



Proyecto | 2

Programación Orientada a Objetos | IC2101

Profesor:

Luis Pablo Soto Chavez

Grupo 60

Pertenece a:

Berenice Amador Pereira | 2023152721

Deywenie Smith Gregory | 2024096722

Hydia Thomas Hodgson | 2023395892

Centro Académico de Limón

Semestre II | 2024

Introducción

El objetivo de este proyecto es implementar un administrador de archivos con interfaz gráfica donde se muestre todo el contenido de la unidad C// por defecto oh si se tiene una unidad extraíble USB que se muestre todo el contenido de esta. Tiene las funcionalidades básicas para manejar archivos como explorar el contenido, crear directorios, eliminar archivos y directorios, consultar propiedades y abrir archivos con su aplicación asociada. Este proyecto se realizó en el lenguaje de programación java y utilizando el patrón Modelo-Vista-Controlador, además, tiene una interfaz intuitiva y amigable con el usuario.

Índice

1) Estrategia de solución	4
1.1 Estrategia Metodológica	4
1.2 Diagrama de Gantt	5
1.3 Diagrama General de casos de uso	6
1.4 Diagrama de arquitectura conceptual	7
1.5 Diagrama de clases de bajo nivel	7
1.6 Justificación de las relaciones establecidas entre los objetos del diagrama	8
1.7 Diagrama de paquetes	8
2) Análisis de Resultados	9
3) Enlace del JavaDoc	9
4) Aspectos relevantes y lecciones aprendidas	9
5) Bitácora o diario de trabajo	11
6) Bibliografía, fuentes digitales y APIs utilizadas	11
Bibliografía	11
APIs Utilizadas	11
7) Manual de usuario	13

1) Estrategia de solución

1.1 Estrategia Metodológica

En este proyecto se siguió un enfoque metodológico basado en el trabajo colaborativo y orientado a objetos, también se aplicó principios de planeación organización y seguimiento para así poder garantizar que el proyecto este completo en el tiempo estimado.

Cada miembro del equipo asumió un rol activo y muy participativo desde la codificación hasta la documentación todas estuvieron involucradas así llevaron a cabo los diferentes puntos que se requerían para la completitud del proyecto.

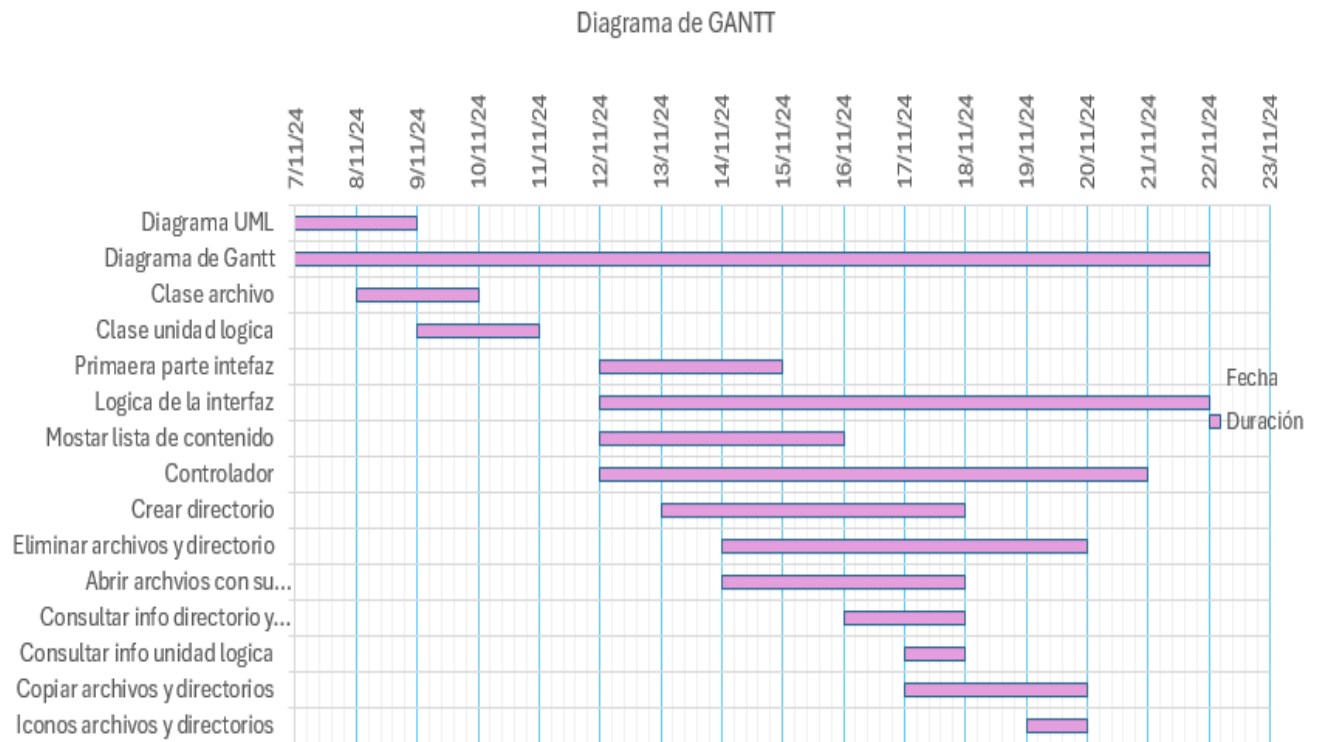
Las tareas se repartieron de acorde a la disponibilidad de cada participante teniendo en cuentas las evaluaciones que cada una tenía que realizar y el tiempo de entrega que tenían se repartió la parte lógica entre las 3 tomando en cuenta lo dicho anteriormente y de la misma forma se repartió la documentación del proyecto.

Se hicieron reuniones de forma periódicas en modalidad remota tanto para llegar a acuerdos y tomar decisiones de codificación como también para probar el programa y así poder detectar vulnerabilidades o errores para poder arreglarlos con tiempo.

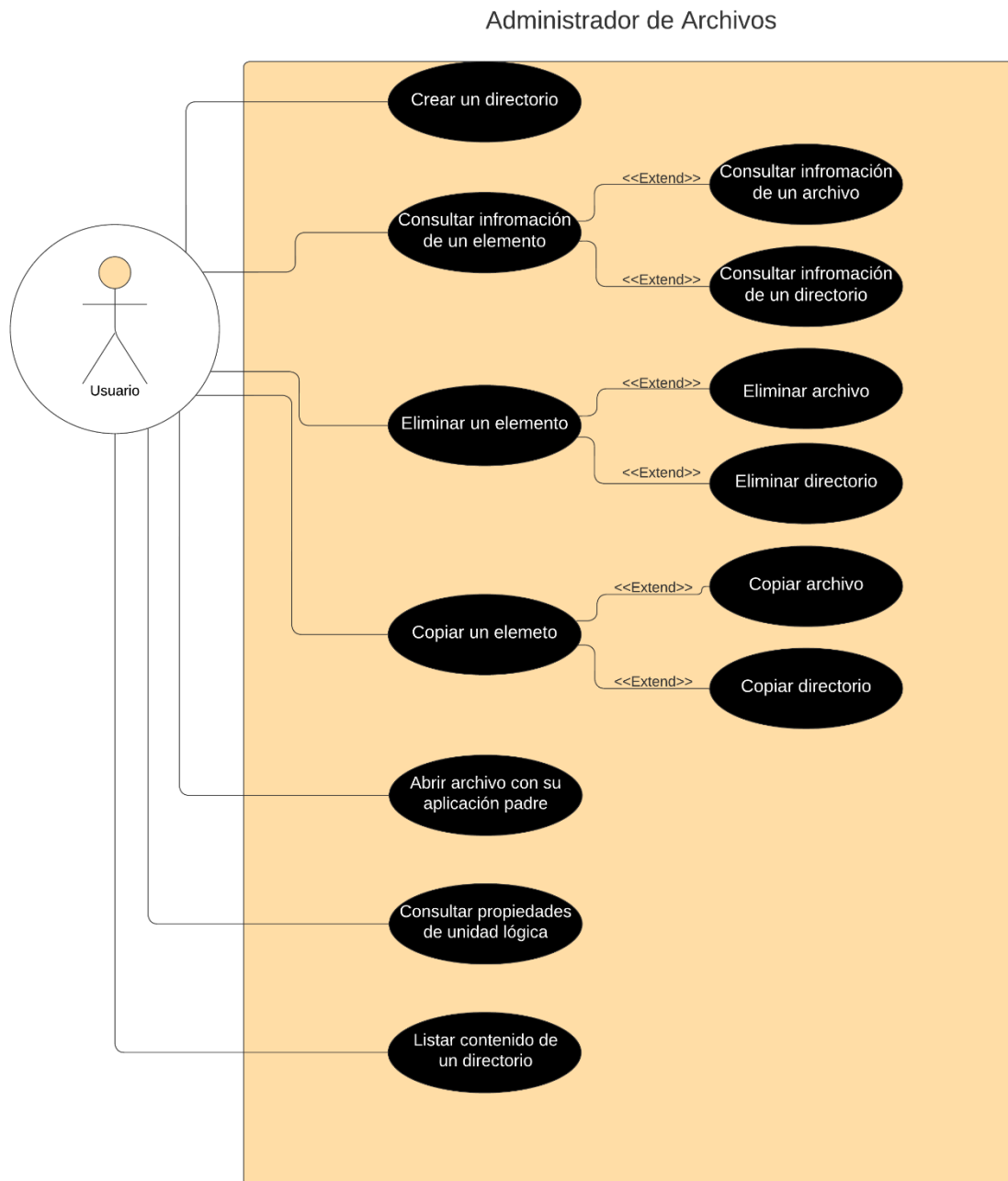
Se utilizaron varias herramientas algunas fueron:

- **Netbeans** como IDLE para la codificación de proyecto.
- **Google Drive** para compartir documentación y archivos relacionados al proyecto.
- **GitHub** para el control de versiones y la consolidación del código.
- **Lucidchart** para que todos los miembros del equipo puedan acceder y editar a los diferentes diagramas para la elaboración del proyecto.
- **Teams** como herramientas para las reuniones que se hicieron.
- **Excel** para la elaboración del diagrama de GANTT

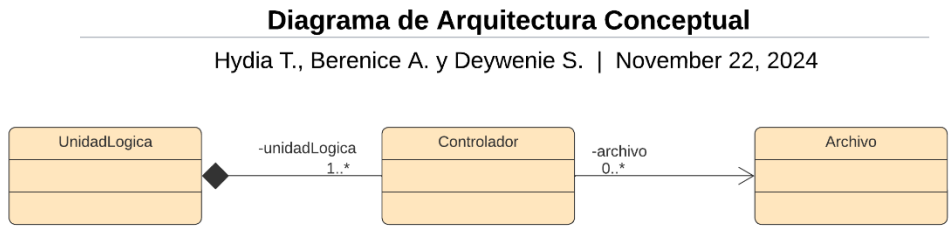
1.2 Diagrama de Gantt



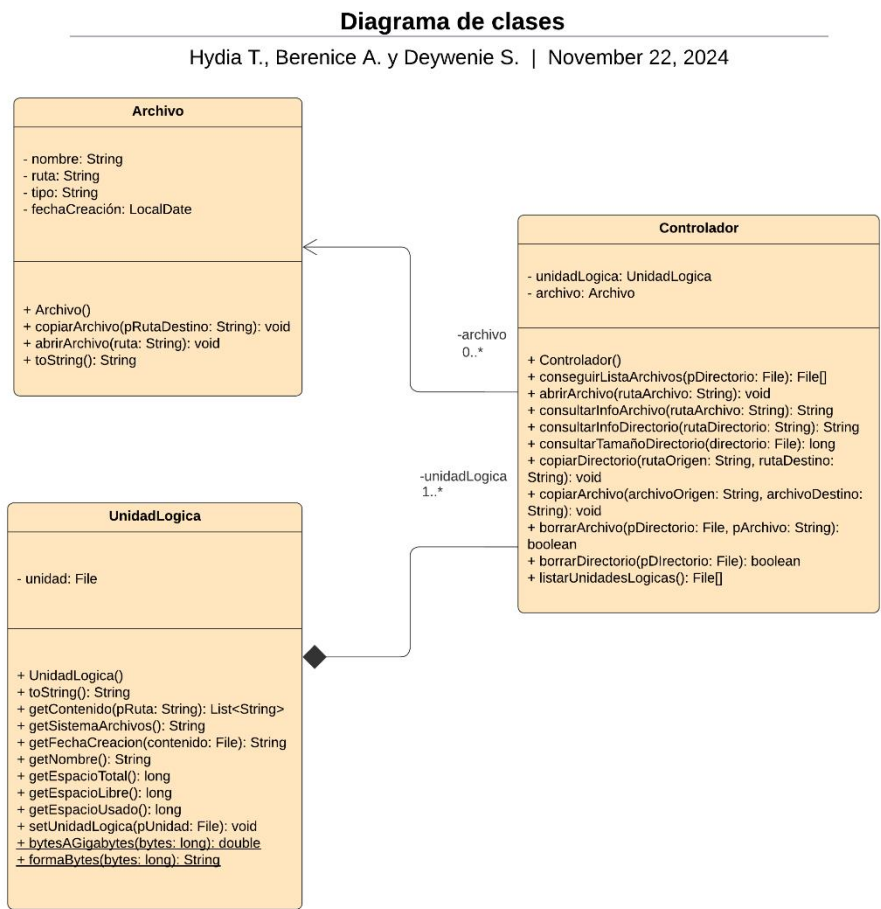
1.3 Diagrama General de casos de uso



1.4 Diagrama de arquitectura conceptual



1.5 Diagrama de clases de bajo nivel

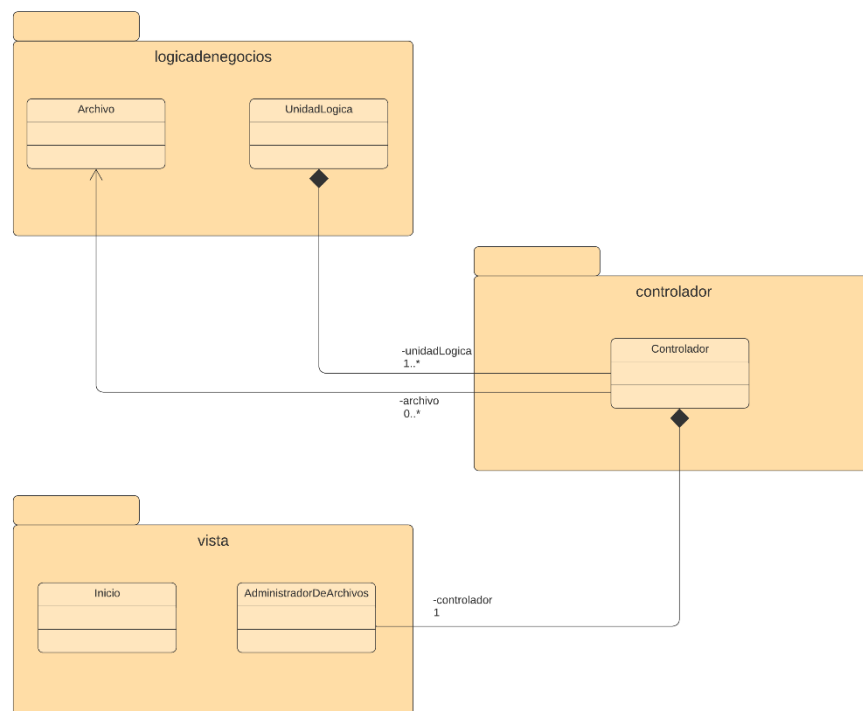


1.6 Justificación de las relaciones establecidas entre los objetos del diagrama

Las relaciones establecidas son:

- 1) Composición, entre la clase UnidadLogica y Controlador: Esto porque si no llegara a existir al menos 1 unidad lógica el programa no podría funcionar, ya que esta es la parte vital de este. En otras palabras, se podría decir que la clase UnidadLogica representa el “todo”, mientras que la clase Controlador una “parte” de esta.
- 2) Asociación, entre las clases Archivo y Controlador: La clase Archivo le proporciona distintos métodos y servicios que la clase Controlador. Pero la vida de ninguna de estas clases depende de la otra.
- 3) Composición, entre las clases Controlador y AdministradorDeArchivos, esto porque sin él la clase Controlador, AdministradorDeArchivos no podría funcionar por sí sola, ya que la vida de esta clase depende de la de Controlador.

1.7 Diagrama de paquetes



2) Análisis de Resultados

Tareas Requeridas	Estado	Justificación
Listar contenido de un directorio	100%	
Crear directorio	100%	
Consultar información de un directorio	100%	
Consultar información de un archivo	100%	
Eliminar directorio	100%	
Eliminar archivo	100%	
Copiar directorio	100%	
Copiar archivo	100%	
Abrir archivo con su aplicación padre	100%	
Consultar propiedades de una unidad lógica	100%	

3) Enlace del JavaDoc

Enlace del archivo comprimido que contiene el enlace al JavaDoc: [apidocs](#)

Descargar la carpeta y abrir "allpackages-index.html".

4) Aspectos relevantes y lecciones aprendidas

Berenice Amador:

- 1) Aprendí que es importante organizar todo antes de empezar a programar. Diseñar bien las clases, métodos y entender lo que queremos lograr ayudó a evitar problemas y a trabajar más fácil.
- 2) Usamos diferentes partes de Java, como java.io y java.nio.File, para trabajar con archivos y carpetas. Me di cuenta de que Java tiene muchas herramientas útiles, pero se necesita investigar para usarlas bien.
- 3) Hacer la parte visual del programa con Swing fue un poco complicado, pero aprendí cómo conectar lo que ve el usuario con lo que hace el programa.

- 4) Ahora sé que es muy importante validar la información que usa el programa y manejar errores correctamente. Esto hace que el programa funcione mejor y no se caiga.
- 5) Durante el proyecto tuve varios problemas, como errores en el código o diseños que no funcionaban. Esto me enseñó que programar no es solo escribir, sino probar, arreglar y seguir mejorando el proyecto.

Deywenie Smith:

- 1) Sentirme cómoda expresando mis ideas y trabajando en equipo. Viendo la importancia de comunicar mis ideas y escuchar las de los demás.
- 2) Aprendí a utilizar distintas clases en Java, como Filé, y ver cómo están facilitan el proceso de codificación. Ayudan a dividir el problema en partes más pequeñas
- 3) Cómo funciona la unidad lógica de la computadora y cómo navegar en esta. Además de cómo se organizan y se gestionan los archivos en la unidad lógica.
- 4) Los diagramas me ayudaron a gestionar mejor mi tiempo, ya que se tenían abstraídas las distintas clases y métodos a codificar.
- 5) La importancia de siempre de reunirse y revisar el trabajo final. Para así buscar errores y ayudar en el proceso de retroalimentación.

Hyldia Thomas:

- 1) Aprendí la importancia de usar el patrón MVC ya que hace que el programa sea más limpio y entendible.
- 2) Usar GitHub ayudo mucho a tener un control de versiones por si el código llegaba a fallar teníamos la opción de devolvernos a la versión anterior y continuar.
- 3) La importancia de hacer reuniones y así poder compartir los diferentes puntos de vista para llegar a una solución y poder probar el programa completo y para detectar si hay vulnerabilidades o errores.

- 4) Aprendí sobre cómo funciona la clase File y otras clases que utilizamos para elaborar el proyecto, la importancia de manejar bien la excepciones y validar bien cada detalle.
- 5) Cuando todos los miembros de un grupo se comunican de forma correcta se puede llegar a mejores acuerdos y se puede trabajar en armonía y de forma consciente.

5) Bitácora o diario de trabajo.

Enlace al repositorio utilizado: <https://github.com/Hyldia/PYO2.git>

Las bitácoras individuales:

Berenice Pereira: <https://pooproveye.blogspot.com/>

Deywenie Smith: <https://proyecto2poodeywenies.blogspot.com/2024/11/>

Hydia Thomas: <https://hyldia.blogspot.com/>

6) Bibliografía, fuentes digitales y APIs utilizadas.

Bibliografía

JavaRush. (s.f.). *Cómo copiar archivos de un directorio a otro en Java*. <https://javarush.com/es/groups/posts/es.3636.pausa-para-el-cafe-99-cmo-copiar-archivos-de-un-directorio-a-otro-en-java-cadenas-en-java>.

Kumar, M. (22 de septiembre de 2019). *Delete a File Using Java*. https://www.geeksforgeeks.org/delete-file-using-java/?ref=gcse_outind

Programming Guide. (s.f.). *List of Java Exceptions* <https://programming.guide/java/list-of-java-exceptions.html>

APIs Utilizadas

1. Java I/O API (java.io.File):

Utilizada para manejar operaciones de archivos y directorios, como listar, crear, copiar y eliminar archivos.

2. Java Swing API (javax.swing):

Empleada para desarrollar la interfaz gráfica, incluyendo componentes como JTable, JButton, y JLabel.

3. Java AWT API (java.awt):

Utilizada para manejar eventos y configurar layouts en combinación con Swing.

7) Manual de usuario

▪ Descripción del Proyecto:

El Administrador de Archivos es una aplicación de escritorio desarrollada en Java utilizando la biblioteca Swing. Su objetivo es facilitar a los usuarios la gestión de archivos y directorios a través de una interfaz gráfica (GUI). Los usuarios pueden realizar tareas comunes como visualizar archivos y carpetas, crear nuevos directorios, eliminar archivos o carpetas, navegar entre diferentes directorios y copiar archivos de manera sencilla y eficiente.

▪ Requisitos Previos:

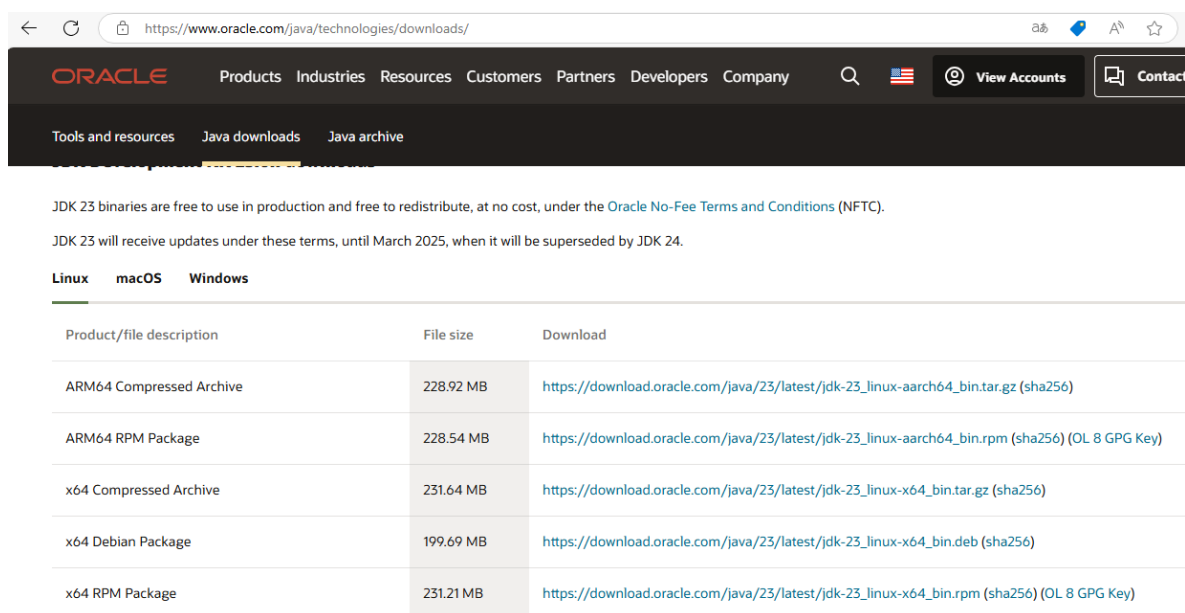
Antes de comenzar a usar el Administrador de Archivos, es necesario que tengas instalados **JDK** (Java Development Kit) y **NetBeans**, que son las herramientas necesarias para desarrollar y ejecutar aplicaciones Java.

Instalación de JDK (Java Development Kit)

El **JDK** es un conjunto de herramientas necesarias para desarrollar aplicaciones en Java. Incluye el compilador de Java (javac), la Máquina Virtual de Java (JVM).

1. Descargar el JDK:

- Ve a la página oficial de Oracle para descargar el JDK: <https://www.oracle.com/java/technologies/javase-jdk11-downloads.html> (o elige la versión más reciente disponible).
- Asegúrate de seleccionar la versión correspondiente a tu sistema operativo (Windows, macOS o Linux).



JDK 23 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 23 will receive updates under these terms, until March 2025, when it will be superseded by JDK 24.

Linux **macOS** **Windows**

Product/file description	File size	Download
ARM64 Compressed Archive	228.92 MB	https://download.oracle.com/java/23/latest/jdk-23_linux-aarch64_bin.tar.gz (sha256)
ARM64 RPM Package	228.54 MB	https://download.oracle.com/java/23/latest/jdk-23_linux-aarch64_bin.rpm (sha256) (OL 8 GPG Key)
x64 Compressed Archive	231.64 MB	https://download.oracle.com/java/23/latest/jdk-23_linux-x64_bin.tar.gz (sha256)
x64 Debian Package	199.69 MB	https://download.oracle.com/java/23/latest/jdk-23_linux-x64_bin.deb (sha256)
x64 RPM Package	231.21 MB	https://download.oracle.com/java/23/latest/jdk-23_linux-x64_bin.rpm (sha256) (OL 8 GPG Key)

2. Instalar el JDK:

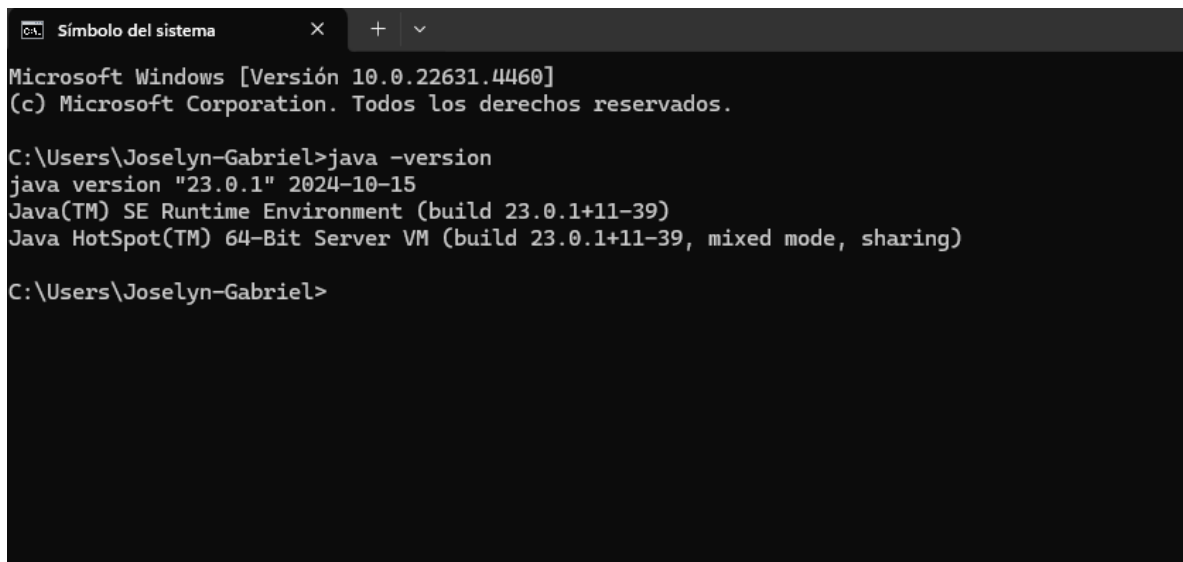
- Una vez descargado el instalador, ejecútalo y sigue las instrucciones para completar la instalación.

3. Verificar la instalación:

- Abre una terminal cmd y escribe el siguiente comando:

```
java -version
```

Si ves la versión de Java instalada, significa que la instalación fue correcta.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22631.4460]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Joselyn-Gabriel>java -version
java version "23.0.1" 2024-10-15
Java(TM) SE Runtime Environment (build 23.0.1+11-39)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.1+11-39, mixed mode, sharing)

C:\Users\Joselyn-Gabriel>
```

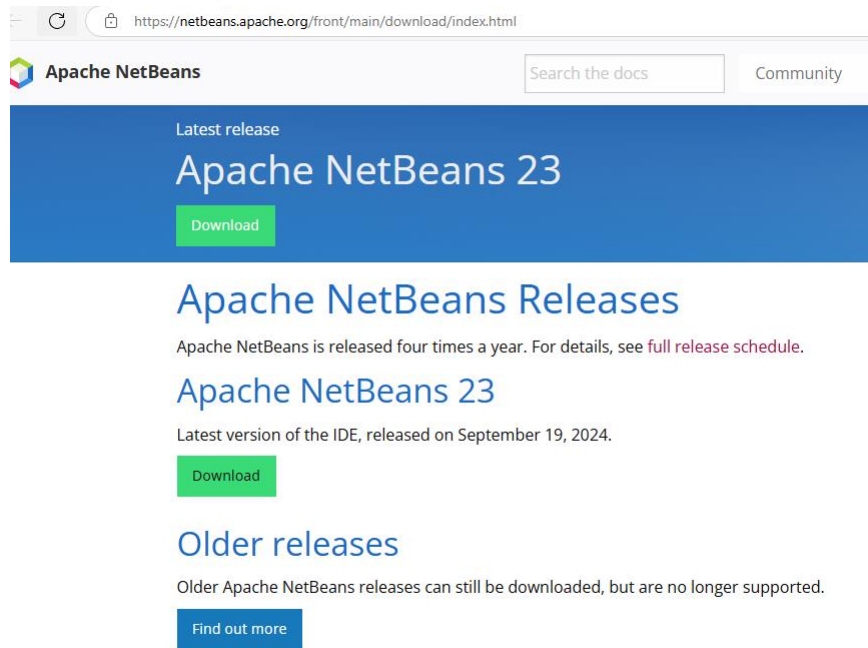
4. Instalación de NetBeans

NetBeans es un entorno de desarrollo integrado (IDE) que facilita la creación de aplicaciones Java. Puedes usarlo para escribir, depurar y ejecutar el código de tu aplicación.

Pasos para instalar NetBeans:

1. Descargar NetBeans:

- Dirígete al sitio oficial de NetBeans:
<https://netbeans.apache.org/download/index.html>.
- Descarga la versión que sea compatible con tu sistema operativo.



2. Instalar NetBeans:

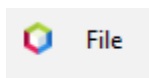
- Ejecuta el instalador descargado y sigue las instrucciones para completar la instalación.

5. Ejecución del Administrador de Archivos

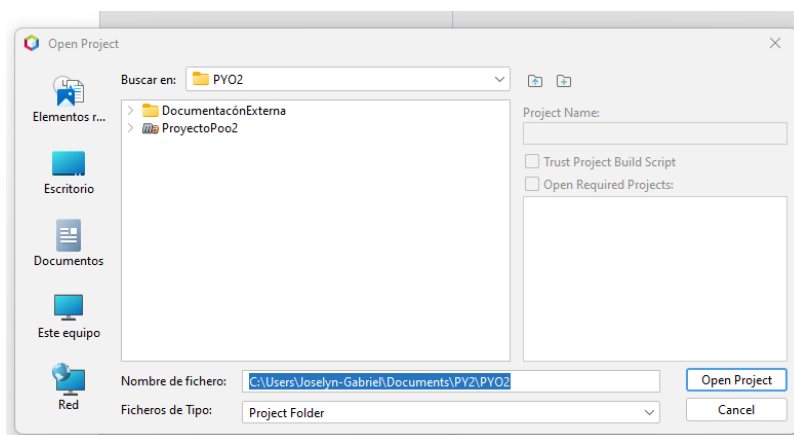
Una vez que hayas instalado tanto el **JDK** como **NetBeans**, podrás ejecutar y desarrollar el proyecto Administrador de Archivos.

1. Abrir el proyecto en NetBeans:

- Abre NetBeans y selecciona Archivo > Abrir Proyecto.

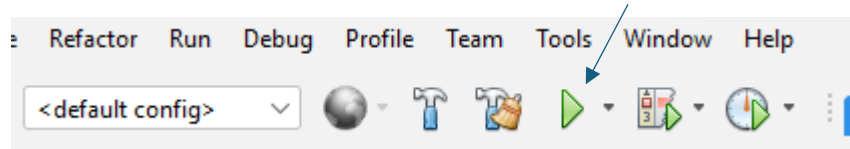


- Navega hasta la carpeta donde descargaste o guardaste el código fuente del proyecto y selecciona la carpeta del proyecto Administrador de Archivos.



2. Ejecutar el proyecto:

- Una vez abierto el proyecto en NetBeans, haz clic sobre el botón verde.

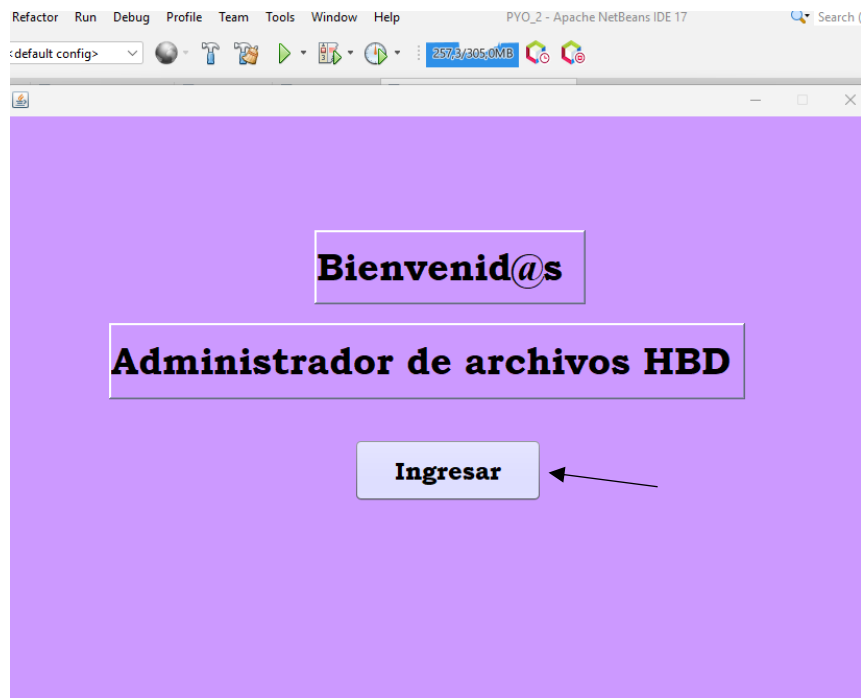


- Deberías ver la interfaz gráfica del **Administrador de Archivos** en tu pantalla.

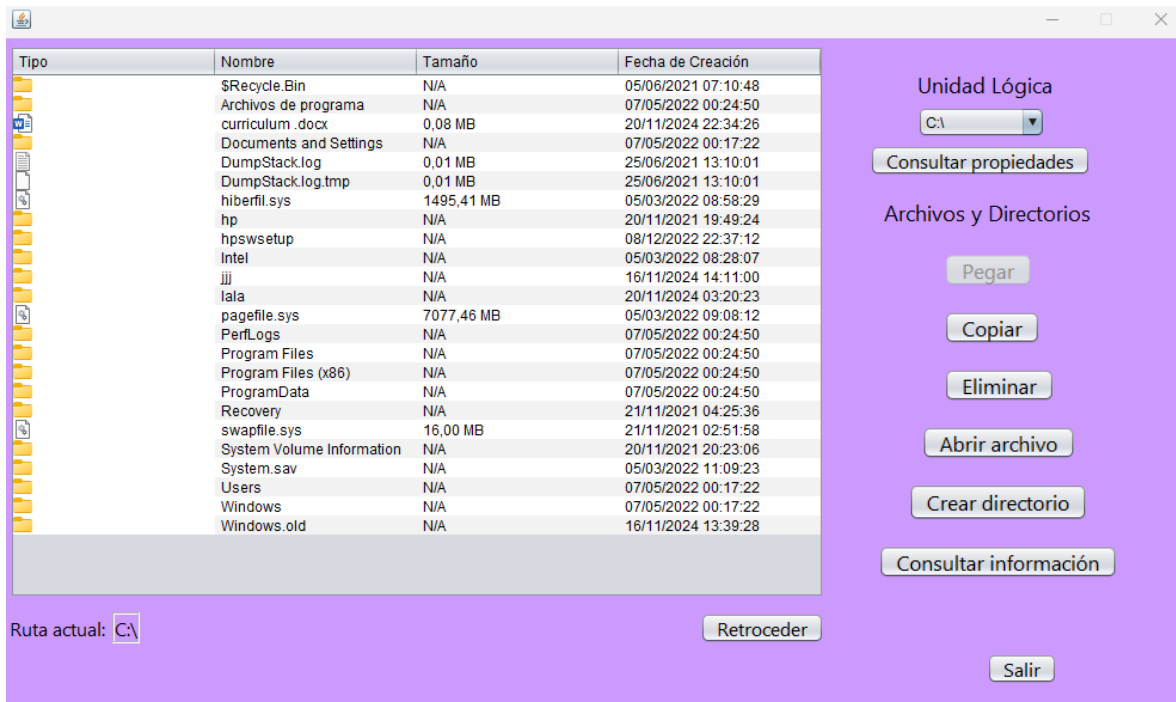
- **Interfaz de Usuario**

La interfaz gráfica de usuario (GUI) está compuesta por los siguientes elementos:

- **Ventana Bienvenida:** Contiene un botón para ingresar al Administrador de Archivos.
- **Tabla de Archivos:** Muestra los archivos y directorios en la ubicación actual.
- **Panel de Herramientas:** Contiene los botones para realizar acciones sobre los archivos y directorios.
- **Etiqueta de Ruta:** Muestra la ubicación actual del directorio donde el usuario se encuentra.



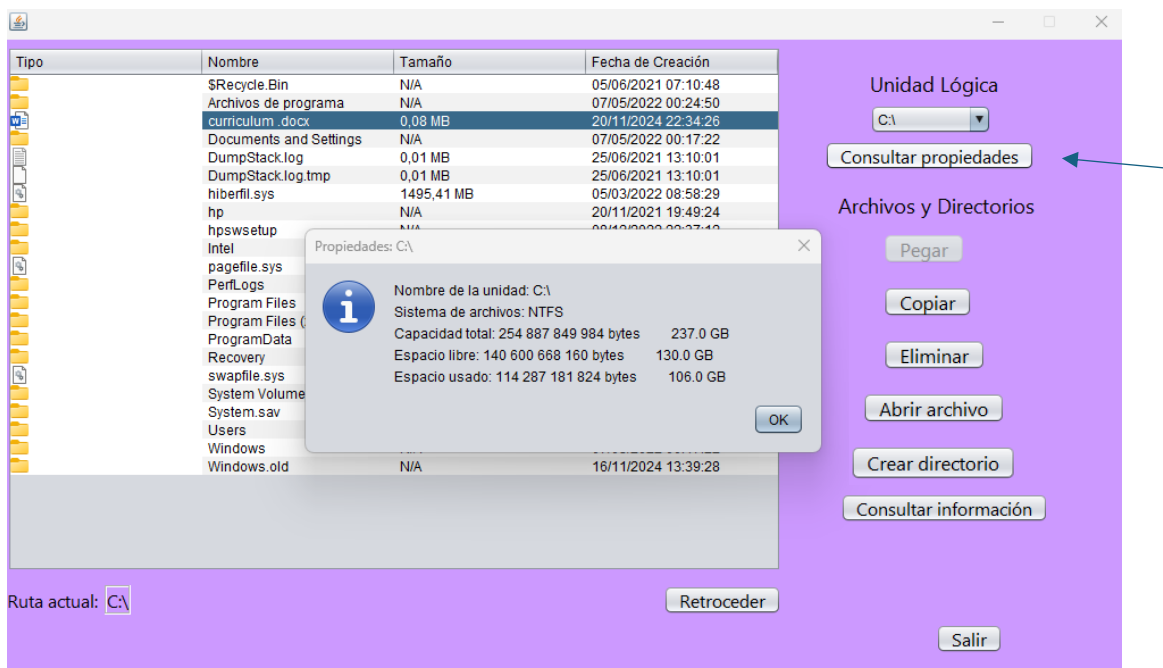
- Una vez en la interfaz de inicio haz clic sobre el botón ingresar.



- **Funcionalidades**

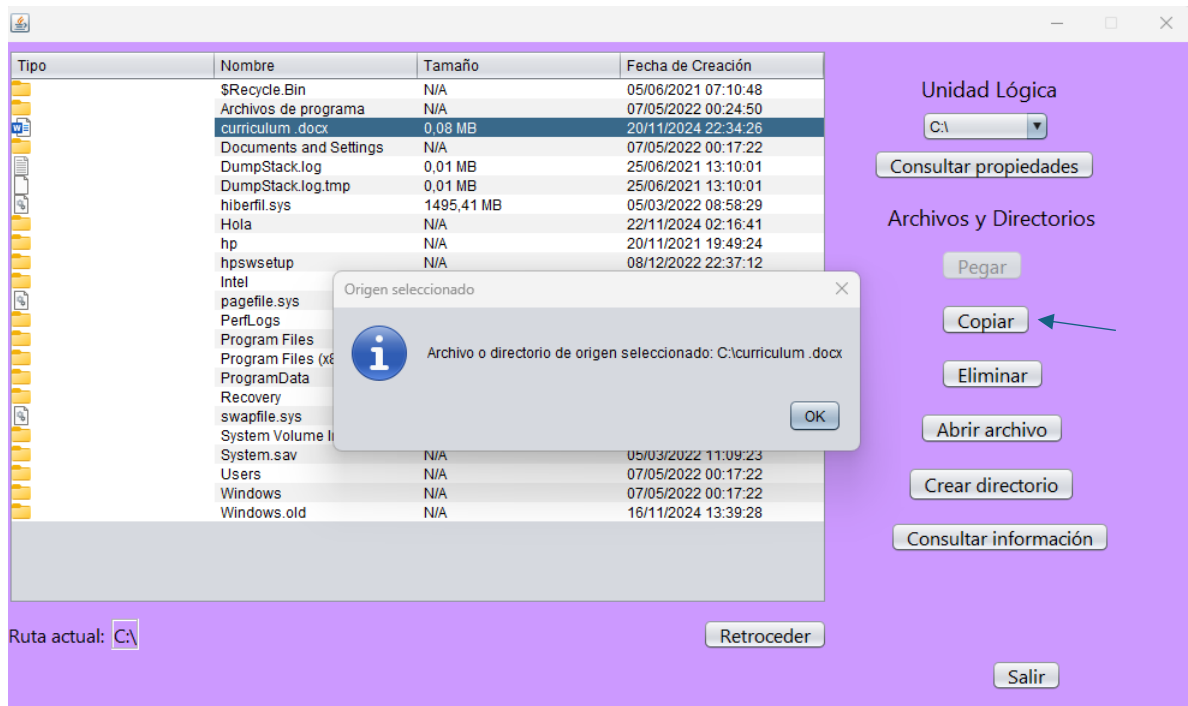
- **Consultar Propiedades.**

- Para consultar las propiedades de la unidad Lógica, haz clic sobre el botón “consultar propiedades”.

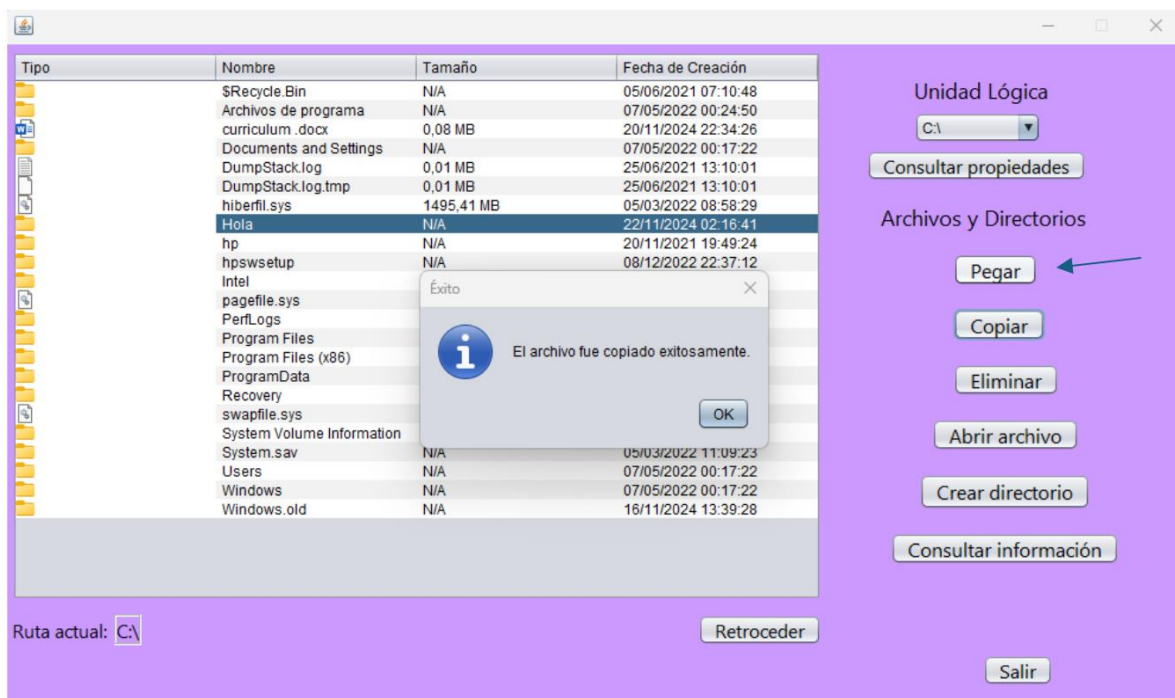


➤ Copiar archivos o directorios

- Selecciona el archivo o directorio que deseas copiar.
- Haz clic en el botón **"Copiar"** para habilitar la opción de copiar el archivo o directorio, a otro directorio.

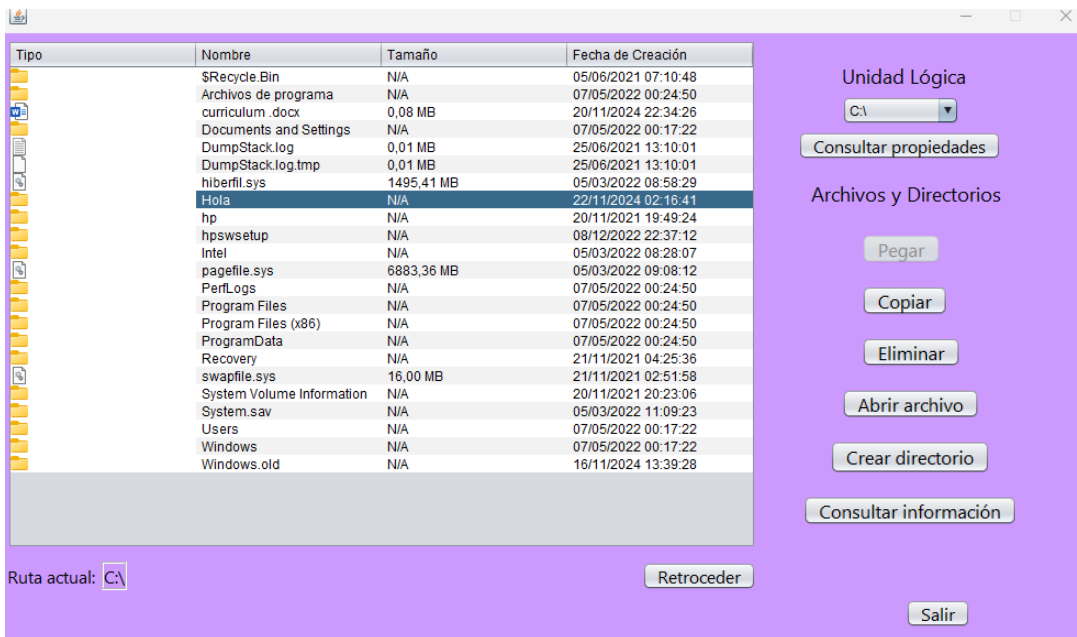


- Selecciona el directorio al que deseas copiar el archivo o directorio.
- Haz clic en el botón **"Pegar"**

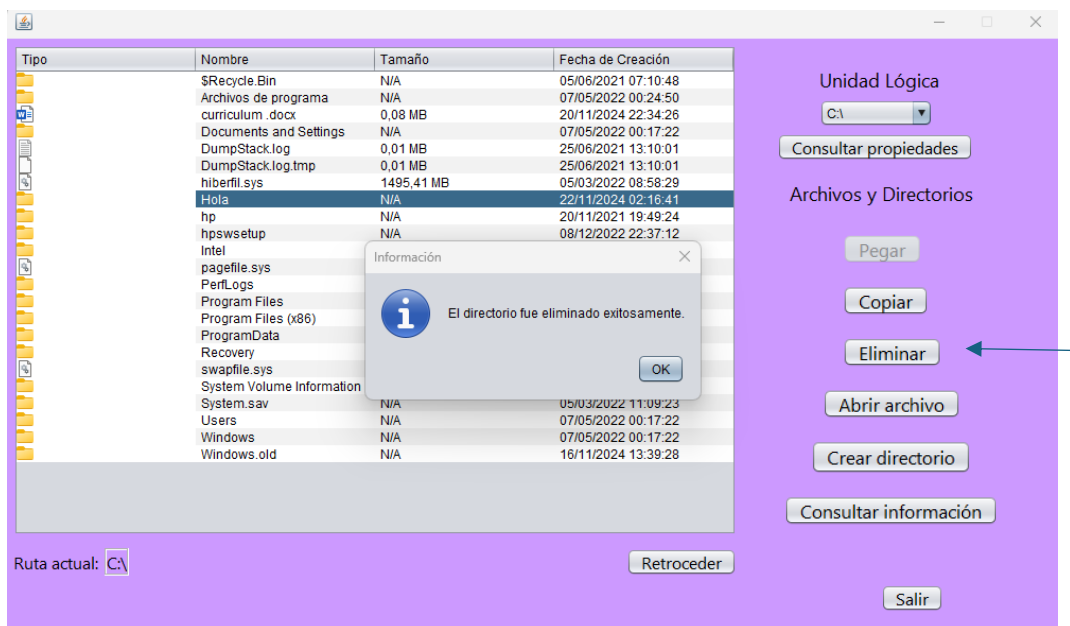


➤ Eliminar archivos o directorios

- Selecciona el archivo o directorio que deseas eliminar.



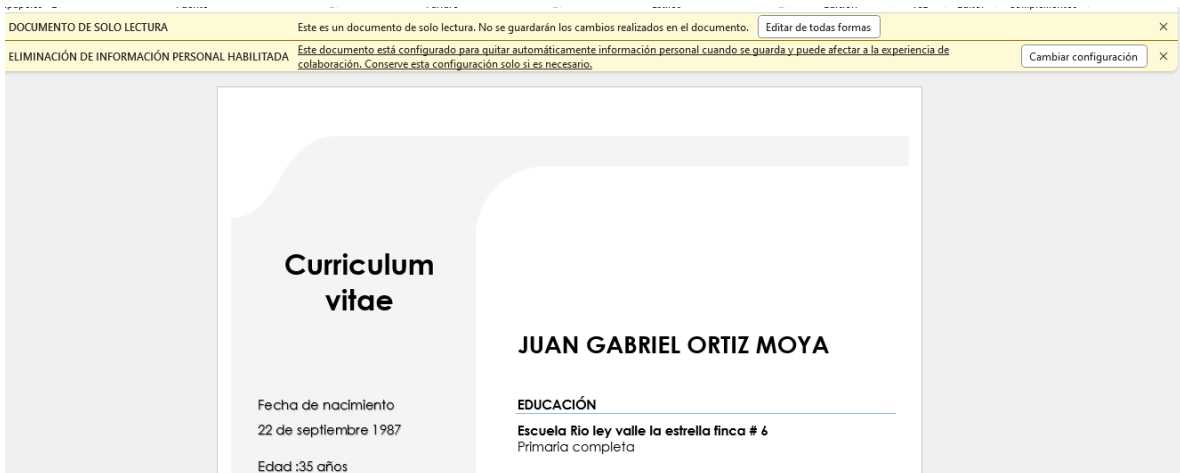
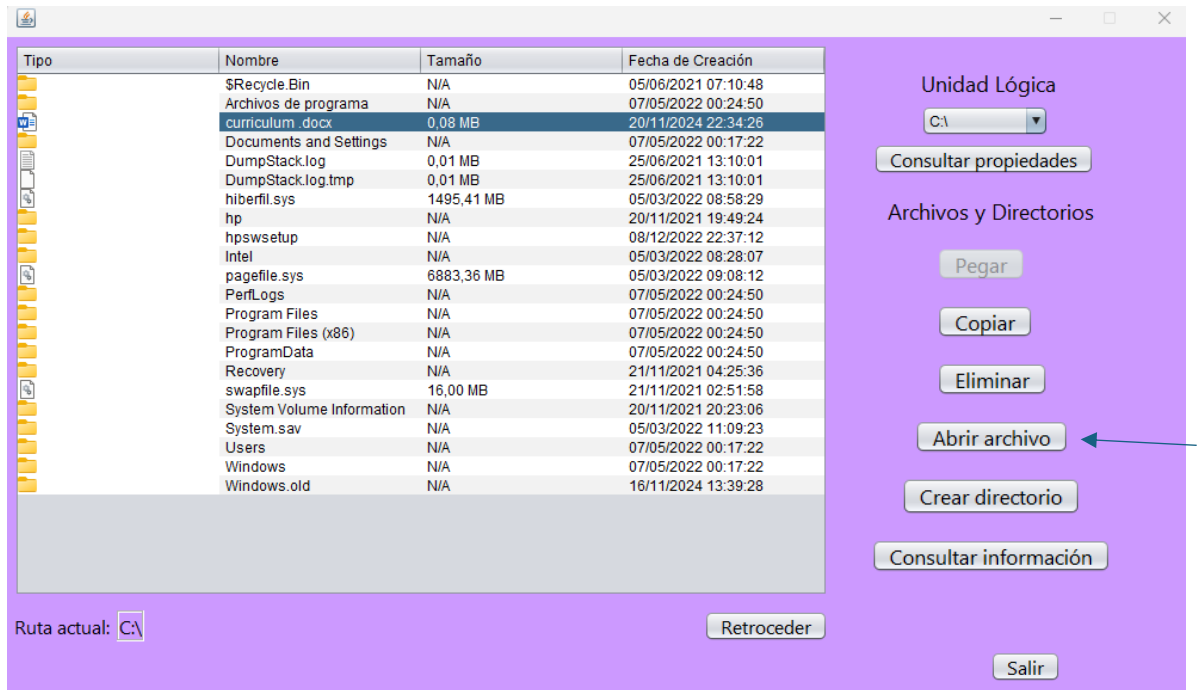
- Haz clic en el botón "Eliminar".



- La aplicación eliminará el archivo o directorio seleccionado. Si es un directorio, todos los archivos dentro de él también se eliminarán.

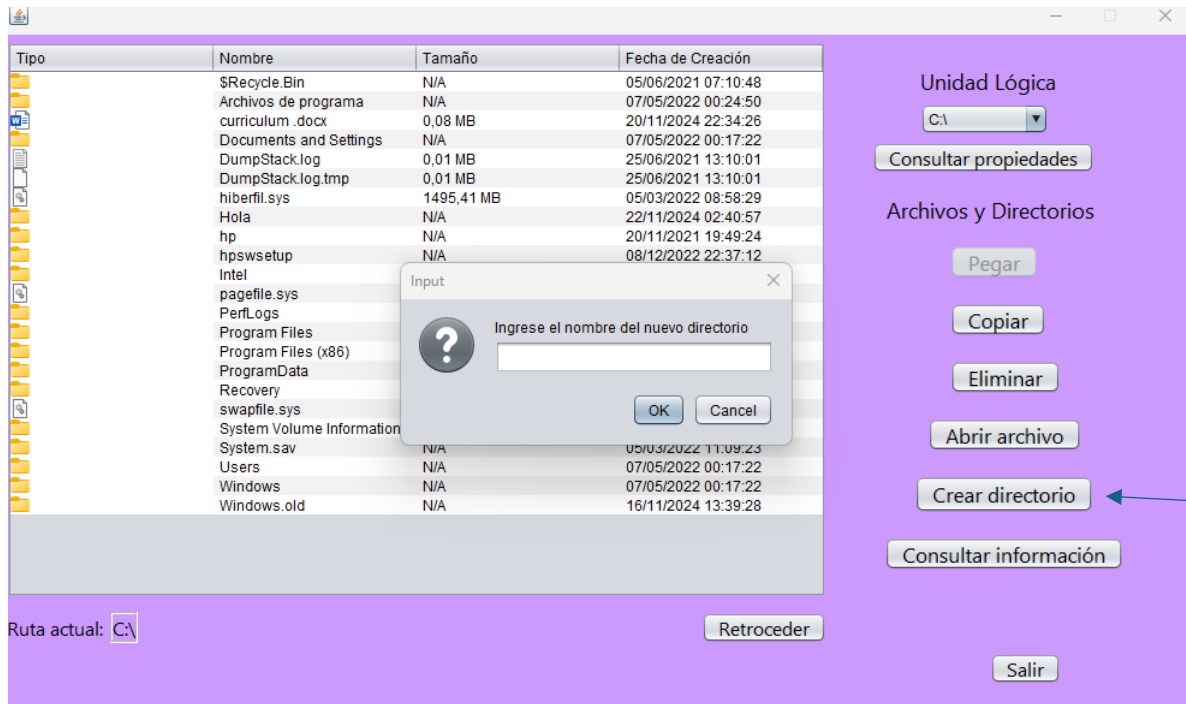
➤ Abrir Archivo

- Para abrir un archivo selecciona el archivo a abrir.
- Haz clic sobre el botón “**Abrir archivo**”.

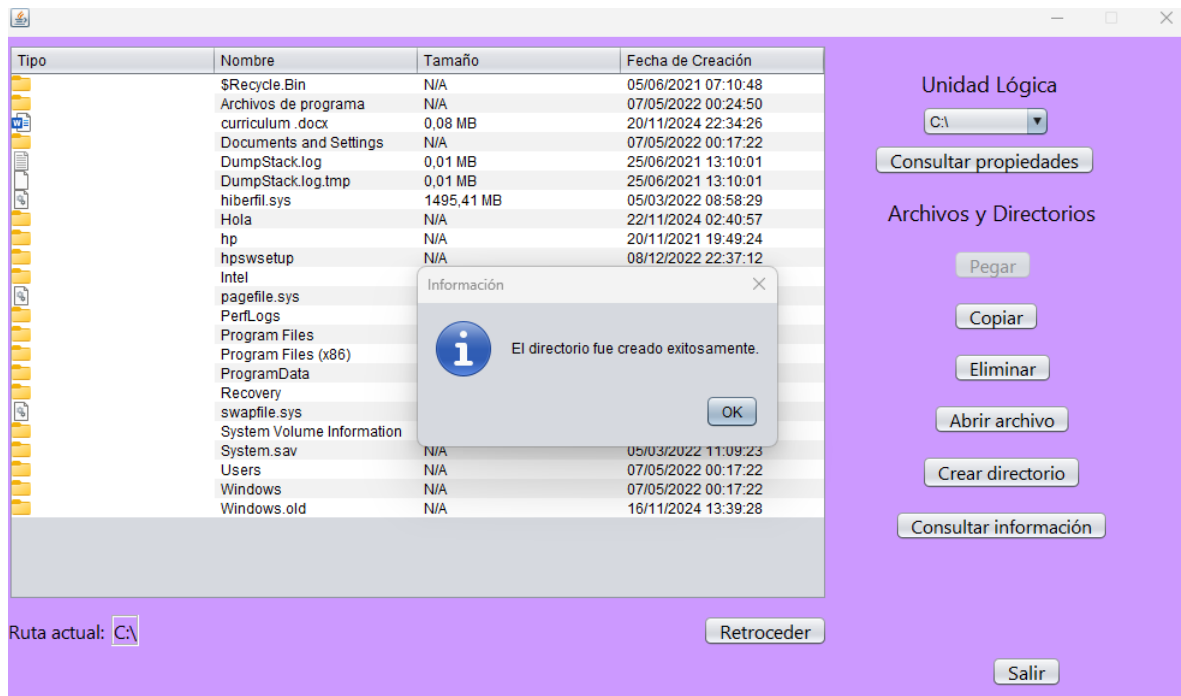


➤ Crear un nuevo directorio

- Haz clic en el botón "Crear Directorio".
- Ingrese el nombre del nuevo directorio en el cuadro de diálogo, no puede dejar en blanco el espacio, ni dejar solo espacios vacíos.

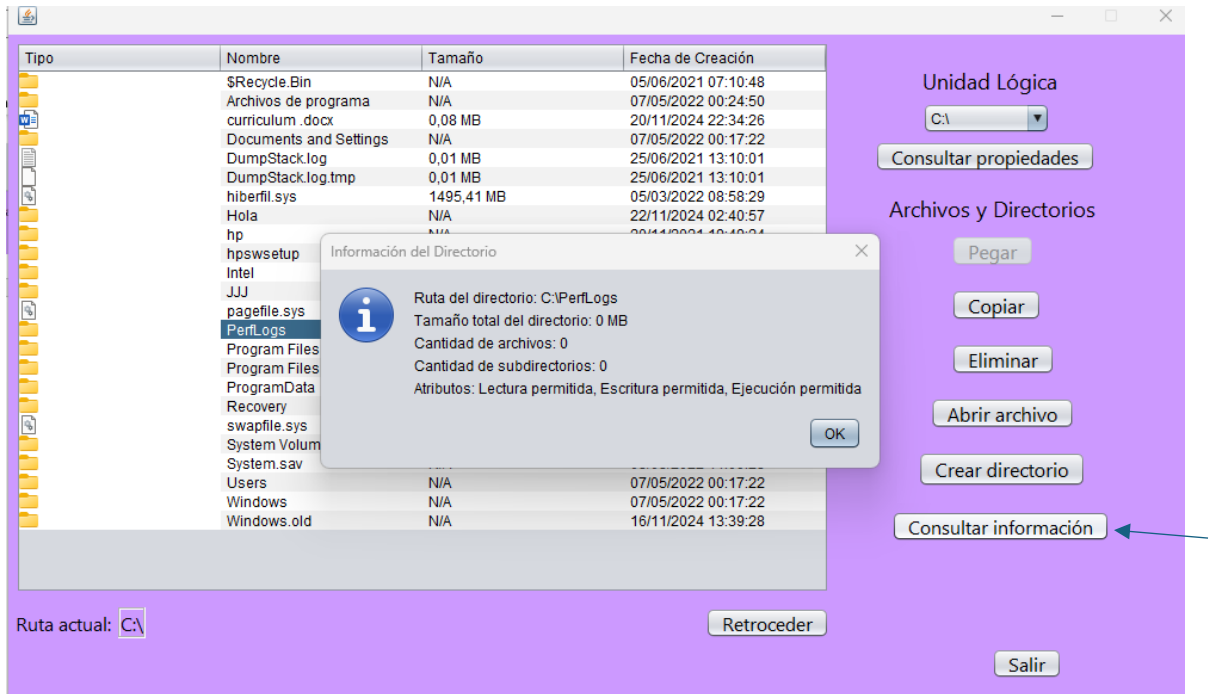


- Si el nombre es válido, el directorio se creará en la ubicación actual. Si ya existe un directorio con ese nombre, recibirás un mensaje de advertencia.



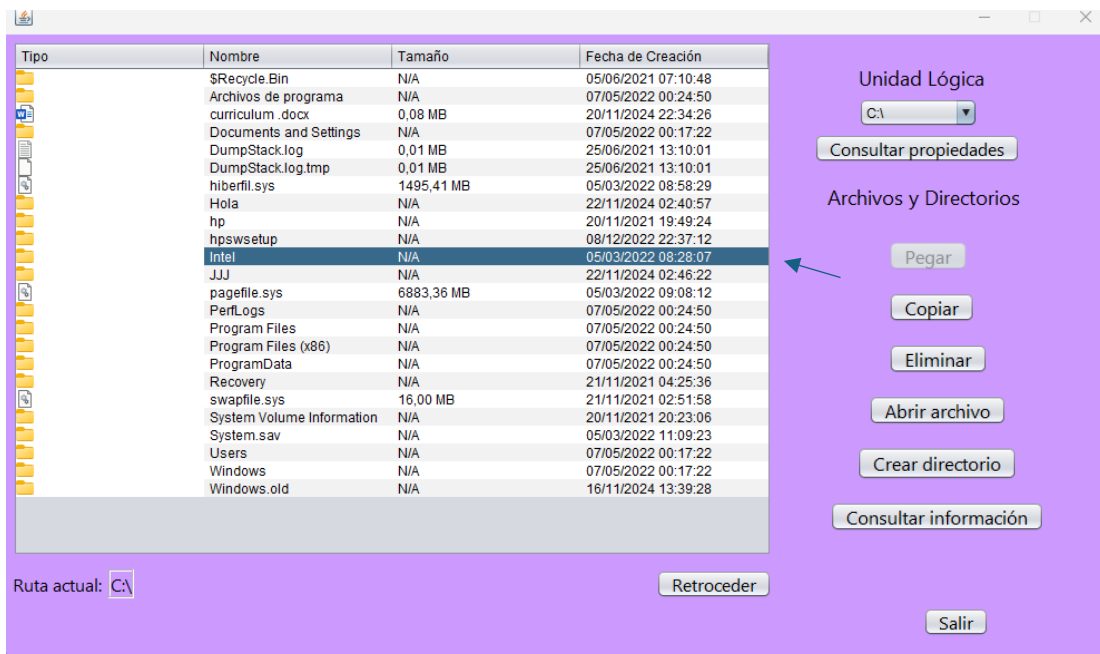
➤ Consultar Información

- Selecciona el archivo o directorio a consultar información
- Haz clic sobre el botón **“Consultar”**.

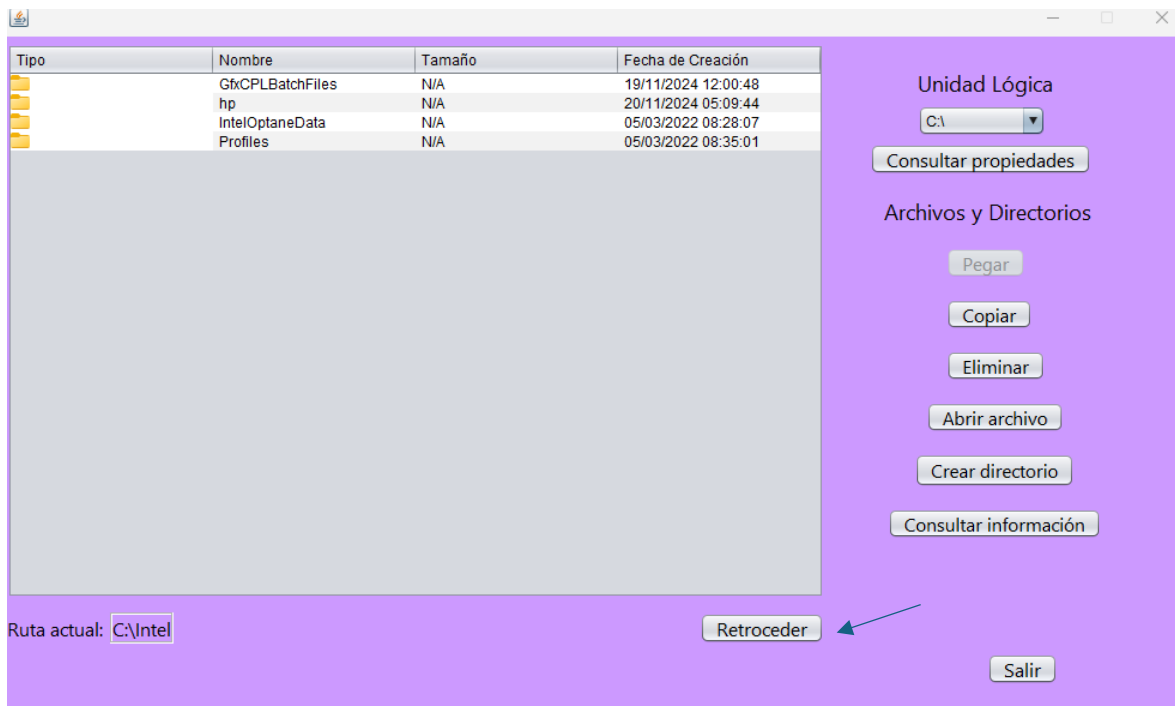


➤ Retroceder

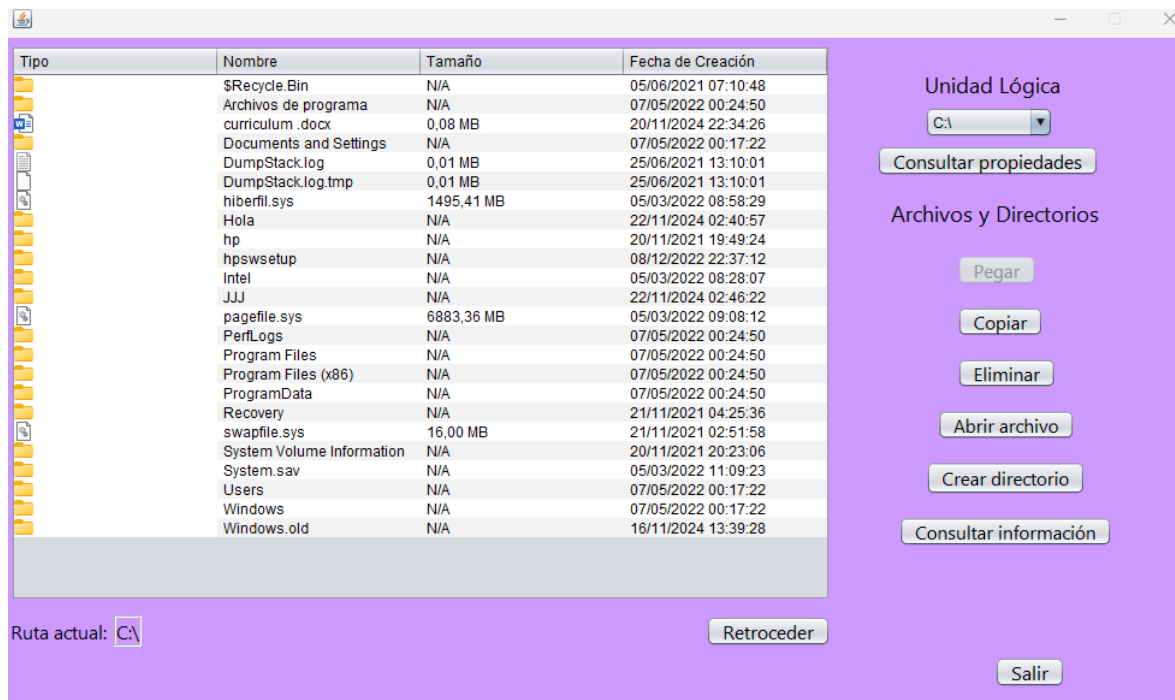
- Seleccione un directorio
- Haz doble clic sobre el directorio



- Una vez dentro del directorio presione el botón de “**Regresar**”

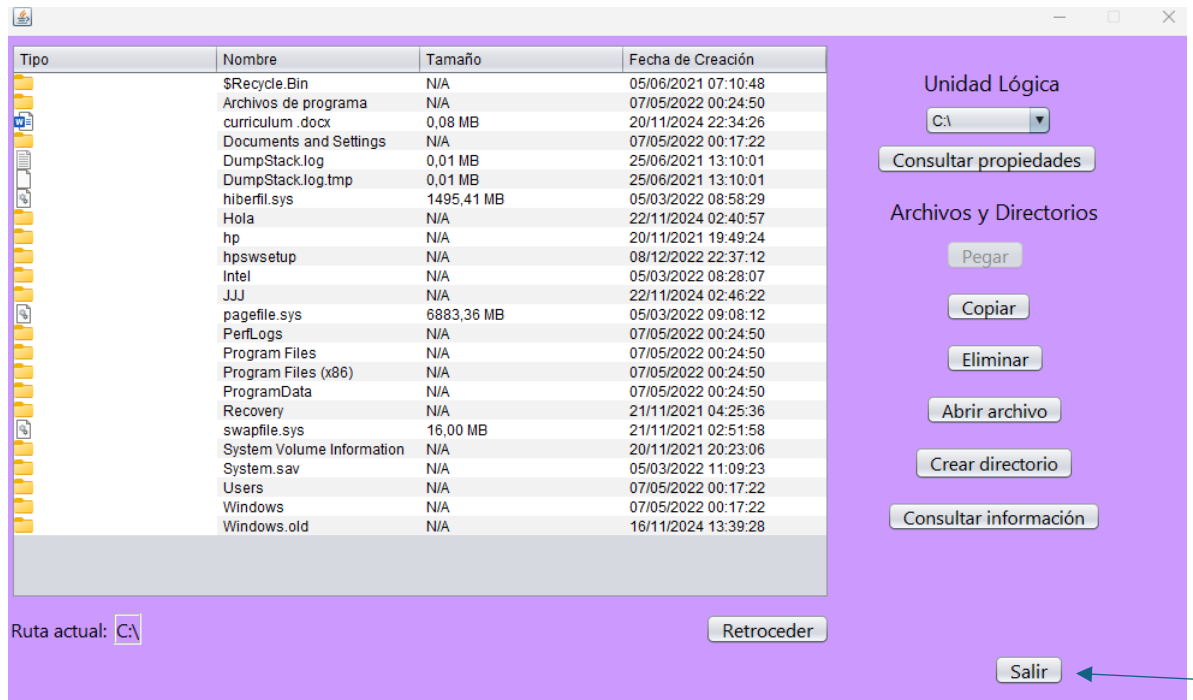


- Regresara a la carpeta anterior



➤ Salir

- Para salir del administrador de archivos
- Haz clic sobre el botón **“Salir”**



➤ Consejos y Recomendaciones

- **Nombres de Directorios**
- Evita caracteres especiales: No uses caracteres como *,:, ?, /, \, <, >, o | en los nombres, ya que pueden causar errores en el sistema operativo.
- Evita espacios al inicio o final: Estos pueden generar problemas de acceso en algunos sistemas.
- Considera usar guiones bajos o medios: En lugar de espacios, usa _ o - para mayor compatibilidad.
- **Evita Eliminar Archivos por Error**
- Lee detenidamente el nombre del archivo o directorio antes de eliminarlo.