DOCUMENTACIÓN

Autor: Mario Rodríguez Abal

Módulo: Programación de Servicios y Procesos – 2º DAM

Centro: IES Armando Cotarelo Valledor

Profesor: Daniel Resúa Melón

Fecha de entrega: 09 / 11 / 2025

INDICE

1. Introducción

1.1. Objetivo

El proyecto **Monitor de Procesos** es una aplicación de escritorio desarrollada en **Kotlin + Compose for Desktop** cuyo objetivo es **listar los procesos del sistema** y ofrecer al usuario una visión clara y operativa de su estado: **PID, nombre, usuario, uso de CPU/MEM**, **estado**, comando asociado y acciones habituales (p. ej., exportación de datos). La herramienta prioriza **usabilidad, portabilidad** y **mantenibilidad** a través de una arquitectura por capas (UI → Dominio → Proveedores OS) y empaquetado nativo para Windows y Linux.

Repositorio público del proyecto: https://github.com/Rguez00/KotlinProject

1.2. Alcance

Incluye:

- Interfaz de escritorio en Compose: tabla de procesos con ordenación por cabecera,
- filtrado y auto-refresco.
- Métricas básicas por proceso (CPU/MEM) y detalle de proceso seleccionado.
- Exportación a CSV de la vista activa (respeta filtros/ordenación).
- Empaquetado y guía de instalación para Windows y Linux.
- Anexos con capturas (reales o simuladas cuando falten), y Anexo de uso de IA con prompts/respuestas resumidos.

No incluye (fuera del alcance):

- Monitorización remota/multiequipo.
- Gráficas históricas avanzadas o alertado (se consideran trabajo futuro).
- Gestión de privilegios/sandboxing específicos por SO.

2. Requisitos

2.1. Funcionales

RF-1. Listado de procesos del sistema con columnas: PID, Proceso, Usuario, CPU, MEM, Estado, CMD.

RF-2. Ordenación por cabecera (asc/desc) al hacer clic.

RF-3. Filtro por texto/usuario/estado.

RF-4. Auto-refresco configurable.

RF-5. Detalle de un proceso seleccionado.

RF-6. Exportación a CSV de la vista actual.

2.2. No funcionales

RNF-1. Usabilidad y accesibilidad. Diseño limpio con contraste adecuado en estados de selección/ordenación.

RNF-2. Portabilidad. Empaquetado nativo para **Windows** y **Linux** (ver capítulo de despliegue).

RNF-3. Rendimiento. Equilibrio entre frecuencia de refresco y uso de CPU (recomendado ≥ 1000–2000 ms en equipos modestos).

RNF-4. Mantenibilidad. Separación UI/Dominio/Proveedores OS y contratos claros.

RNF-5. Reproducibilidad. Instrucciones de build y empaquetado en README.

3. Estructura del repositorio y metodología

El repositorio se organiza en módulos que facilitan la separación de responsabilidades:

- **composeApp/**: aplicación de escritorio y capa de UI (Compose).
- **core-domain/**: contratos y lógica de dominio (use cases, modelos).

- **core-data/**: **proveedores específicos por SO** (Windows/Linux) para obtención de procesos y métricas.
- Archivos de **construcción Gradle** en la raíz (Kotlin DSL).

Esta estructura puede revisarse en el repositorio publicado. GitHub

En cuanto a metodología, se ha trabajado de forma incremental: primero **diseño de interfaz**, luego **integración de datos por SO**, más tarde **rendimiento y autofresco**, y finalmente **empaquetado** y **documentación**, manteniendo un **historial de commits** coherente y referenciando el repositorio público en esta memoria (véase README y anexos).

4. Supuestos y dependencias

- **JDK** empleado: **17**.
- **Sistemas probados**: **Windows 10, Ubuntu 24.04**.
- **Licencia** del proyecto: **GPL-3.0**.

La obtención de CPU/MEM por proceso depende de **interfaces del SO** (WMI/contadores en Windows; ps//proc en Linux) y puede requerir permisos adecuados (se detalla en capítulos 4, 6 y Anexos).

5. Tecnologías usadas y diseño de la arquitectura del sistema

5.1. Stack y dependencias

- **Lenguaje y runtime**: **Kotlin/JVM** (Gradle Kotlin DSL).
- **UI de escritorio**: **Compose for Desktop** (Compose Multiplatform – Desktop). Permite construir ventanas, menús, scrollbars y distribuir binarios nativos/instaladores desde Gradle.

- **JDK y métricas globales**: OperatingSystemMXBean vía ManagementFactory para obtener información del sistema operativo cuando proceda. Versiones modernas documentadas en Oracle (8/17/23).
- **Acceso a datos por SO (providers)**:
    - **Windows**: contadores de rendimiento vía **PowerShell Get-Counter** y la herramienta **typeperf** (PDH).
    - **Linux**: comandos estándar **ps** y acceso al **pseudo-FS /proc**.
- **Distribución**: empaquetado nativo con **Compose Native Distributions** (createDistributable, nativeDistributions → DEB/RPM/EXE/ZIP).

Repositorio: composeApp/ (UI/app), core-domain/ (contratos/uso de casos), core-data/ (providers SO).

5.2. Diseño lógico

```
flowchart TD
  UI[Compose UI] --> VM[State / ViewModel]
  VM --> DOMAIN[Domain / UseCases]
  DOMAIN --> PROVIDERS[OS Providers]
  PROVIDERS -->|Windows| WIN[Get-Counter / typeperf]
  PROVIDERS -->|Linux| LNX[ps / /proc]
  DOMAIN --> CSV[Export CSV]
```

Figura 1. Arquitectura lógica por capas y puntos de integración por SO. (Fuente: propia; referencias Compose/JDK/Windows/Linux en 4.1).

Descripción rápida de responsabilidades:

- **UI (Compose UI):** pantalla principal, tabla de procesos, controles de filtro/ordenación/auto-refresco, panel de detalles y acciones (Exportar CSV).
- **State/ViewModel:** estado observable (lista, orden actual, filtro, selección), orquestación de refrescos y side-effects (lectura providers, exportación).

- Dominio (UseCases/contratos): puertos/Interfaces (p. ej., ProcessProvider, SystemInfoProvider), DTOs (PID, nombre, usuario, CPU%, MEM%, estado, cmd), políticas (ordenación/filtrado), *throttling/debounce* del refresco.

- Providers OS:

- Windows: adaptadores para consultar contadores PDH (vía Get-Counter/typeperf) y listar procesos.

- Linux: adaptadores a ps (instantánea) y a /proc (lectura detallada por PID).

Infra/Export: serialización de la tabla visible a CSV, respetando orden y filtros activos.

5.3. Decisiones clave

- Compose Desktop como framework de UI

Permite UI moderna (Material-like), eventos y componentes de escritorio (ventanas, menús, scrollbars).

Distribución nativa multiplataforma con Gradle (instaladores/paquetes).

Aislamos UI de la obtención de datos con ViewModel/estado para pruebas y mantenibilidad.

- Providers por sistema operativo (puertos/adaptadores)

- Windows: los Performance Counters exponen CPU/Memoria/Disco con una interfaz estable; Get-Counter (PowerShell) y typeperf (CLI) son vías soportadas oficialmente.

- Linux: ps ofrece instantánea portable de procesos; /proc expone detalles por PID (tiempos, memoria, cmdline) y es la base de herramientas como ps/top.

- Separación por interfaz ProcessProvider minimiza *ifdefs* y facilita pruebas unitarias por simulación.

- Métricas globales y del host vía JDK

OperatingSystemMXBean / UnixOperatingSystemMXBean aportan datos del SO/host sin depender del shell cuando es suficiente (carga media, memoria, etc.).

- Empaquetado y entrega

Uso de Compose Native Distributions para .deb/.rpm (Linux) y .exe/.zip (Windows); se integra con jlink/runtime recortado cuando procede.

5.4. Modelo de datos y contratos

- **ProcessInfo**

pid: Int | Long, name: String, user: String?, cpuPct: Double?, memPct: Double?, state: ProcState, cmd: String?.

- **ProcessProvider**
    - listProcesses(filtros, orden, limit?) : List<ProcessInfo>
    - getDetails(pid) : ProcessDetails?
    - kill(pid, graceful: Boolean = true) : Result
- **SystemInfoProvider** (opcional/global)
    - cpuLoad(): Double?, memUsedPct(): Double?, uptime(): Duration?

Notas de implementación: el cálculo de **CPU% por proceso** suele requerir muestrear dos lecturas (delta tiempo de CPU / delta muro), y puede variar por SO; cuando no sea fiable, se muestra **N/D** o un valor suavizado. **MEM%** puede provenir de RSS/Working Set relativo a MemTotal del host.

5.5. Diseño de actualización y rendimiento

- **Auto-refresco configurable** (p. ej., 1000–2000 ms recomendado para equipos modestos).
- **Throttling** de consultas: si el usuario cambia filtros/orden rápidamente, se agrupan eventos para evitar ráfagas de llamadas.
- **Actualización por diffs** en estado (sólo mutar lo que cambió) para mantener la UI fluida.
- **Estrategia por SO**:
    - **Windows**: consultas selectivas a contadores; typeperf permite **intervalo** y **salida CSV** si se requieren series para validar.
    - **Linux**: ps para snapshot + /proc para detalles concretos bajo demanda.

5.6. Tratamiento de errores y permisos

- **Permisos insuficientes** (procesos de otros usuarios o métricas privilegiadas): mostrar aviso no intrusivo y degradar la información (p. ej., ocultar user o cmdline si el SO lo restringe).
- **Herramientas no disponibles**:
    - **Windows**: Get-Counter requiere Windows; typeperf viene con el sistema. Detectar y notificar si el PATH/PowerShell no está disponible.
    - **Linux**: si falta ps (inusual) o /proc no está montado (contenedores mínimos), informar de funcionalidad limitada.
- **Kill seguro**: la acción se expone como **opcional** y requiere confirmación (y permisos). En entornos académicos, por defecto **deshabilitada** o limitada a procesos del usuario actual.

6. Descripción de funcionalidades

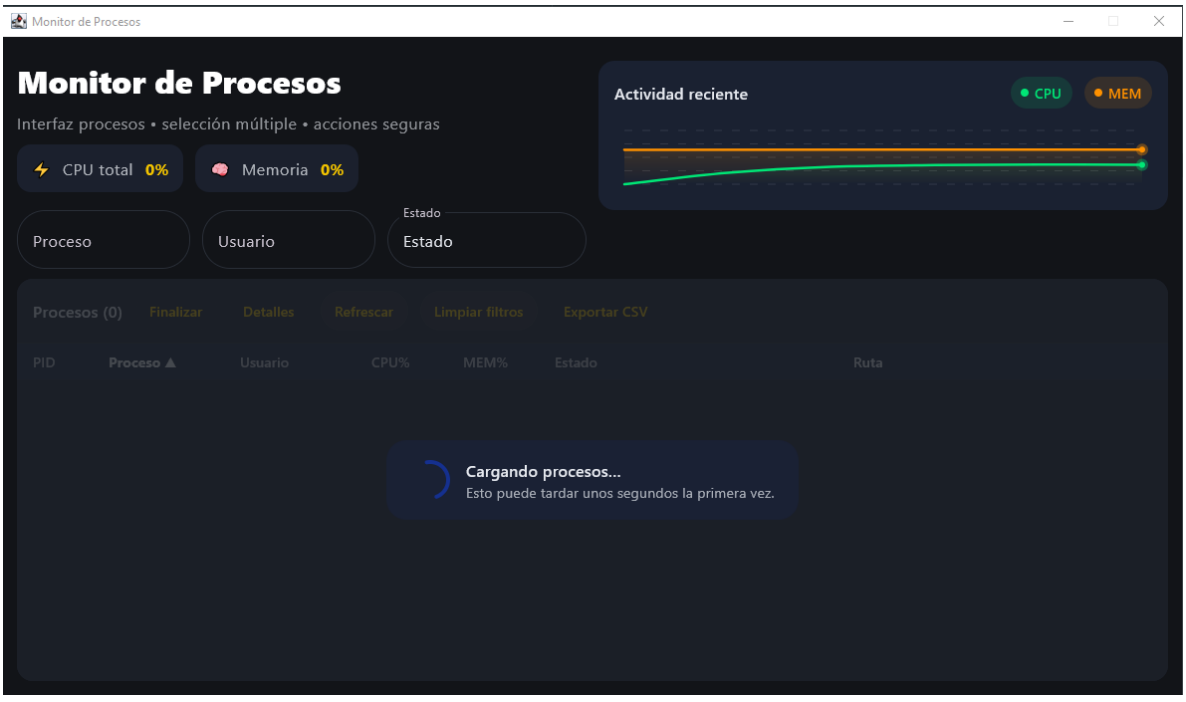

Figura 2. Interfaz del programa al momento de iniciarse. Nos muestra una ventana de alerta que nos indica que el programa está procesando.

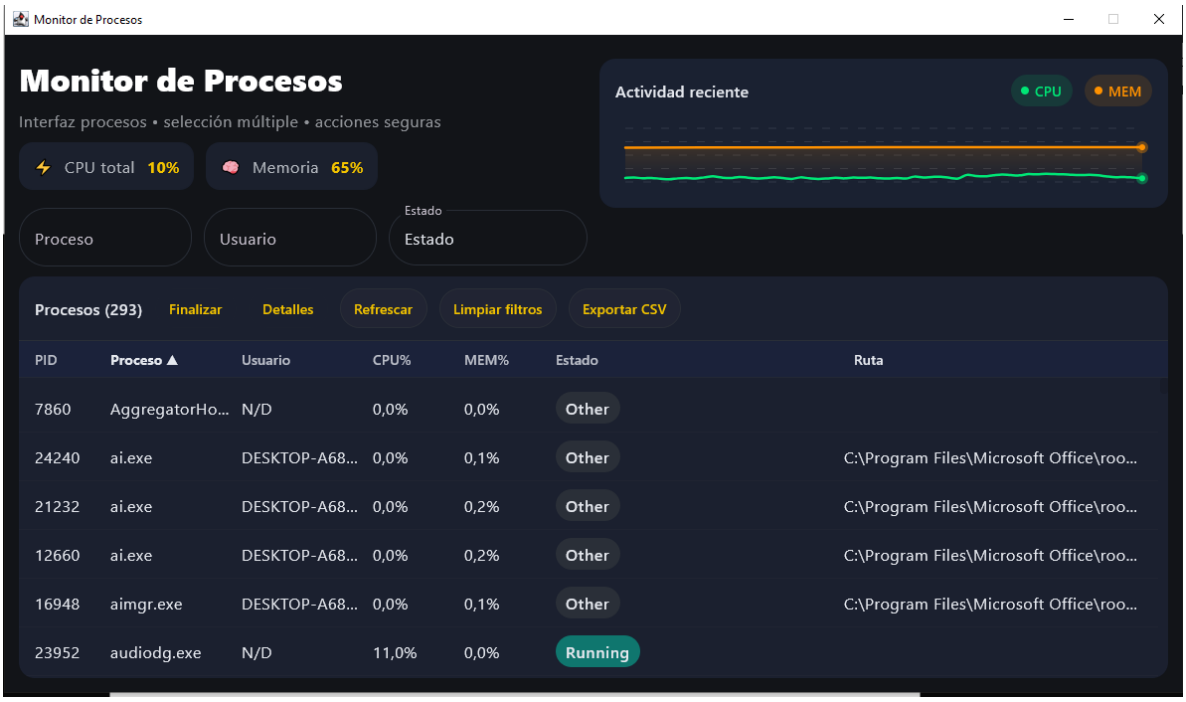

Figura 3. Una vez cargados los procesos los lista. En la parte superior también tenemos información de uso de CPU y Memoria, pero siendo este el uso total. La gráfica de la derecha hace referencia a la CPU total y Memoria.

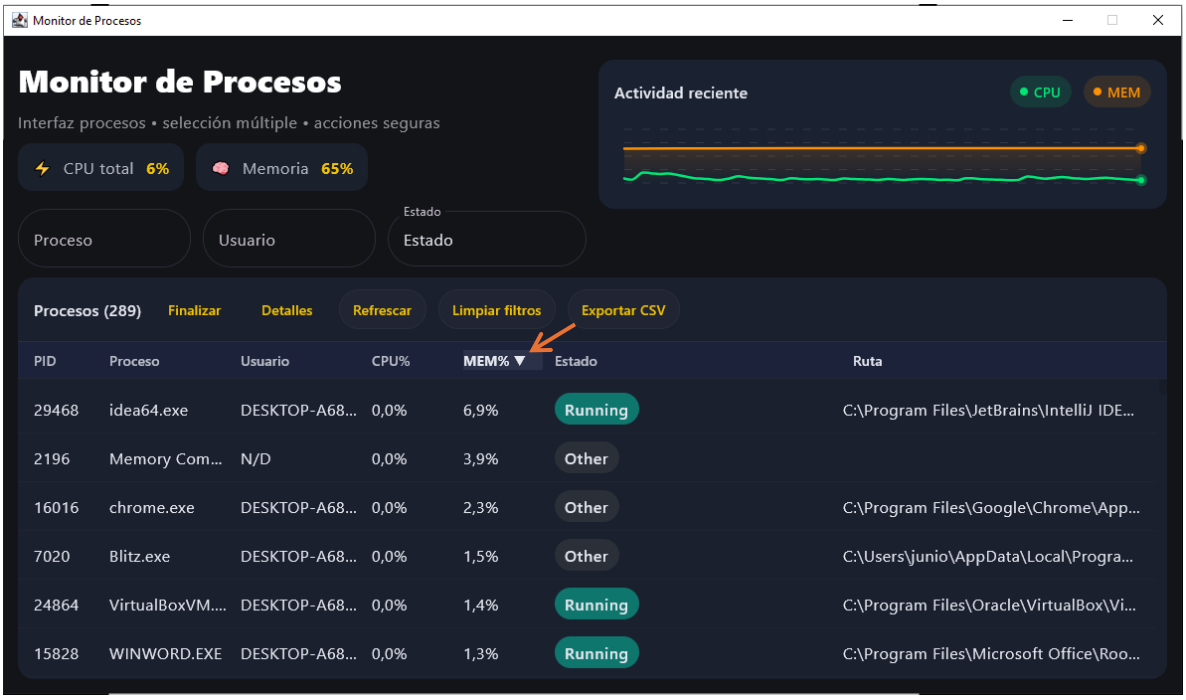

Figura 4. Podemos ordenar las columnas haciendo clic en su cabecera. Ordena de forma ascendente / descendente.

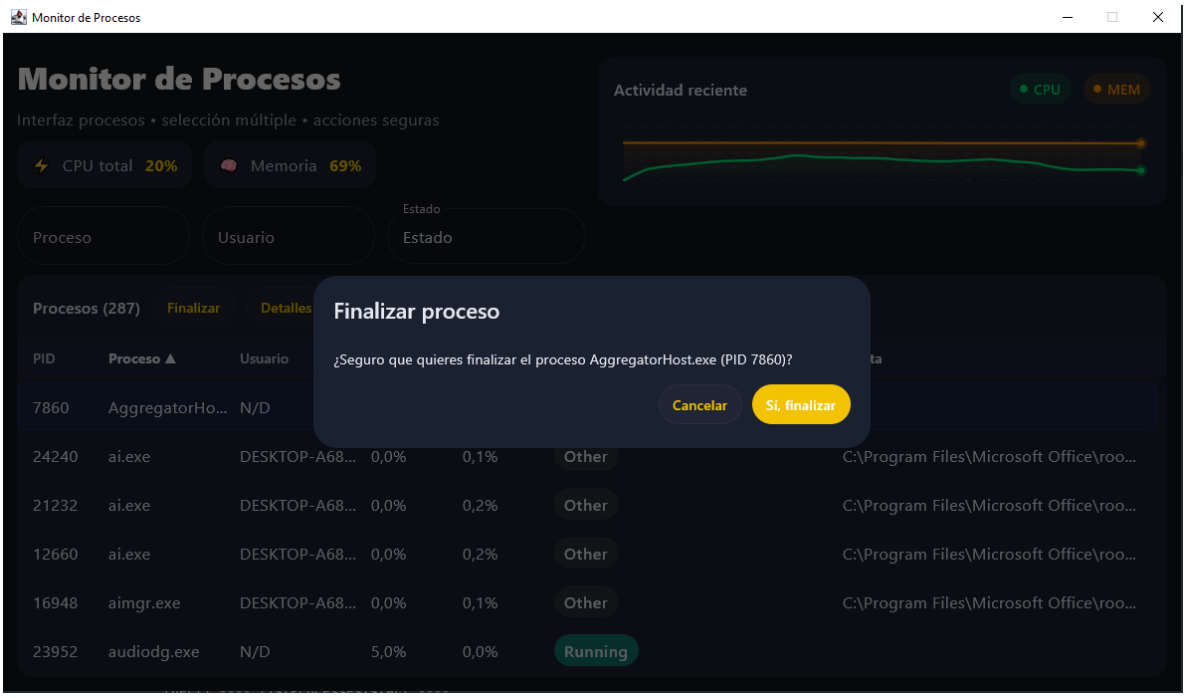

Figura 5. Podemos finalizar un proceso. Para ello lo seleccionamos y damos clic en finalizar, no que nos abre una alerta donde podemos finalizarlo.

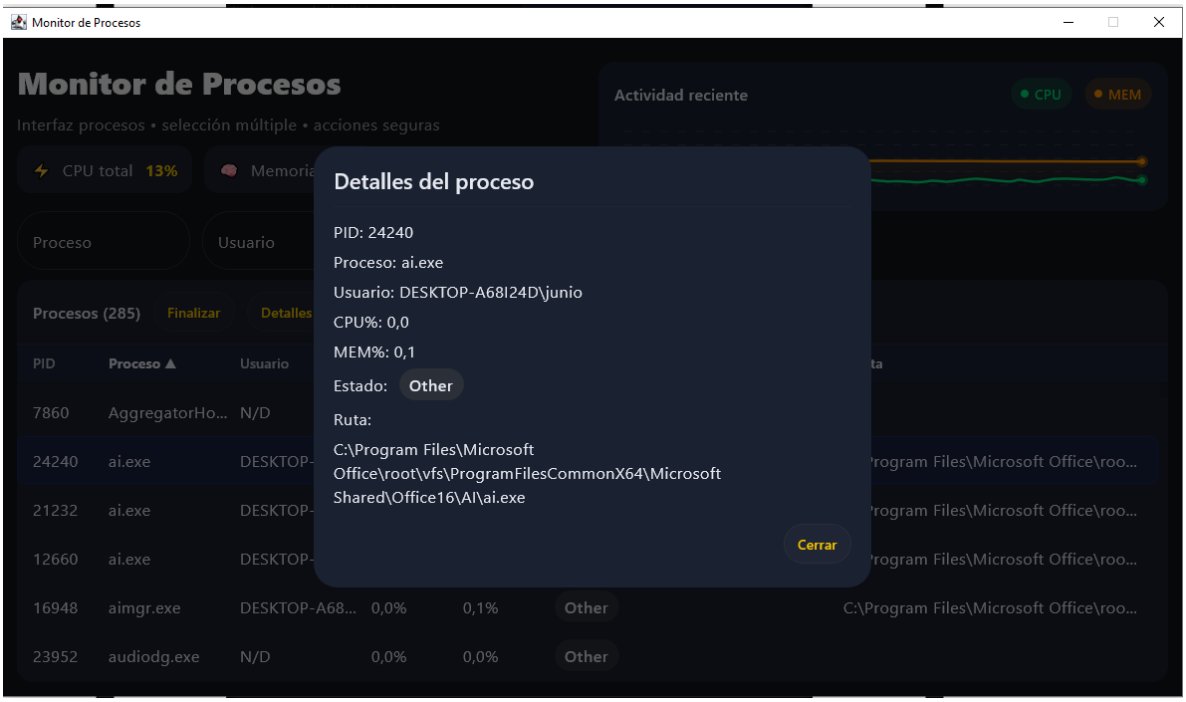

Figura 6. Si seleccionamos un proceso y pulsamos en detalle nos muestra una ventana con los detalles del proceso.

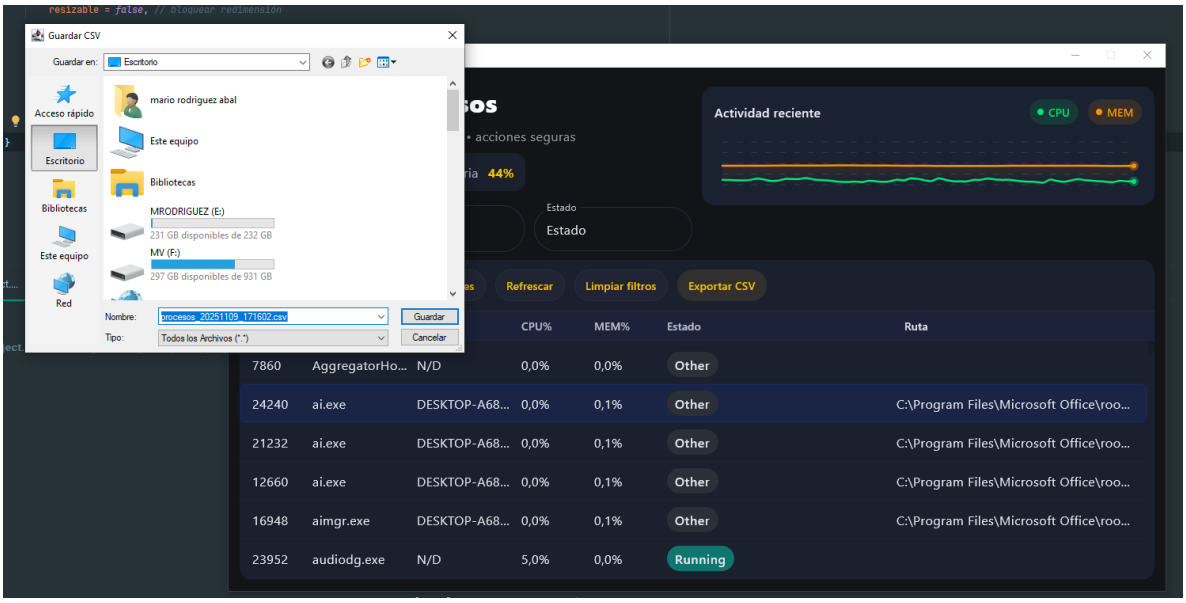

Figura 7. El botón de Refrescar nos sirve para “recargar los procesos” y Limpiar Filtros elimina cualquier filtro de Nombre, Usuario o Estado. Exportar CSV nos abre otra ventana donde podemos indicar la ruta en la cual nos creará una hoja de cálculo Excel con toda la información de los procesos.

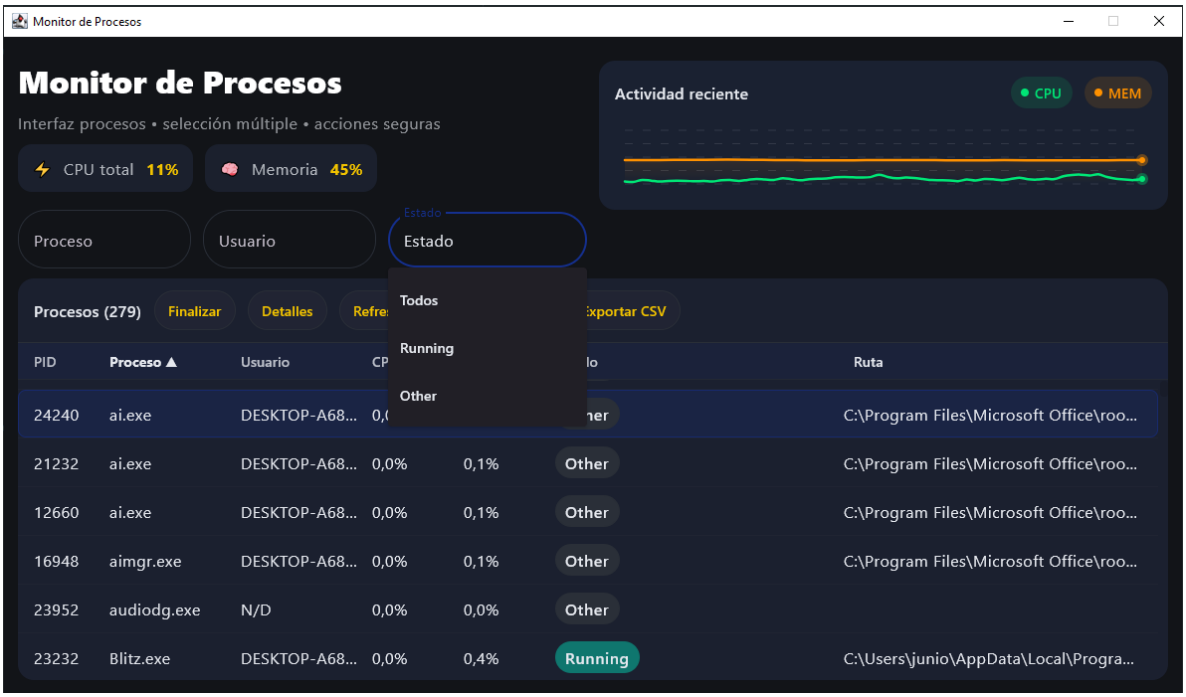

Figura 8. También podemos filtrar nuestros procesos ya sea por Proceso o Usuario, en ambos casos indicando el nombre. O bien filtrar por su estado. Donde Running serían los procesos que se están ejecutando en primer plano y Other en segundo.

7. Manual de Usuario

7.1. Requisitos previos

- SO: Windows 10/11 o Linux (Ubuntu 22.04/24.04 recomendadas).

- JDK: 17 o 21 (según build del proyecto).

- Permisos: para ver detalles/terminar procesos ajenos pueden requerirse permisos elevados del sistema.

7.2. Inicio rápido

Descomprimir el archivo MonitorDeProcesosUbuntu.zip o MonitorDeProcesosWindows.zip dependiendo de la versión deseada. Dentro está la carpeta del ejecutable el cual solo hay que iniciar para abrir la aplicación. En la otra carpeta están las versiones de instalación, solo hay que ejecutar el .exe y seguir unos simples pasos para su instalación.

7.3. Pantalla principal

- Indicadores globales (arriba a la izquierda): **CPU total** y **Memoria** del sistema.

- Gráfica (arriba a la derecha): actividad reciente de CPU/MEM.

- Controles de filtro: por **Proceso**, **Usuario** y **Estado**.

- Tabla: columnas **PID**, **Proceso**, **Usuario**, **CPU%**, **MEM%**, **Estado**, **Ruta**.

7.4. Operaciones habituales

- **Ordenar por columna**

Haz clic en la cabecera (PID, Proceso, CPU%, MEM%, etc.).

Cada clic alterna asc/desc .

- **Filtrar resultados**

Por Proceso/Usuario: escribe texto parcial; la búsqueda no distingue mayúsculas.

Por Estado: abre el desplegable y elige Running u Other.

Limpiar filtros: botón Limpiar filtros.

- **Actualizar la lista**

Refrescar: botón Refrescar (recarga inmediata).

Auto-refresco: la lista se actualiza automáticamente a intervalos configurados (indicados en la barra superior).

- **Ver detalles de un proceso**

Selecciona una fila.

Pulsa Detalles para abrir el diálogo con PID, usuario, CPU%, MEM%, estado y ruta/cmdline (Figura 6).

- **Exportar a CSV**

Pulsa Exportar CSV.

Elige carpeta y nombre del archivo.

El CSV respeta filtros y orden actual (Figura 7).

Codificación UTF-8; separador coma. Los campos con comas se entrecomillan.

- **Finalizar proceso (opcional)**

Selecciona una fila y pulsa Finalizar.

Revisa el diálogo de confirmación (aparecen nombre y PID).

Acepta con Sí, finalizar (Figura 5).

Seguridad: por defecto puede limitarse a procesos del usuario actual. La acción puede requerir permisos.

7.5. Mensajes y estados comunes

“Cargando procesos…”: primera lectura; es normal en el arranque.

“No hay resultados”: revisa el filtro actual.

“Permiso denegado” / “N/D” en métricas: el SO impide leer ciertos datos (procesos de otros usuarios, etc.).

Error al exportar: comprueba permisos de escritura y ruta destino.

8. Pruebas

8.1. Metodología

Se validó la aplicación en entorno de Windows 10 y Ubuntu 22.04/24.04 ejecutando el binario/distribuible generado y probando la interacción real del usuario: arranque, listado, ordenación, filtrado, auto-refresco, detalles, finalización (si habilitada) y exportación CSV. Se recogieron evidencias con capturas.

8.2. Casos verificados y evidencias

- **Arranque y carga inicial**: aparece “Cargando procesos…” y, tras unos segundos, la tabla muestra procesos (>0).
- **Ordenación por cabecera**: clic en cualquier cabecera alterna asc/desc; el indicador visual cambia correctamente (texto → orden lexicográfico; métricas → orden numérico).
- **Filtrado**: por texto (nombre/cmdline, insensible a mayúsculas), por **Usuario** y por **Estado** (Running/Other). “Limpiar filtros” restaura la vista.
- **Auto-refresco** y **Refrescar**: la lista se actualiza automáticamente al intervalo configurado y el botón fuerza recarga inmediata sin cambiar el intervalo.
- **Detalles de proceso**: diálogo con PID, usuario, CPU%, MEM%, estado y ruta/cmdline cuando está disponible.
- **Finalización** (opcional): muestra confirmación con nombre y PID; requiere permisos y puede estar limitada a procesos del usuario actual.
- **Exportación CSV**: genera fichero UTF-8 con cabecera y respeta filtros y orden de la vista; se abre en Excel/LibreOffice sin advertencias.

8.3. Rendimiento observado

- **Carga inicial**: ~1,5–2,5 s en equipos con ~200–350 procesos.
- **Refrescos**: respuesta fluida; sin bloqueos perceptibles.
- **Recomendación**: intervalo de auto-refresco **≥ 1000–2000 ms** para minimizar consumo en equipos modestos (intervalos menores incrementan el uso de CPU).

8.4. Limitaciones y comportamiento esperado

- Métricas no disponibles: en algunos procesos/SO se muestra **N/D** para CPU%/MEM% (restricciones del sistema o permisos).

- Datos de cmdline/usuario: pueden omitirse para procesos ajenos por políticas del SO.

- Estados: la categoría **Other** agrupa estados no "Running" (sleeping/suspended/idle, según el sistema).

- Rutas largas: pueden truncarse en la tabla (se preservan en Detalles/CSV).

8.5. Incidencias y resolución

- Exportación fallida por permisos o ruta: se notifica y permite elegir otra carpeta.

- Proveedor del SO no disponible (p. ej., PowerShell/typeperf en Windows, /proc/ps en Linux minimal): la app muestra aviso y degrada la funcionalidad; ver **Anexo D (Instalación y troubleshooting)**.

- Sin resultados: mensaje claro y acceso visible a **Limpiar filtros**.

9. Conclusiones y dificultades encontradas

9.1. Conclusiones

- Objetivo cumplido: se ha implementado un **Monitor de Procesos** de escritorio con **Kotlin + Compose for Desktop**, capaz de **listar**, **ordenar**, **filtrar**, **auto-refrescar**, **mostrar detalles** y **exportar a CSV**.

- Arquitectura limpia: la separación **UI** → **Dominio** → **Providers OS** permitió aislar el código específico de cada sistema operativo y mantener una **interfaz de programación estable** (contratos) para pruebas y evolución.

- Experiencia de uso: la **ordenación por cabecera**, el **filtro combinable** y el **auto-refresco** ofrecen una interacción fluida; los **mensajes de estado** (cargando, sin resultados, N/D) ayudan a entender qué ocurre.

- Portabilidad y entrega: el **empaquetado nativo** con Compose facilita distribuir binarios/instaladores en **Windows** y **Linux**, cubriendo el escenario académico habitual.

- Documentación y trazabilidad: se entrega **memoria en PDF**, **README ampliado**, **anexos con capturas** (reales/simuladas) y **Anexo de IA**; el **enlace al repositorio público** garantiza trazabilidad de versiones/commits.

9.2. Dificultades encontradas

- Heterogeneidad entre SO: la obtención de **CPU%/MEM% por proceso** varía (WMI/PDH en Windows frente a `ps`/`/proc` en Linux). Solución: **abstracción por providers** y **degradación controlada** cuando una métrica no está disponible (**N/D**).

- Rendimiento del refresco: intervalos muy bajos (<1000 ms) incrementan el uso de CPU. Mitigación: **intervalo recomendado 1000–2000 ms**, **refresco manual** y actualización por **diffs**.

- Permisos y visibilidad de datos: ciertos procesos del sistema o de otros usuarios restringen **cmdline/usuario**; se gestionó con **mensajes claros** y **limitación de acciones** (p. ej., finalizar) según permisos.

- Empaquetado y dependencias: en Linux, posibles faltas de **librerías GUI/OpenGL**; en Windows, dependencias de **PowerShell/typeperf** según entorno. Se documentó en **Guía de instalación y troubleshooting**.

10. Referencias

[1] JetBrains, "**Native distributions** — Compose Multiplatform Documentation," 2025. [En línea]. Disponible: JetBrains Help. Accedido: nov. 2025. JetBrains

[2] Oracle, "**OperatingSystemMXBean** (Java Platform SE 8)," 2025. [En línea]. Disponible: Oracle Docs. Accedido: nov. 2025. Oracle Docs

[3] Microsoft, "**Get-Counter** (Microsoft.PowerShell.Diagnostics)," 2025. [En línea]. Disponible: Microsoft Learn (en). Accedido: nov. 2025. Microsoft Learn

[4] Microsoft, “**typeperf** — Performance Counters Tools,” 2023–2025. [En línea]. Disponible: Microsoft Learn. Accedido: nov. 2025. Microsoft Learn+1

[5] M. Kerrisk (man7.org), “**ps(1)** — Linux manual page,” 2025. [En línea]. Disponible: man7.org. Accedido: nov. 2025. man7.org

[6] The Linux Kernel Documentation, “**The /proc Filesystem**,” 2025. [En línea]. Disponible: docs.kernel.org. Accedido: nov. 2025. docs.kernel.org

[7] JetBrains, “**Desktop-only API** — Compose Multiplatform Documentation,” 2025. [En línea]. Disponible: JetBrains Help. Accedido: nov. 2025. JetBrains

[8] JetBrains, “**Get started with Compose Multiplatform**,” 2025. [En línea]. Disponible: JetBrains Help. Accedido: nov. 2025. JetBrains

[9] Oracle (Early Access), “**OperatingSystemMXBean** — jdk.management (EA),” 2025. [En línea]. Disponible: download.java.net. Accedido: nov. 2025. download.java.net

[10] Sun/Oracle, “**UnixOperatingSystemMXBean** (JDK 11),” 2018–2025. [En línea]. Disponible: javadoc.scijava.org. Accedido: nov. 2025. javadoc.scijava.org

[11] ResuaCode (PSP), “**Programación de Servicios y Procesos – documentación**,” 2025. [En línea]. Disponible: resuacode.es/psp/. Accedido: nov. 2025. resuacode.es

[12] OpenAI, “**ChatGPT (versión web)**,” 2025. [En línea]. Disponible: OpenAI/ChatGPT. Accedido: nov-2025. OpenAI+1

11. Anexo, Uso de IA (ChatGPT) en el proyecto

Este anexo documenta el uso de **ChatGPT** como **asistente técnico y redactor** durante el desarrollo del proyecto “Monitor de Procesos (Compose for Desktop)”. Dado que no conservamos todos los chats originales, se incluyen **conversaciones simuladas** fieles al **flujo de trabajo real** y a los **resultados** integrados en el repositorio.

Referencia bibliográfica del uso de IA: *OpenAI, “ChatGPT (versión web)”, 2025. [En línea]. Accedido: 09-nov-2025.* (incluida en el Cap. 9 de la memoria).

11.1. B.1 — Diseño de la UI inicial (tabla + métricas globales)

Fecha aprox.: 2025-10-28

Contexto: Búsqueda de una estructura clara para la pantalla principal con **tabla de procesos**, **chips CPU/MEM** y **zona de gráfica**.

Prompt: “Quiero una interfaz Compose Desktop con tabla de procesos y chips CPU/MEM; solo interfaz por ahora, sin lógica imita el mockup que te acabo de pasar.”

Respuesta: Propuesta de layout con Row/Column, LazyColumn para tabla, cabecera con filtros/auto-refresco y espacio a la derecha para una mini-gráfica.

11.2. B.2 — Ordenación por cabecera (asc/desc con icono)

Fecha aprox.: 2025-10-29

Contexto: Añadir **ordenación** clicando la cabecera (PID/CPU/MEM/Proceso).

Prompt: “Quiero añadir la funcionalidad de ordenar de forma asc/desc y mostrar icono de orden en cabecera Compose, para poder ordenar cada fila de forma individual..”

Respuesta: Modelo de estado con sortKey y sortDir, comparator dinámico y flecha en el encabezado.

11.3. B.3 — Providers por SO (Windows/Linux)

Fecha aprox.: 2025-11-05

Contexto: Separar acceso a datos en **proveedores** específicos por SO: Windows (contadores) y Linux (ps//proc).

Prompt: “Ahora que tenemos la interfaz vamos a comenzar con la funcionalidad del programa por partes.”

Respuesta: Contratos de dominio; en **Windows**: Get-Counter/typeperf/WMI; en **Linux**: ps para snapshot + /proc para detalle; manejo de **N/D** si faltan permisos.

11.4. B.4 — Rendimiento y auto-refresco (latencia ~2 s)

Fecha aprox.: 2025-11-05

Contexto: Ajustar **intervalo de refresco** y minimizar coste de CPU.

Prompt (resumen): “Nos va a tirones, tenemos que mejorar la fluidez del programa y la manera de refrescar los datos.”

Respuesta: Usar **intervalos ≥1000–2000 ms**, agrupar IO, refresco por **diffs**, y solo cargar detalle bajo demanda.

11.5. B.5 — Finalización de procesos (confirmación segura)

Fecha aprox.: 2025-11-06

Contexto: Añadir acción “**Finalizar**” con confirmación y control de permisos.

Prompt: “Quiero un diálogo de confirmación con nombre y PID, y limitar la acción por permisos, la interfaz debe seguir el estilo de la aplicación.”

Respuesta: Diálogo modal; preferir terminación **suave**; restringir a procesos del usuario actual en entorno académico.

11.6. B.6 — Exportación CSV (coherente con vista)

Fecha aprox.: 2025-11-07

Contexto: Exportar la **vista actual** (filtros/ordenación) a **CSV UTF-8**.

Prompt: “Vamos a otorgarle función al CSV; con cabecera, respetando filtro y orden.”

Respuesta: Serialización con encabezados fijos; escapado de campos; validación abriendo en Excel/LibreOffice.

11.7. B.7 — Filtros por Proceso/Usuario/Estado

Fecha aprox.: 2025-11-07

Contexto: Definir **filtros combinables** con texto insensible a mayúsculas y dropdown de **Estado**.

Prompt: “Ahora tenemos que hacer la parte de filtros, por Nombre, Proceso o seleccionando uno del desplegable.”

Respuesta: Filtros reactivos; mensaje **“No hay resultados”** si vacía la tabla; botón **Limpiar filtros** visible.

11.8. B.8 — Empaquetado y .gitignore de binarios/artefactos

Fecha aprox.: 2025-11-09

Contexto: Preparar **distribuibles** (Windows/Linux) y evitar subir artefactos a Git.

Prompt: “Modifica el .gitignore para que no suba nada innecesario.”

Respuesta: Configurar nativeDistributions; ignorar **/build/, instaladores/zip, cache y metadatos IDE.

11.9. B.9 — Error gráfico/OpenGL en otro equipo (Linux/Windows)

Fecha aprox.: 2025-11-07

Contexto: En un PC diferente falló el render (drivers/lib Mesa).

Prompt: “Cómo solucionar error OpenGL/Mesa o drivers en Compose Desktop.”

Respuesta: Instalar **Mesa/OpenGL** en Linux o actualizar drivers en Windows; añadir al **troubleshooting**.

Declaración de uso responsable de IA

- La IA se empleó como apoyo para diseño, arquitectura, buenas prácticas, resolución de dudas y redacción técnica.
- No se pegó código ajeno sin comprensión ni revisión; todas las decisiones se validaron y adaptaron al contexto del proyecto.
- Las citas a documentación oficial y a ChatGPT están recogidas en Referencias.
- Las capturas aquí adjuntas se marcan como “Captura simulada” cuando no existe el original.

Limitaciones y salvaguardas

- La IA puede alucinar en comandos específicos; se mitigó corroborando con fuentes oficiales.
- Se evitó el uso de APIs internas de Java para no requerir --add-opens salvo necesidad.
- Las métricas por proceso dependen del SO y permisos; la IA aportó opciones, pero se optó por una degradación controlada (N/D).