UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

ELO 329

Tarea 2

- Grupo 1 (paralelo 201)
- Integrantes:
  - Francisco Encina
  - Matías Torres
  - José Beltrán
  - Maximiliano Pozo

- Fecha de entrega: 20 abril de 2023

Índice

Resumen

El siguiente informa trata de la tarea 2 de la clase ELO 329 la cual consiste en programar, a través de lenguaje Java junto al software *JavaFX*, un simulador gráfico de una alarma domiciliaria. Después de entregar la descripción de lo que solicita la entrega, se va a continuar dando a conocer los diversos problemas y dificultades que surgieron al momento de desarrollar el código. Luego, se prosigue con el diagrama UML correspondiente, el cual se basa en la realización de la tarea misma. Finalmente, se da a conocer toda la información adicional para que se pueda comprender a gran escala la solución implementada en la entrega final del simulador solicitado.

Descripción del programa

Esta entrega consiste en adaptar la tarea pasada sobre una alarma domiciliaria y extenderla a un simulador gráfico a través de *JavaFX*. Para esto se necesita considerar los diferentes elementos de la entrega anterior a lo largo de las cuatro etapas, los cuales permiten el avance progresivo del mismo programa. En la primera etapa, se solicita crear una ventana en la interfaz y que se pueda interactuar con la misma con respecto a la apertura; en la segunda etapa se pide crear otra ventana adicional, dos puertas, una sirena y la central con sus botones correspondientes; la tercera etapa consiste en agregar un detector PIR y una persona la cual debe ser movible al arrastrar el mouse; y por último la cuarta etapa en l cual se deben cumplir con todas las funcionalidades del sistema de alarma, es decir, debe cumplir con todas las especificaciones de las secciones anteriores incluyendo un menú que permita el armado perimetral, agregar personas y eliminar personas.

Solución

Hemos desarrollado una aplicación en *JavaFX* que cuenta con una interfaz intuitiva para controlar ventanas, puertas y una alarma. Con esta aplicación, los usuarios pueden armar y desarmar la alarma según sus necesidades de seguridad.

La aplicación presenta una serie de ventanas y puertas que se muestran en la pantalla principal. Cada elemento tiene una representación gráfica y un estado asociado que indica si está abierto o cerrado. Los usuarios pueden interactuar con estos elementos para simular su apertura o cierre.

La alarma se puede armar o desarmar a través de la central y sus botones específicos en la interfaz. Cuando la alarma está armada y una ventana o puerta se abre, se activa automáticamente una alarma sonora. Esto alerta a los usuarios sobre una posible intrusión en el lugar protegido.

Una de las características clave de esta aplicación es que no se puede armar la alarma si hay una ventana o puerta abierta. Esto garantiza que los usuarios no puedan dejar la propiedad desprotegida al activar la alarma.

Además, la aplicación proporciona retroalimentación visual y sonora para indicar el estado actual de la alarma y la condición de las ventanas y puertas. Esto ayuda a los usuarios a tener un control claro sobre la seguridad de su entorno.

En resumen, nuestra aplicación de *JavaFX* ofrece una solución fácil de usar para controlar ventanas, puertas y una alarma. Con la capacidad de armar y desarmar la alarma, y una funcionalidad que evita armarla si hay una ventana o puerta abierta, brindamos a los usuarios un mayor nivel de seguridad y tranquilidad en sus hogares o lugares de trabajo.

Dificultades

Una de las dificultades que se presentó durante la etapa 2 tiene relación con la clase House, ya que el mismo código base presentaba un problema en particular. Al momento de crear ventanas, estas se creaban correctamente, sin embargo, no se establecía adecuadamente su sensor magnético en la central, vale decir, no se estaba agregando el sensor magnético al array que tenía la central. Esto generaba que, al abrir ventana, la central no detectara la ventana abierta y por ende no sonaba la sirena. Junto con esto, también se permitía el armado total a pesar de tener la misma ventana abierta, generando un incorrecto funcionamiento de la aplicación.

Otra dificultad para destacar fue que, en el archivo *MagneticSensorView*, se realiza la construcción correcta de la parte gráfica, sin embargo, no se indica en el código ni en la tarea si se debía agregar alguna animación dentro de la interfaz para mostrar cuando el sensor magnético está abierto o cerrado. Debido a todo lo anterior, se decidió dejar sin la animación.

El siguiente inconveniente que se presentó durante el desarrollo de esta entrega fue que, a pesar de haber dado una pequeña descripción en la tarea del modelo de diseño “MVC”, en el código base no se crearon archivos “*controller*”, por lo que manejar los archivos no fue algo simple de organizar y generó mucha confusión al momento de realizar y desarrollar el código.

Como última dificultad a remarcar, fue que al observar el código base para la animación de la puerta, se encuentra un atributo de tipo *TranslateTransition* el cual no permite generar animaciones que roten, por ende, la puerta se programó como un acceso que simplemente se desliza (como por ejemplo las puertas deslizantes de japón). Cabe destacar que para generar animaciones que roten, se debe utilizar *RotateTransition* pero, como no estaba en el código base, se prefirió no modificar nada.

Dificultades no solucionadas

Durante el proceso de implementación del sistema de seguridad, se encontraron diversas dificultades, una de las dificultades que no pudimos solucionar fueron las de los detectores PIR y el menú de creación de personas necesarias para la etapa 4.

A pesar de que se crearon estas clases, no interactúan como deberían. Es decir, los sensores no detectaban a las personas, incluso cuando la alarma está armada y las personas pasan frente a ellos.

Se realizaron múltiples pruebas para tratar de identificar la causa del problema, pero por el tiempo no se encontró la solución definitiva, Se realizaron ajustes a los sensores y se ha verificado que estén funcionando correctamente, pero aun así no se logró que detecte a las personas de forma efectiva.

Sobre la etapa 4. El menú de la ventana principal para agregar personas u eliminar, solo se logró el crear personas, sin embargo, estas no son detectadas por los sensores. El armado perimetral no existe en nuestra entrega.

Diagrama UML

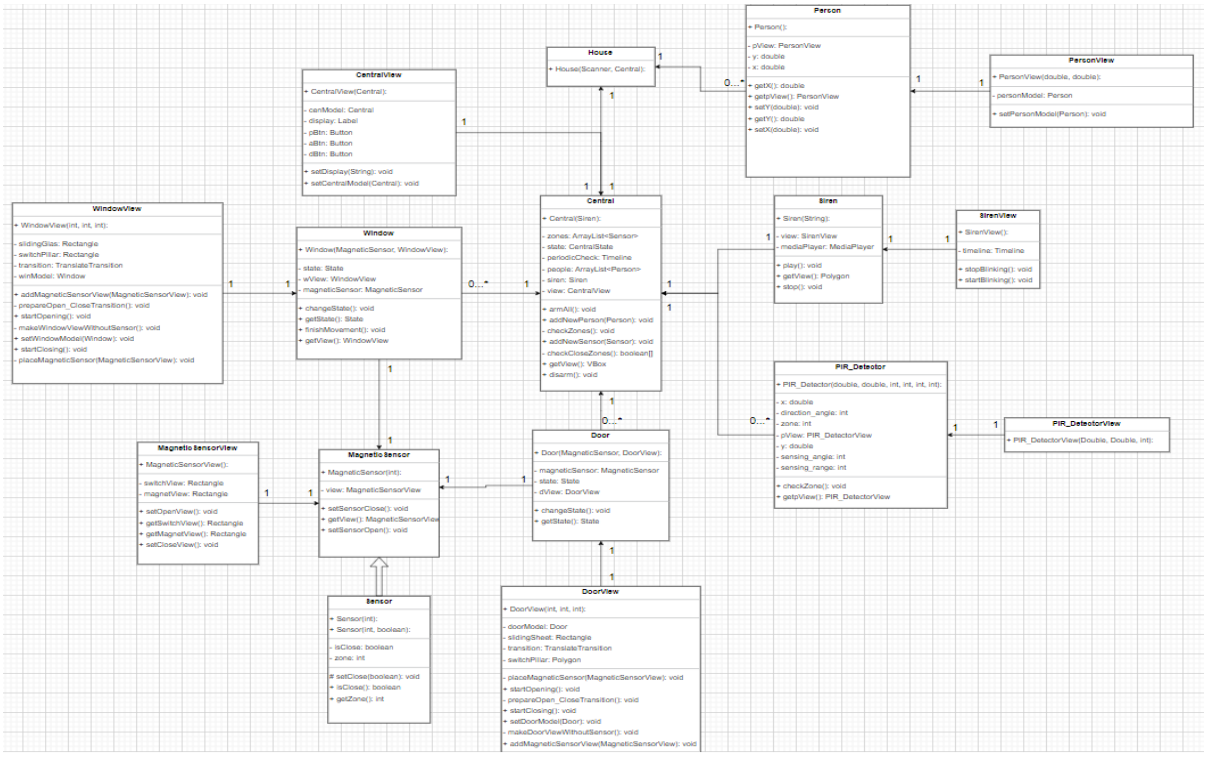

Fig. 1. Diagrama de clases