Memoria del Proyecto: Monitor de procesos sencillo

Título: Monitor de procesos sencillo

Nombre: Lois Figueira Castiñeiras

Módulo y curso: PSP | 2ºDAM

Fecha de entrega: 16/11/2025

Índice

1. Introducción

El proyecto consiste en la creación de una aplicación desarrollada en Kotlin para listar los procesos del sistema en ejecución mostrando su PID, nombre, usuario y uso de CPU/memoria, siendo un monitor de procesos sencillo. Esta estará disponible para distintos sistemas operativos y ofrecerá una interfaz gráfica moderna y limpia con ayuda de compose.

2. Tecnologías usadas y arquitectura del sistema

2.1 Tecnologías usadas

- **Kotlin:** Lenguaje de programación usado para todo el código del programa.
- **IntelliJ IDEA:** Entorno de desarrollo utilizado para la creación de la aplicación.
- **Gradle:** Herramienta de compilación utilizada para gestionar todo el proyecto.
- **OSHI:** Librería externa utilizada para obtener toda la información necesaria de los procesos en sistemas Windows (necesaria para obtener la CPU).
- **Git:** Sistema de control de versiones utlizado.
- **GitHub:** Sistema de control de versiones utilizado donde se aloja el proyecto remotamente.
- **JDK 17:** Kit de desarrollo Java utilizado por el proyecto.
- **Compose Wizard:** Herramienta para hacer el proyecto multiplataforma.
- **Compose:** Herramienta utilizada para la interfaz gráfica del monitor de procesos.

2.2 Arquitectura del sistema

La aplicación sigue una arquitectura **MVVM (Model-View-ViewModel)**, para tener una clara separación entre:

- **Modelo:** modelo de los procesos y toda su información (los modelos u objetos).
- **Vista:** interfaz construida con Compose (lo que ve el usuario).

- **ViewModel:** funciones de filtrado, actualización de procesos, control de la UI, etc (la lógica del programa).

3. Funcionalidades

- Listar procesos en ejecución con PID, nombre y usuario.
- Eliminar o matar procesos seleccionados.
- Actualizar la lista de procesos.
- Filtrar procesos por nombre o usuario.
- Mostrar resúmenes visuales de CPU y RAM.

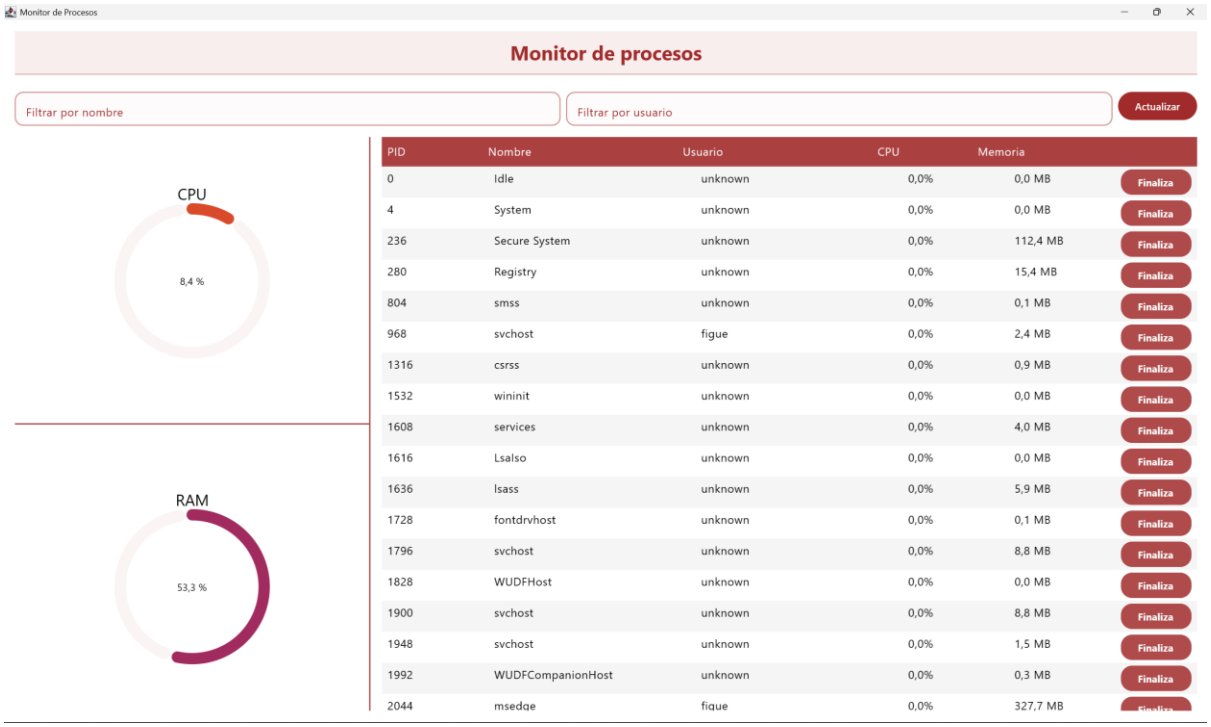

Imagen 1: Captura de pantalla de la interfaz de usuario

Descripción: Esta imagen muestra la lista de procesos con sus métricas, las zonas de filtrado, el botón para actualizar, los resúmenes y la opción para finalizar los procesos.

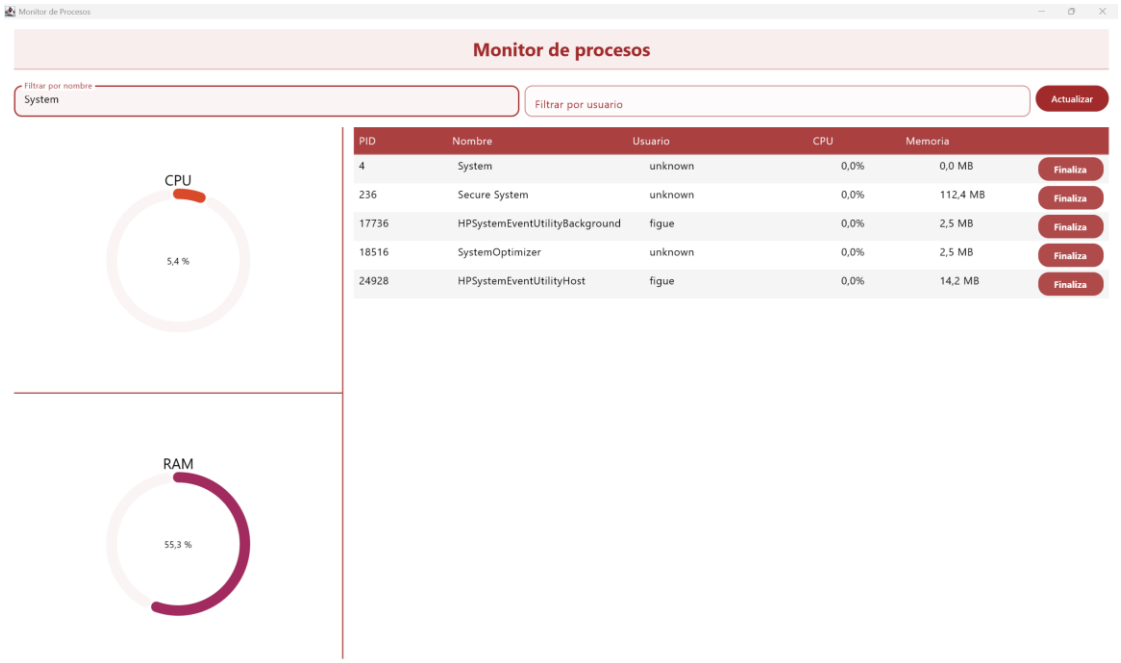

Imagen 2: Captura de pantalla del uso de los filtros

Descripción: En esta captura se muestra que ocurre al filtrar por ejemplo el nombre de los procesos a “System”.

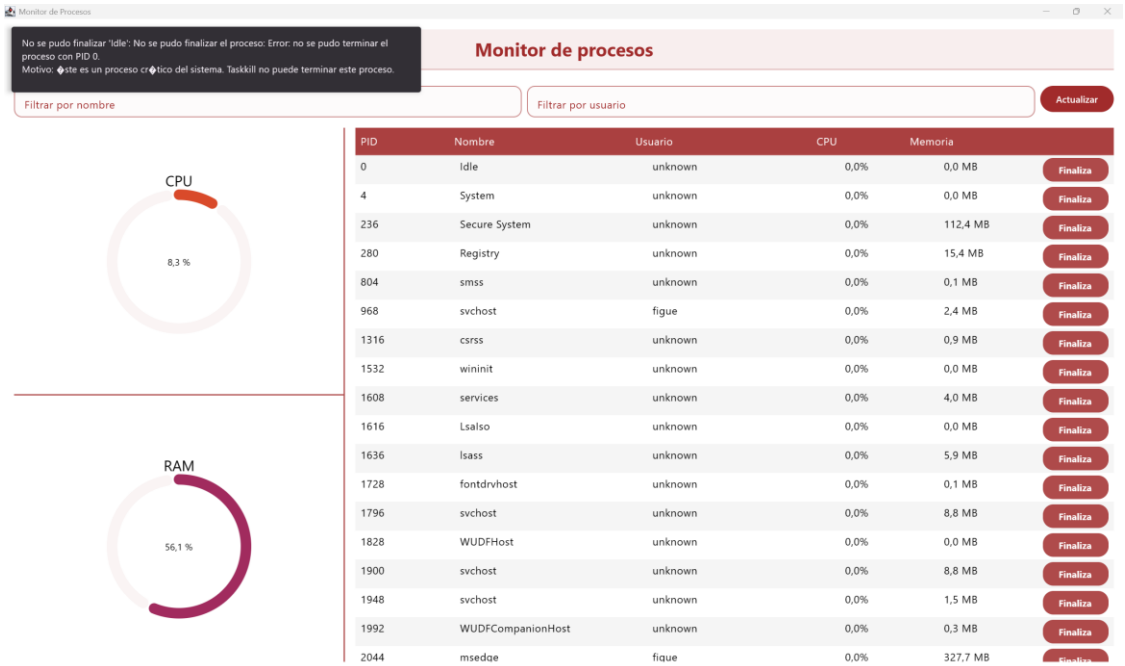

Imagen 3: Captura de pantalla del uso del botón para eliminar.

Descripción: En esta captura se muestra que ocurre al intentar eliminar un proceso crítico o del que no se tienen suficientes permisos.

4. Manual de usuario

4.1 Abrir desde un IDE

1. Para hacer uso del proyecto desde un IDE como IntelliJ IDEA, se deberá clonar el repositorio en el enlace proporcionado en el anexo. Desde un terminal se haría “git clone enlace_al_repo”.

2. Una vez clonado, si es necesario se configurará el archivo “gradle.properties” y en la línea “#org.gradle.java.home=C:\\Program Files\\Java\\jdk-17.0.4.1” se se descomenta y se pondrá la ruta al JDK, preferiblemente JDK 17 para compatibilidad.

3. Una vez todo listo se ejecuta.

4.2 Instalar en Windows

1. Se accede al repositorio y en el apartado releases se descarga el .msi.

2. Luego se sigue el proceso de instalación.

3. Por último, se busca una carpeta con el nombre “es.lfigueira”, donde se encuentra la aplicación.

4.3 Instalar en Linux/Mac

1. Se accede al repositorio y en el apartado releases se descarga el .deb.

2. Una vez hecho se localiza y se ejecuta:
   “sudo dpkg –i rutaAlDirectorio/ProcessMonitor_Package_Linux.deb”.

3. Una vez hecho, la aplicación ya aparecerá entre las aplicaciones y en la tienda.

4.4 Manual de uso

Manual de uso del monitor de procesos:

1. Al abrir la aplicación se mostrará una interfaz con múltiples elementos.
2. En la zona superior debajo del encabezado de la app serán visibles dos campos de búsqueda, una para filtrar los procesos por nombre y otro para filtrarlos por usuario.
3. A la derecha de los filtros, estará el botón actualizar para actualizar la lista de procesos en ejecución.
4. En la zona izquierda se mostrarán los resúmenes de CPU y RAM en uso por los procesos activos.
5. En la parte derecha se mostrará la lista completa de procesos en ejecución con un encabezado con las distintas columnas de cada proceso PID, nombre, CPU...
6. A la derecha de cada proceso aparecerá la opción de finalizar el proceso seleccionado, esto solo ocurrirá si el proceso no es crítico o si se tienen suficientes permisos para eliminarlo.

5. Pruebas

Para comprobar el correcto funcionamiento de la aplicación y como actúa en distintos sistemas operativos se han realizado una serie de pruebas diferentes.
Se ha probado a instalar la aplicación en distintos sistemas operativos como Windows y Linux para comprobar la correcta escalabilidad y adaptabilidad de la interfaz gráfica.
También se ha probado a filtrar los procesos por nombre, por usuario y por ambos campos a la vez.
Otra prueba, fue comprobar si al finalizar procesos estos se eliminaban correctamente, también se ha comprobado si al intentar matar procesos críticos o de los que no se tienen suficientes permisos muestra los respectivos mensajes informativos al usuario y no se hace.
También se ha comprobado si la lista de procesos se actualiza correctamente al utilizar el botón para actualizar.

Por último, se ha comprobado que los instaladores funcionen para sus respectivos sistemas operativos.

6. Conclusiones y dificultades encontradas

Durante la realización del proyecto se han adquirido una serie de conocimientos y han surgido muchas dificultades.

En cuanto al aprendizaje, se ha descubierto la librería Oshi para mostrar información de procesos. También se ha aprendido a usar compose para hacer una interfaz gráfica limpia y moderna con colores y estilos propios. Otro conocimiento es aprender a identificar sistemas operativos en los que se usa la aplicación. También se ha aprendido a trabajar un poco con procesos y como interactúan con el sistema.

En cuanto a dificultades, ha sido tedioso hacer compatibles el Gradle con el JDK usado y resolver todos los errores que el Gradle causaba por dependencias o tareas que no era capaz a hacer. El uso de compose ha sido bastante difícil dado al nulo conocimiento y uso de este con antelación u otros proyectos. Otra dificultad ha sido la realización de la lógica del programa en sí puesto a la falta de práctica con procesos.

En resumen, lo más tedioso y difícil, ha sido el uso de Gradle por las múltiples veces que fallaba y a veces sigue fallando en el proyecto o al desarrollarlo desde otro dispositivo con otro JDK instalado.

7. Referencias

- Apuntes de Kotlin: El lenguaje Kotlin | Programación de Servicios y Procesos
- Documentación de Compose: Cómo comenzar a usar Jetpack Compose | Jetpack Compose | Android Developers
- Documentación de Gradle: Gradle User Manual
- Librería OSHI: oshi/oshi: Native Operating System and Hardware Information
- Apuntes sobre procesos: Fundamentos de la Programación Concurrente y Procesos | Programación de Servicios y Procesos
- GitHub repositorio: Monitor_procesos_sencillo
- Compose Wizard: Kotlin Multiplatform Wizard | JetBrains

8. Anexos

8.1 Repositorio GitHub

Enlace al repositorio de GitHub donde se aloja el proyecto y los instaladores.

loisfigueira/Monitor_procesos_sencillo

8.2 Uso de la IA

Resumen de uso de IA:

- Se utilizó ChatGPT para resolución de problemas de Compose, rescalado de elementos, configuración de Gradle (lo más preguntado), automatización de generación de instaladores (con uso de workflows) y formas de obtener información de los procesos.
- Se recibieron sugerencias de código, correcciones de errores y mejoras de interfaz.

Ejemplos:

Prompt:
“¿Cómo genero instaladores para Windows y Linux desde GitHub?”

Respuesta:
Creó un workflow completo con tres jobs:

- Compilar MSI
- Compilar DEB
- Crear release y adjuntar artefactos incluyendo configuración de Java 17, rutas correctas y permisos.

Prompt:
“Haz que la interfaz se rescale según resolución.”

Respuesta:
Expliquó que se debía usar WindowState y LocalDensity, y propuse una solución correcta basada en WindowInsets y tamaño de ventana, evitando errores de APIs Android.

Prompt:
«Gradle dice que no encuentra la tarea *packageDesktopDistribution*. ¿Qué hago?»

Respuesta de la IA:

Explicó que Compose Multiplatform ha cambiado su sistema de tareas, y que en la versión del proyecto ya no existe esa tarea e indicó qué tareas válidas se podían usar (packageMsi, packageDeb, packageDmg) y cómo listarlas con ./gradlew tasks.

Prompt (resumido):

“Quiero una interfaz con dos columnas: a la izquierda un monitor de recursos y a la derecha una lista de procesos.”

Respuesta de la IA (resumida):

Generó un layout en Compose Multiplatform usando Row, Column, tarjetas Material3, separación equilibrada y ejemplos de uso de LazyColumn para la lista de procesos.

Conclusión del anexo:

El asistente de IA facilitó la resolución de problemas técnicos y agilizó la creación de interfaz, sin sustituir la experimentación y pruebas propias del desarrollador.