida ejecución.

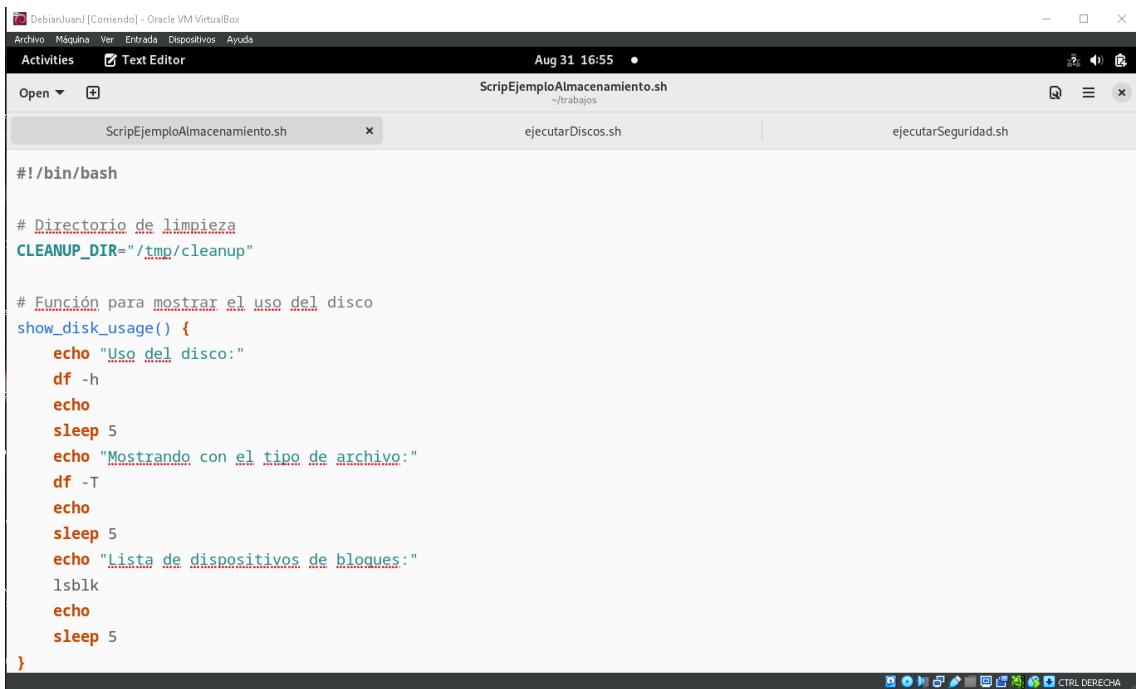
### **Scripts linux:**

En el aparto de scripts como en linux ejecuta por directorios se recomienda colocar la carpeta “trabajos” que se encuentra en la parte de Linux en el GitHub en la carpeta home para no realizar cambios dentro del script.



En la imagen se muestra el lugar donde se debe encontrar la carpeta “trabajos” para realizar su debido funcionamiento del programa.

## **Script Gestión de almacenamiento:**



The screenshot shows a Linux desktop environment with a terminal window titled "Text Editor". The window contains a script named "ScripEjemploAlmacenamiento.sh" located at "/trabajos". The script content is as follows:

```
#!/bin/bash

# Directorio de limpieza
CLEANUP_DIR="/tmp/cleanup"

# Función para mostrar el uso del disco
show_disk_usage() {
    echo "Uso del disco:"
    df -h
    echo
    sleep 5
    echo "Mostrando con el tipo de archivo:"
    df -T
    echo
    sleep 5
    echo "Lista de dispositivos de bloques:"
    lsblk
    echo
    sleep 5
}
```

En el script de Gestión de Almacenamiento se encuentra los siguientes comandos además de tener cada comando separado por un tiempo de espera de 5 segundos entre cada comando::

**Comando 1:** el comando “df -h” se encarga de mostrar el uso del disco, osea la información del almacenamiento del equipo.

**Comando 2:** el comando “df -T” se encarga de mostrar el uso del disco pero esta vez va a mostrar dependiendo el tipo de archivos.

**Comando 3:** el comando “lsblk” se encarga de mostrar la lista de los dispositivos de almacenamiento por bloques.

```
#!/bin/bash

# Función para limpiar los archivos temporales
cleanup_temp_files() {
    echo "Mostrando en pantalla el contenido de las carpetas Temporales"
    ls /tmp/*
    echo
    sleep 5
    echo "Limpiando archivos temporales..."

    # Limpiar /tmp
    sudo rm -rf /tmp/*
    echo "/tmp limpiado."
    sleep 5

    # Muestra un mensaje de que se termino de limpiar
    echo "Limpieza completada."
    echo
}

cleanup_temp_files
```

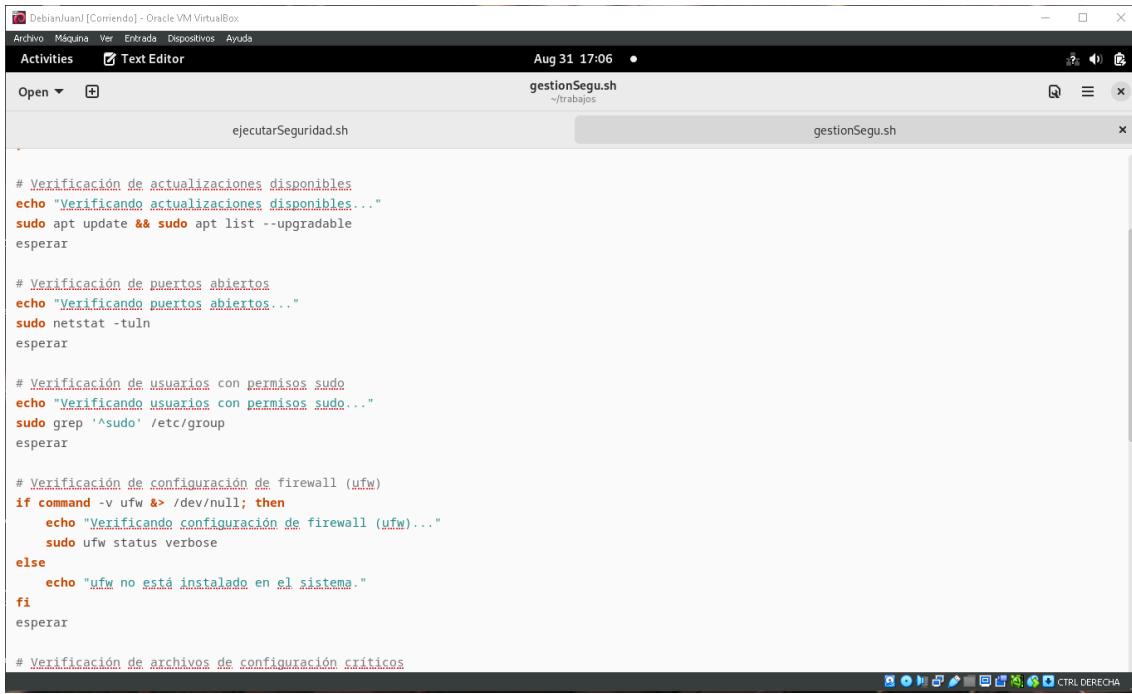
**Comando 4:** el comando “ls /tmp/\*” se encarga de mostrar todos los archivos presentes en la carpeta de archivos Temporales o TMP.

**Comando 5:** el comando “sudo rm -rf /tmp/\*” se encarga que mediante el comando sudo se ingrese a tener prioridades de administrador para realizar la función de rm que es para eliminar directorios y -rf para borrar archivos y la última parte /tmp/\* se encarga de darle la ubicación que va a eliminar.

**Comando 6:** el comando “sudo rm -rf /var/tmp/\*” este se encarga de realizar la misma función de la anterior solo que a otra carpeta en este caso la “/var/tmp”.

Estos serían todos los comandos que se realizaron para el script de Gestión de almacenamiento.

## **Script Gestión de Seguridad:**



```
# Verificación de actualizaciones disponibles
echo "Verificando actualizaciones disponibles..."
sudo apt update && sudo apt list --upgradable
esperar

# Verificación de puertos abiertos
echo "Verificando puertos abiertos..."
sudo netstat -tuln
esperar

# Verificación de usuarios con permisos sudo
echo "Verificando usuarios con permisos sudo..."
sudo grep '^sudo' /etc/group
esperar

# Verificación de configuración de firewall (ufw)
if command -v ufw > /dev/null; then
    echo "Verificando configuración de firewall (ufw)..."
    sudo ufw status verbose
else
    echo "ufw no está instalado en el sistema."
fi
esperar

# Verificación de archivos de configuración críticos
```

El script de gestión de seguridad se encarga de darle mantenimiento de la seguridad del sistema operativo de Debian o de donde se ejecute el script dependiendo si es compatible con los comandos para Debian:

**Comando 1:** el comando “sudo apt update && sudo apt list –upgradable” este comando se encarga de realizar la verificación de actualizaciones del equipo como de las librerías que este tiene además de los servicios, esto hace que el equipo esté actualizado a las últimas funciones en términos de la seguridad.

**Comando 2:** el comando “sudo netstat -tuln” se encarga de verificar los puertos en la red que se encuentre abiertos así verificando que puertos nomas se encuentra el equipo y verificar que este en los necesarios.

**Comando 3:** el comando “sudo grep '^sudo' /etc/group” se encarga de verificar los usuarios que se encuentre con permisos de sudo o de administrador.

**Comando 4:** el comando “ufw status verbose” este se encarga de verificar la configuración de firewall del equipo, verificando su estado y si se encuentra operativo.

```

# Verificación de archivos de configuración críticos
echo "Verificando archivos de configuración críticos..."
echo "Contenido de /etc/passwd:"
cat /etc/passwd | head -n 10
esperar

echo "Contenido de /etc/shadow:"
sudo cat /etc/shadow | head -n 10
esperar

echo "Operaciones de verificación de seguridad completadas."

```

**El comando 5:** el comando “cat /etc/passwd | head -m 10” este se encarga de verificar los archivos de configuración importantes o críticos como lo sería las contraseñas de usuarios.

**El comando 6:** el comando “sudo cat /etc/shadow | head -n 10” este se encarga de mostrar también más información importante de las cuentas de usuarios del equipo.

### *Script Gestión de Discos:*

```

# Mostrar la información de los discos
echo "Información de los discos:"
lsblk -f
esperar

# Mostrar información detallada de los discos
echo "Información detallada de los discos:"
sudo fdisk -l
esperar

# Crear una partición
# Cambia /dev/sdx por el disco en el que deseas crear la partición
DISCO="/dev/sdx"
PARTICION="${DISCO}1"
echo "Creando una partición en $DISCO..."
echo -e "n\np\n1\n\n\n" | sudo fdisk $DISCO
esperar

# Mostrar información después de crear la partición
echo "Información de los discos después de crear la partición:"
lsblk -f
esperar

# Eliminar una partición
# Cambia ${DISCO}1 por la partición que deseas eliminar

```

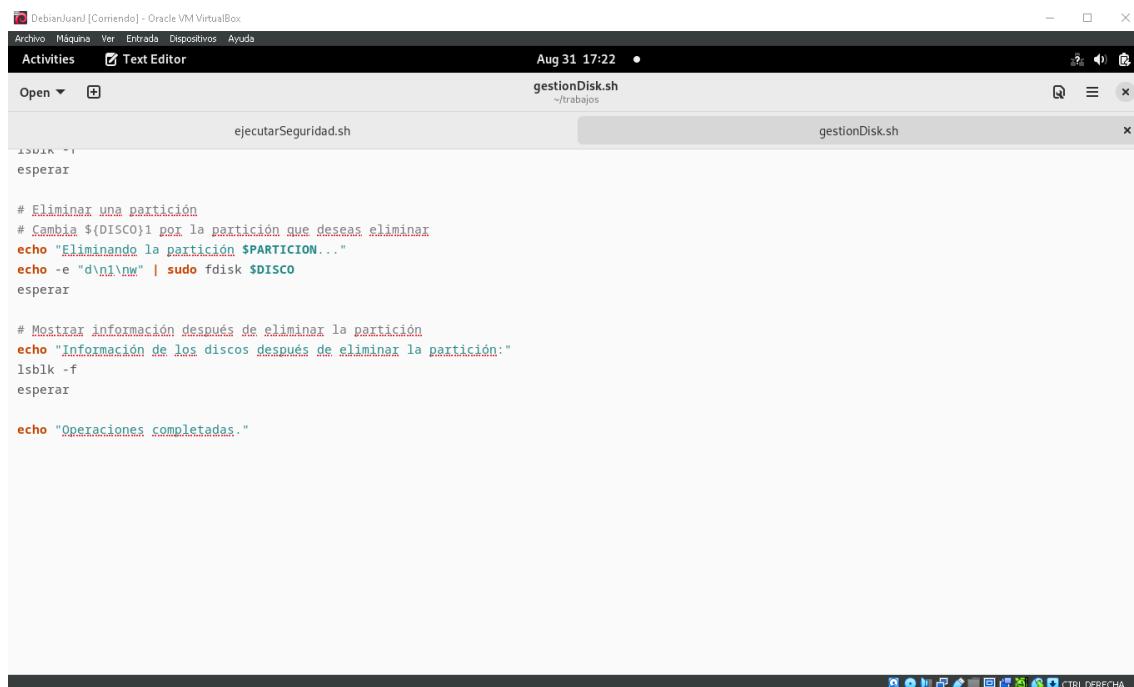
El script de gestión de discos es el encargado de llevar la gestión de los discos del equipo que se encuentra el SO a continuación se dirán los comandos del script:

**Comando 1:** el comando “lsblk -f” es el encargado de mostrar información de los discos como lo sería su espacio, el nombre, entre otras.

**Comando 2:** el comando “sudo fdisk -l” es el encargado de mostrar de manera más detallada la información de los discos debido a eso requiere el comando sudo, este mostrará más información de los discos de manera que el usuario tenga un mejor entendimiento del estado del disco.

**Comando 3:** el comando “echo -e “n\np\n1\ rw” | sudo fdisk \$DISCO” este se encarga de crear una nueva partición del disco en este caso como parámetros se creó variables para ahorrar caracteres.

**Comando 4:** se vuelve a colocar el comando “lsblk -f” para mostrar de nuevo las particiones y comprobar que se creó la nueva partición correctamente.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Text Editor". Inside the terminal, a shell script named "gestionDisk.sh" is displayed. The script contains the following code:

```
# Eliminar una partición
# Cambia ${DISCO}1 por la partición que deseas eliminar
echo "Eliminando la partición $PARTICION..."
echo -e "d\n1\ nw" | sudo fdisk $DISCO
esperar

# Mostrar información después de eliminar la partición
echo "Información de los discos después de eliminar la partición:"
lsblk -f
esperar

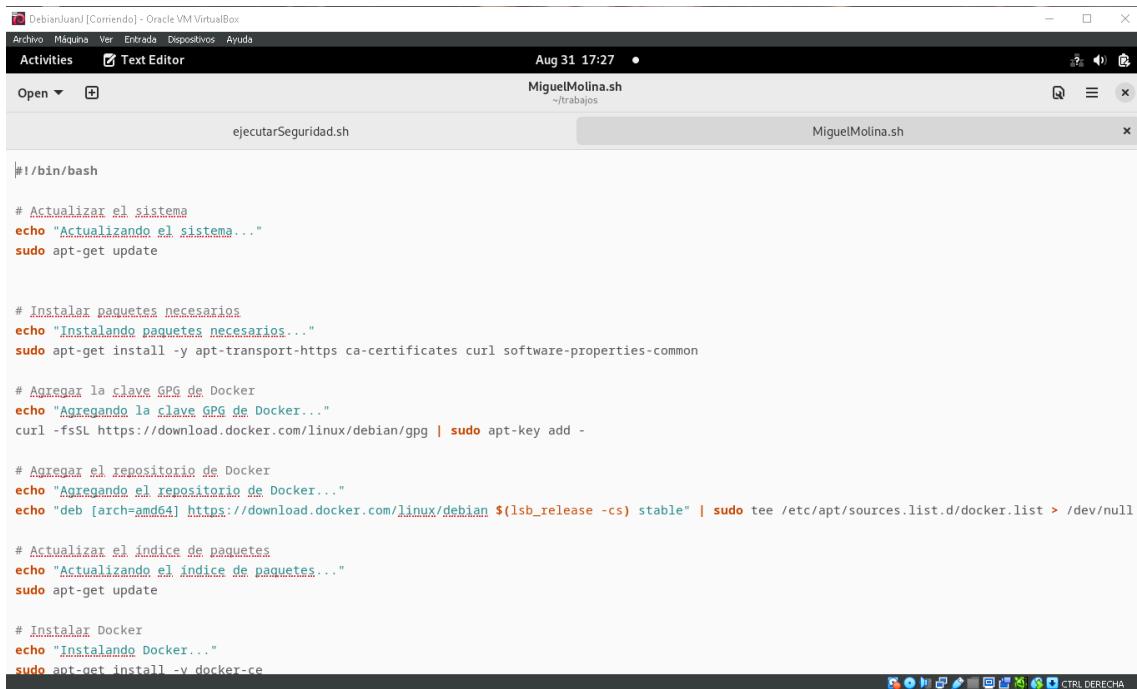
echo "Operaciones completadas."
```

**El comando 5:** es el “echo -e “d\n1\ rw” | sudo fdisk \$DISCO” el comando realiza la función de

eliminar la partición creada anteriormente debido a que se considera que no es necesario sino sólo para realizar el ejemplo de la actividad.

**El comando 6:** nuevamente es el comando para realizar la función de mostrar las particiones del disco con el comando “lsblk -f” este para verificar que se elimina correctamente la partición.

### **Script Gestión de Docker:**



```
#!/bin/bash

# Actualizar el sistema
echo "Actualizando el sistema..."
sudo apt-get update

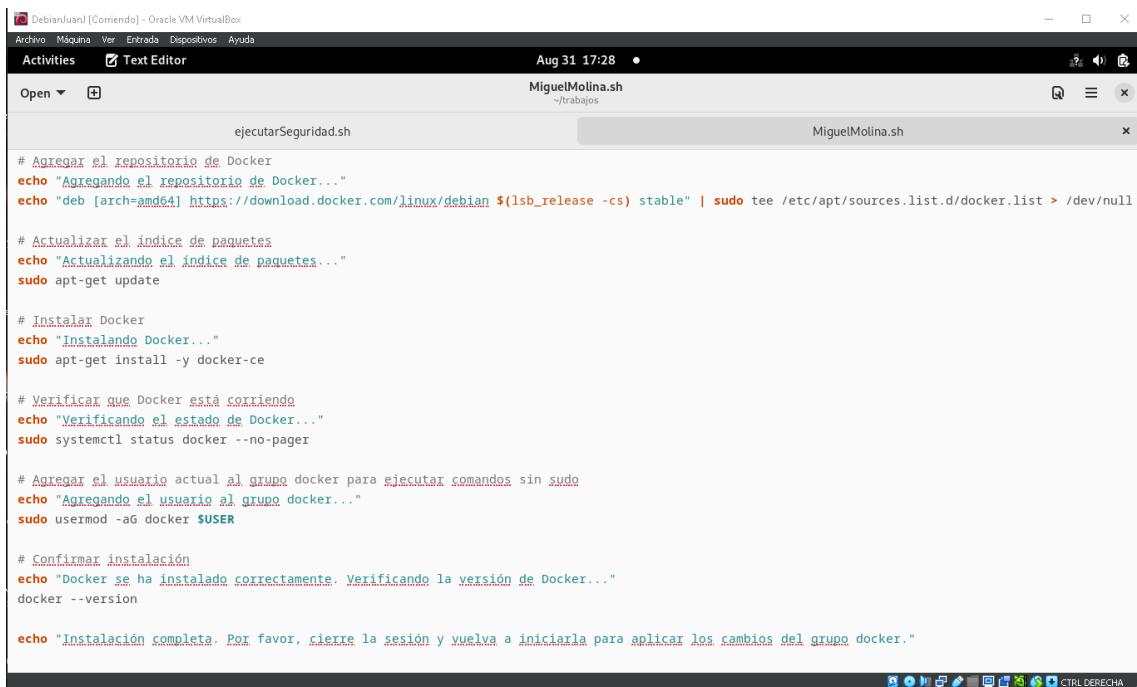
# Instalar paquetes necesarios
echo "Instalando paquetes necesarios..."
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

# Agregar la clave GPG de Docker
echo "Agregando la clave GPG de Docker..."
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -

# Agregar el repositorio de Docker
echo "Agregando el repositorio de Docker..."
echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Actualizar el índice de paquetes
echo "Actualizando el índice de paquetes..."
sudo apt-get update

# Instalar Docker
echo "Instalando Docker..."
sudo apt-get install -y docker-ce
```



```
# Agregar el repositorio de Docker
echo "Agregando el repositorio de Docker..."
echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Actualizar el índice de paquetes
echo "Actualizando el índice de paquetes..."
sudo apt-get update

# Instalar Docker
echo "Instalando Docker..."
sudo apt-get install -y docker-ce

# Verificar que Docker está corriendo
echo "Verificando el estado de Docker..."
sudo systemctl status docker --no-pager

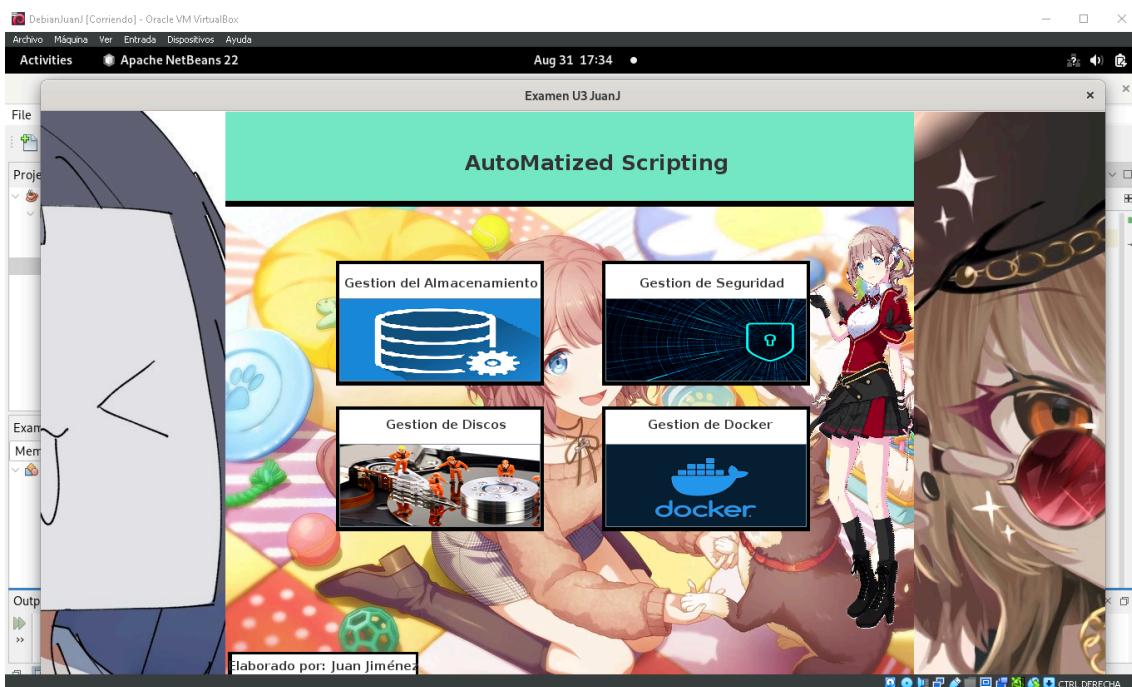
# Agregar el usuario actual al grupo docker para ejecutar comandos sin sudo
echo "Agregando el usuario al grupo docker..."
sudo usermod -aG docker $USER

# Confirmar instalación
echo "Docker se ha instalado correctamente. Verificando la versión de Docker..."
docker --version

echo "Instalación completa. Por favor, cierre la sesión y vuelva a iniciarla para aplicar los cambios del grupo docker."
```

El Script de gestión de docker ya fue presentado con anterioridad en otro trabajo debido a que fue explicado como actividad de la Evaluación parcial 1 de la 3ra Unidad debido a eso se decidió para este informe realizar únicamente la explicación del funcionamiento del script, este inicialmente actualiza los repositorios del equipo para siguiente instalar la aplicación de Docker para el sistema de Debian así que con eso este script ya no funcionaría para otro archivo que no sea Debian, se agrega el repositorio de Docker al archivo que contiene los repositorios del equipo vuelve a actualizar los repositorios para siguiente se inicializa el docker, se verifica que esté funcionando su servicio, se agrega el usuario que ejecute el script al grupo que pueda realizar uso de los comandos Docker y se confirma la instalación para finalizar el script mostrará la versión instalada de docker.

### **Ejecución de la interfaz en Debian:**



La ejecución del programa de interfaz en debian es similar a su parte de Windows debido a que este presenta la misma interfaz, como se explicó anteriormente lo unico distinto es la función de ejecución de los scripts y que estamos en otro SO que funciona con otros tipos de archivos en este caso los de extensión .sh.

#### **4. Conclusiones**

- Crear un programa con interfaces gráficas específicas para Windows y Debian muestra la capacidad de adaptar el software a diferentes sistemas operativos. Esto resalta la importancia de considerar las particularidades de cada entorno, como los comandos y scripts específicos, lo que demuestra una sólida comprensión de la portabilidad y la necesidad de un enfoque personalizado para cada plataforma.
- La implementación de una interfaz gráfica en Java para ejecutar scripts del sistema operativo simplifica la interacción del usuario con tareas complejas.
- La automatización de tareas de administración de sistemas a través de una interfaz gráfica mejora la productividad y reduce el margen de error humano. La capacidad de controlar scripts del sistema operativo desde una GUI permite una administración más eficiente y menos propensa a errores.

#### **5. Recomendaciones**

- Se recomienda que la interfaz gráfica sea diseñada de manera intuitiva, con una disposición de botones clara y coherente entre las versiones de Windows y Debian.
- Es crucial realizar pruebas exhaustivas en ambas plataformas (Windows y Debian) para asegurar que los scripts se ejecuten correctamente y que la interfaz gráfica funcione sin problemas.
- Proporcionar documentación clara y detallada para la instalación y uso del programa en ambos sistemas operativos es esencial. Además, se recomienda establecer un canal de soporte técnico para ayudar a los usuarios a resolver problemas que puedan surgir, especialmente en la ejecución de scripts personalizados o en configuraciones específicas del sistema operativo.

## **6. Bibliografía/ Referencias**

- "Sistemas Operativos Modernos" de Andrew S. Tanenbaum.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (Año desconocido). Conceptos de Sistemas Operativos.
- Tanenbaum, A. S. (Año desconocido). Sistemas Operativos Modernos.
- Love, R. (Año desconocido). Desarrollo del Kernel de Linux.
- Russinovich, M., Solomon, D. A., & Ionescu, A. (Año desconocido). Internals de Windows

## **7. Anexos:**

**Link GitHub:** <https://github.com/Juandjr/Sistemas-Operativos.git>