Aplicación Android: *Awakener*

Hecho por: Galder García

Primera Entrega (17/03/2024)

Índice de contenidos

Introducción

Código

Esta documentación es parte de la primera entrega de la asignatura Desarrollo Avanzado de Software. El código de la aplicación se puede encontrar en el siguiente repositorio público de GitHub:

https://github.com/galdergcupv/Awakener/tree/master

Descripción de la aplicación

Awakener es un despertador/alarma inspirado en la aplicación para dispositivos móviles *Alarmy*, un despertador que destaca por las diferentes formas que ofrece para apagar sus alarmas con el objetivo de asegurar que el usuario termina despertándose. *Alarmy* incluye métodos como requerir que el usuario escaneé un código de barras, agite el dispositivo un determinado número de veces, o resuelva problemas matemáticos. Es en esta última opción en la que ***Awakener*** pretende enfocarse y expandir más allá de los problemas matemáticos, ofreciendo minijuegos que deben completarse para poder apagar la alarma. También ofrece un menú en el que se pueden probar dichos minijuegos a modo de práctica o test, sin requerir la programación de ninguna alarma.

Descripción técnica de la aplicación

Diseño y estructura

La aplicación está compuesta por dos actividades “molde” que hacen uso de fragmentos para modificar la información que se muestra en cada momento y cumplir con las diferentes funcionalidades de forma modular y organizada. Además de estas actividades y sus fragmentos, también cuenta con una base de datos Room que se utiliza para almacenar las alarmas programadas.

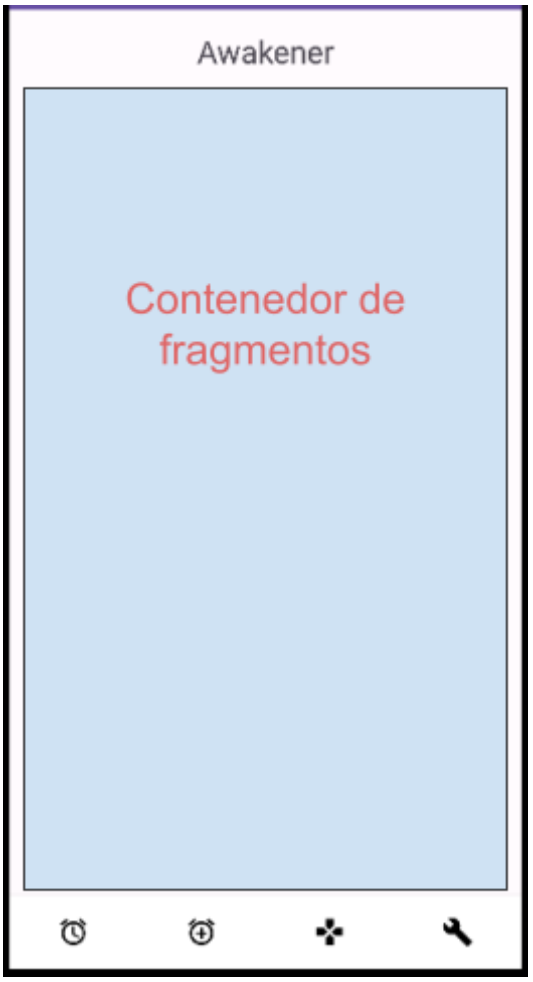

Imagen 1. Estructura de MainActivity

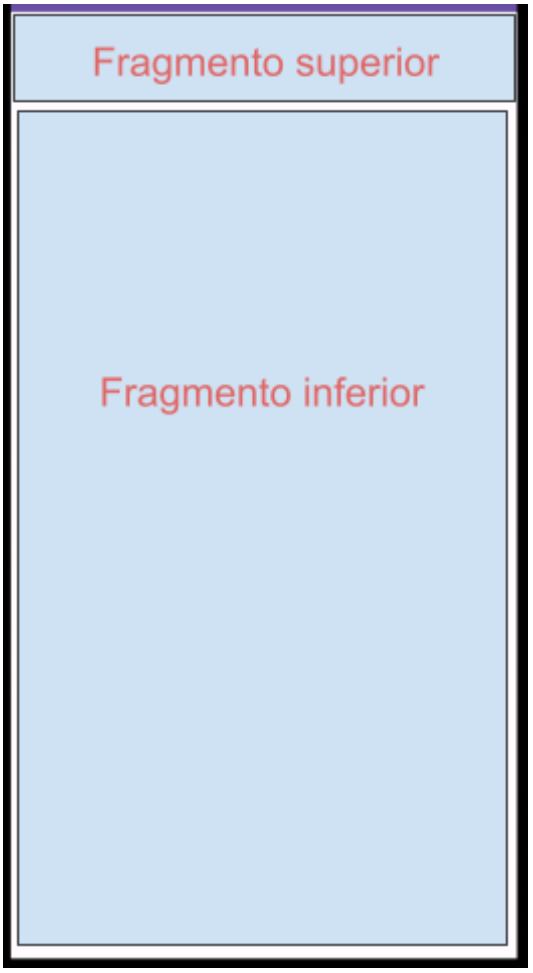

Imagen 2. Estructura de AlarmActivity

Clases

Relacionadas con la base de datos

AlarmDatabase

Esta clase se encarga de implementar la base de datos Room. Tiene asociado un DAO (Data Access Object) con el que podemos realizar las operaciones necesarias en la base de datos. La usamos para almacenar las alarmas y su información.

Alarm

Es la entidad que usamos para almacenar las alarmas en la base de datos. Tiene los siguientes atributos:

- **id**: El identificador único de la alarma, se incrementa de forma automática.
- **time**: La hora programada de la alarma, en formato String HH:MM.
- **name**: El nombre de la alarma.
- **type**: Un número que representa el tipo de alarma que es (Clásica, Adivina el número…).

AlarmDao

Es una interfaz que nos permite ejecutar sentencias SQL sobre la base de datos usando métodos. Se utiliza tanto para insertar y eliminar alarmas, como para acceder a su información.

Relacionadas con MainActivity

MainActivity

Es la actividad principal, está compuesta por: El título de la aplicación en la parte superior, una barra de navegación con 4 botones en la parte inferior, y un contenedor de fragmentos en el centro, como se puede observar en la **Imagen 1.** En función del botón de la barra inferior que pulsemos, se mostrará un fragmento u otro en el contenedor central.

Al ejecutar la aplicación o pulsar el primer botón “Home”, se muestra el fragmento **AlarmListFragment**, que muestra una lista con todas las alarmas programadas. Si no hay ninguna alarma programada, en su lugar, MainActivity mostrará un mensaje indicándolo. Al mantener pulsada una alarma de la lista, se abre un diálogo de confirmación para eliminar esa alarma de la base de datos y desprogramarla. Cuando esto sucede se genera un mensaje Toast indicándolo.

Al pulsar el segundo botón “AddAlarm”, se reemplaza el fragmento actual por **AddAlarmFragment**, que permite al usuario introducir información para programar una

alarma nueva. Cuando se programa una alarma nueva se muestra un mensaje Toast indicándolo.

Al pulsar el tercer botón “Activities”, se reemplaza el fragmento actual por **ButtonsFragment**, que contiene botones para lanzar los distintos tipos de alarmas en modo de prueba.

Finalmente, al pulsar el cuarto botón “Settings”, se reemplaza el fragmento actual por **SettingsFragment**, dónde el usuario puede cambiar la configuración de la aplicación.

AlarmInfoFragment

Este fragmento se encarga de mostrar la lista de alarmas. Carga las alarmas programadas de la base de datos y las muestra utilizando un RecyclerView y el adaptador **AlarmAdapter**.

AlarmAdapter

Este adaptador maneja el RecyclerView para mostrar las alarmas. También se encarga de “avisar” a MainActivity cuando se mantiene pulsada una alarma, para que gestione la lógica de eliminarla.

AddAlarmFragment

Es el fragmento que permite al usuario agregar una nueva alarma. Está compuesto por: un EditText en la parte superior, que permite establecer el nombre de la alarma; un TextView que muestra la hora a la que se programará la alarma, por defecto se inicializa a la hora actual más un minuto, pero si se hace click en este TextView se abre un diálogo de selección de hora; un botón que abre un diálogo para elegir el tipo de alarma y un TextView que muestra el tipo seleccionado; y un botón para añadir la alarma con los parámetros establecidos. Al pulsar este botón se programa la alarma a la hora deseada y se añade a la base de datos.

ButtonsFragment

Este fragmento contiene un botón para cada tipo de alarma, al pulsar un botón se abrirá la actividad correspondiente a ese tipo de alarma en modo prueba. Permite al usuario probar los distintos tipos de alarmas sin tener que programarlas.

SettingsFragment

En este fragmento se muestran las opciones de configuración de la aplicación. De momento permite seleccionar el idioma mediante un Spinner. Al pulsar el botón de confirmar, se aplica la configuración y se guarda de forma permanente en el archivo de preferencias. Esto hace que la configuración elegida por el usuario se mantenga entre una sesión y otra.

Relacionadas con AlarmActivity

AlarmActivity

Es la actividad que se abre cuando “suena” una alarma (o al usar el modo de prueba). Está compuesta por dos contenedores de fragmentos: uno superior, el de información; y otro inferior, el de control (**Imagen 2.**). Además, si se abre por una alarma, también reproduce un sonido, hace que el móvil vibre, lanza una notificación con la información de la alarma, borra la alarma de la base de datos, y gestiona el bloqueo de pantalla para que suene aunque el dispositivo esté con la pantalla apagada.

Dependiendo de cómo se haya abierto la actividad, los contenedores tendrán unos fragmentos u otros:

En el contenedor de información (el superior) irá **AlarmInfoFragment** si se abre como consecuencia de una alarma, y **TestInfoFragment** si se abre en modo prueba.

En el contenedor de control (el inferior) irá el fragmento correspondiente al tipo de alarma: **AlarmControlFragment** si es una alarma clásica, y **GuessGameFragment** si es del tipo “Adivina el número”.

AlarmReceiver

Este receptor se encarga de recibir el aviso de cuando una alarma programada tiene que sonar (aunque la aplicación no esté en marcha), y abre **AlarmActivity** para que se gestione la alarma.

AlarmInfoFragment

Este fragmento se muestra cuando se activa una alarma, y muestra el nombre y la hora de la alarma.

TestInfoFragment

Este fragmento se muestra cuando se abre una prueba, tiene un botón de retorno para salir de la prueba sin completarla. También muestra el nombre de la prueba.

AlarmControlFragment

Es el fragmento de control para la alarma clásica, cuenta con un botón para apagarla y volver a la actividad principal.

GuessGameFragment

Es el fragmento del primer minijuego: “Adivina el número”. Se genera un número aleatorio del 0 al 999, y el usuario debe adivinarlo para apagar la alarma. Después de cada intento se ofrece una pista indicando si el número a adivinar es mayor o menor que el proporcionado. Cuando el usuario acierta el número se abre un diálogo felicitándole, y permite apagar la alarma para volver a la actividad principal.

Requisitos cumplidos

- Se usa un RecyclerView en el fragmento AlarmInfoFragment para mostrar las alarmas programadas.
- Se utiliza una base de datos local Room para almacenar las alarmas programadas.
- Se usan distintos tipos de diálogos en la aplicación (diálogo de confirmación al eliminar alarma, diálogo de selección de hora al añadir alarma, diálogo de selección de lista al elegir tipo de alarma, etc…).
- Se utilizan notificaciones locales cuando suena una alarma.

- Se controlan los giros guardando la información de los fragmentos que tengan distintos estados y restaurándola cuando es necesario. También se guardan los fragmentos actuales de cada actividad para que aunque se gire el dispositivo, se mantengan los mismos fragmentos.
  Nota de interés: Los cambios de fragmentos se hacen mediante una transacción con FragmentManager, esta transacción no es atómica y, si queremos comprobar algo después de que se cambie un fragmento, puede que la comprobación ocurra antes de que se efectúe la transacción. Para solucionar esto hay que usar runOnCommit() para ejecutar la comprobación después de que finalice la transacción.

- Se hace uso de fragmentos por toda la aplicación para cambiar el comportamiento de las actividades en función de parámetros como el tipo de alarma, o el botón presionado en la barra de navegación.
- La aplicación tiene capacidad multi idioma, y está traducida al Inglés, Castellano, y Euskera.
- Se utilizan preferencias para hacer que la selección del idioma sea permanente y se mantenga entre sesiones.
- La aplicación cuenta con una barra de acciones que permite navegar por los diferentes menús de la actividad principal.

Posibles mejoras

- Añadir más minijuegos. La idea era ofrecer una selección de varios minijuegos, pero debido a otras complicaciones que han surgido durante el desarrollo de la aplicación, no hemos podido implementar más minijuegos para esta entrega. Aún así, el diseño modular de la aplicación facilita el añadir nuevos minijuegos en el futuro.
- Mejorar la interfaz visual para hacer que la experiencia del usuario sea más fluida.
- En algunas versiones altas de android (API 32) hay problemas al enviar notificaciones y al hacer que la alarma suene cuando la aplicación no está abierta. Aunque se han probado distintos métodos para intentar solucionar este problema, aún no está corregido del todo.
- Añadir más opciones de configuración.
- Añadir funcionalidades extra a las alarmas (Por ejemplo, que se repita cada X días).