Aplicación gráfica sencilla en Kotlin con JavaFX para ver todos los procesos que se están ejecutando en el sistema

[crayon-6a9d61ccece5a945951778/]

1. import javafx.application.Application: Importa la clase Application de JavaFX, que se utiliza para crear aplicaciones con interfaces gráficas.
2. import javafx.scene.Scene: Importa la clase Scene de JavaFX, que representa el contenido visual de una ventana.
3. import javafx.scene.control.Button: Importa la clase Button de JavaFX, que representa un botón en la interfaz gráfica.
4. import javafx.scene.control.TextArea: Importa la clase TextArea de JavaFX, que se utiliza para mostrar y editar texto en la interfaz.
5. import javafx.scene.layout.VBox: Importa la clase VBox de JavaFX, que es un contenedor vertical para organizar elementos en la interfaz.
6. import javafx.stage.Stage: Importa la clase Stage de JavaFX, que representa la ventana principal de la aplicación.
7. class ProcessListApp : Application() {: Comienza la definición de la clase ProcessListApp, que extiende Application, lo que la convierte en una aplicación JavaFX.
8. override fun start(primaryStage: Stage) {: Inicia la implementación del método start, que es el punto de

entrada de la aplicación JavaFX. Recibe un objeto Stage como argumento, que representa la ventana principal.

9. primaryStage.title = "Lista de Procesos en Ejecución": Establece el título de la ventana principal de la aplicación.
10. val layout = VBox(10.0): Crea un objeto VBox llamado layout con un espaciado vertical de 10 píxeles para organizar elementos en la interfaz.
11. val processListTextArea = TextArea(): Crea un objeto TextArea llamado processListTextArea para mostrar texto.
12. val refreshButton = Button("Actualizar Lista de Procesos"): Crea un botón con el texto «Actualizar Lista de Procesos».
13. refreshButton.setOnAction { ... }: Asigna un manejador de eventos al botón. Cuando se hace clic en el botón, se ejecutará el código dentro de las llaves.
14. layout.children.addAll(refreshButton, processListTextArea): Agrega el botón y el área de texto al contenedor layout.
15. val scene = Scene(layout, 500.0, 400.0): Crea un objeto Scene con el contenedor layout y dimensiones de 500×400 píxeles.
16. primaryStage.scene = scene: Establece la escena en la ventana principal.
17. primaryStage.show(): Muestra la ventana principal.
18. private fun getAllProcesses(): String {: Inicia la definición de la función getAllProcesses, que obtendrá la lista de procesos en ejecución.
19. val processHandle = ProcessHandle.allProcesses().toList(): Obtiene una lista de objetos ProcessHandle representando los procesos en ejecución y la convierte en una lista.
20. val processList = processHandle.joinToString("\n") { ... }: Convierte la lista de ProcessHandle en una cadena de texto, aplicando la función lambda a cada elemento de la lista.
21. "ID del Proceso: ${it.pid()}\n": Agrega el ID del

proceso a la cadena de texto.

22. "Nombre del Proceso: ${getProcessName(it)}\n": Agrega el nombre del proceso (basado en el comando ejecutado) a la cadena de texto.
23. "Comando Ejecutado: ${getProcessCommand(it)}\n": Agrega el comando ejecutado por el proceso a la cadena de texto.
24. "----------": Agrega una línea divisoria.
25. return processList: Retorna la cadena de texto que representa la lista de procesos.
26. private fun getProcessName(process: ProcessHandle): String { ... }: Define la función getProcessName que recibe un objeto ProcessHandle y devuelve el nombre del proceso.
27. return process.info().command().orElse("Desconocido"): Utiliza el método info().command() para obtener el nombre del proceso o «Desconocido» si no está disponible.
28. private fun getProcessCommand(process: ProcessHandle): String { ... }: Define la función getProcessCommand que recibe un objeto ProcessHandle y devuelve el comando ejecutado por el proceso.
29. return process.info().commandLine().orElse("Desconocido"): Utiliza el método info().commandLine() para obtener el comando ejecutado o «Desconocido» si no está disponible.
30. fun main() { ... }: Define la función main, que inicia la aplicación JavaFX llamando a Application.launch y pasando la clase ProcessListApp como argumento.