Programa para gestionar el tiempo mediante avisos

Alberto Lara Alberola

Lenguaje: Java

Aplicación escritorio

Desarrollado sobre SO Linux 20.04.

Entorno desarrollo: IntelliJ IDEA 2021.3.3 (Community Edition)

21 de abril de 2022

Programa

Se ha creado un programa usando **JavaFX** y **SceneBuilder**, utilizado el modelo vista controlador, que nos ayuda con la gestión del tiempo mediante alarmas acompañadas de texto que nos avisaran en un momento determindo. Es un programa simple que puede resultar bastante útil y que ademas se puede tomar como una parte de un proyecto más grande.

Aunque la función que cumple el programa sea bastante simple, partes de su implementación tiene algo de dificultad. Se han utilizado hilos que ejecutan tareas en momentos determinados usando TimerTask, se ha habilitado que estas tareas puedan ser modificadas o eliminadas antes del tiempo de su cumplimiento. Estas tareas son visibles en un ListView y desaparecen llegado el momento.

Esta es la primera versión, donde hace faltara factorizar algo de código. Se ha dejado sin implementar un método concreto para el botón modificar, no

obstante es muy fácil de implementar ya que se pueden recuperar los objetos del ListView y modificarlos en función de lo que se necesite. Como siempre, todo el código esta completamente documentado para su comprensión.

Vista principal. GestionTarea

Esta es la ventana principal del programa. Esta formada por un *ListView*, el rectángulo de color más claro, un botón que nos permite añadir un evento, el botón de modificar que no tiene asociado método, sendos botones que nos permiten añadir alarmas rápidas sin necesidad de introducir datos y la imagen de la papelera que nos permitirá eliminar ítems seleccionados del *ListView*.

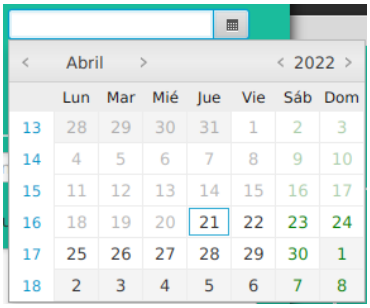
Controlador. GestionTarea

Como parte más destacable del controlador encontramos el método asociado al botón añadir evento. Este método nos abre una nueva ventana y carga el controlador asociado a esa nueva vista, mediante el cual podremos recuperar los datos que se introducirán en ella usando los métodos que hemos implementado para ello. También es importante comentar la necesidad de recuperar el estado del objeto *Timer* asociado al objeto *Alarma*. Mediante este objeto podremos eliminar o modificar las tareas programadas asociadas a un único objeto *Alarma*.

Vista nueva alarma. NuevaAlarma

Esta es la nueva ventana, que es modal, se abre cuando presionamos el botón añadir alarma. Esta compuesta de un *DatePicker* que nos permitirá elegir una fecha, dos *Spinners* de los que extraeremos las horas y los minutos, un titulo que aparecerá en el ListView y nos ayudara a identificar el aviso así como de un texto más extenso que se nos mostrara cuando se active la alarma. También podemos ver un CheckBox para seleccionar si queremos un aviso sonoro y el botón que confirma la alarma.

El DatePicker se ha modificado para que no nos permita seleccionar fechas pasadas y nos muestre los fines de semana en color verde.

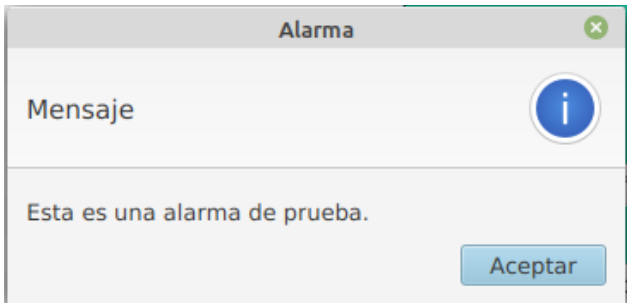


Controlador nueva alarma. NuevaAlarma

El controlador asociado a la vista para crear la nueva alarma no tiene mucha complejidad. Dispone de un método para inicializar los elementos que hemos modificado, como el DatePicker y los valores de los Spinners. Quizá lo más destacable es la gestión del hilo al que le indicamos una acción que debe realizar a una determinada hora, es importante almacenar el estado del objeto *Timer* para una gestión posterior así como almacenar también el estado del objeto Alarma para poder recuperarlo desde el controlador de la vista anterior. Existen diferentes formas de pasar información entre escenas, para los datos que necesita el programa es más simple ajustar el controlador para poder recuperar el objeto. Siempre hay que apostar por la simplicidad, en ocasiones puede no parecer lo más correcto o profesional pero encontrar los caminos más simples siempre será la mejor solución. No debemos nunca crear códigos complejos que solo entandamos nosotros.

Destacar la importancia de finalizar siempre los hilos que creemos, por ejemplo la clase *Timer Task* que recibe un objeto *Timer*, con el que programamos la ejecución de una parte de nuestro código a una determinada hora, necesita ser finalizado. Quizá sea normal pensar que estos hilos programados una vez ejecuten la parte del código que les corresponda finalizaran, pero no es así. Al realizar pruebas sobre este software se ha comprobado que estos hilos no finalizan nunca su ejecución, no se entiende el motivo, pero aunque llegue la hora y estos ejecuten su código, si no indicamos al final de este código la finalización del proceso estos no terminan. Esto causa que cada vez el programa tenga más y más procesos que nunca finalizan y cuando se cierra el programa estos continúan ejecutándose en el sistema. Evidentemente esto no se puede permitir puesto que terminaría siendo un software muy poco eficiente y con una capacidad limitada.

Cuando llega el momento se ejecuta el código programado y se genera un mensaje *Alert*.



Para mostrar el mensaje se ha utilizado *Platform.runLater* dentro del codigo que ejecuta el hilo. Esto nos permite que se ejecute en el subproceso de la aplicación con el resto de la interfaz de usuario.

También es importante proteger, tanto los métodos de este controlador como cualquiera que trabaje con el objeto *Alarma*, para que no reciban un objeto *Null* que pueda causar errores.

Clase Alarma

Se ha creado una clase que representa el objeto Alarma. Sus atributos coinciden, como es natural, con la información solicitada para crear el aviso o evento. Además de la abstracción que nos da la POO, también nos facilita el paso de información entre escenas. Lo más relevante de la clase es la sobre-escritura de su método *ToString*. Es importante ya que el ListView llamara a este método para mostrar información sobre el objeto que contiene, así podremos personalizar la información que se mostrara.