UNIVERSIDAD DE OVIEDO

ASTURIAS
CAMPUS DE EXCELENCIA
INTERNACIONAL
AD FUTURUM

MÁSTER UNIVERSITARIO EN INGENIERÍA WEB

TRABAJO FIN DE MÁSTER

"Notificación accesible de eventos en entornos de Inteligencia Ambiente"

SARA GARCÍA PÉREZ

El tutor: Bernardo Martín González autoriza la defensa del Trabajo Fin de Más

Oviedo, JUNIO de 2014

Resumen

El boom de los dispositivos móviles inteligentes o smart phones viene acompañado de un crecimiento en el número de usuarios que descubren la utilidad de las aplicaciones que se pueden instalar en éstos. Desde aplicaciones de entretenimiento y ocio hasta aplicaciones que nos facilitan la vida o el trabajo.

Este proyecto surge de la necesidad de aprovechar la tecnología para ayudar a personas con una discapacidad, en este caso, una deficiencia auditiva.

El sistema desarrollado es una aplicación sensor/receptor. Se podrá activar el modo sensor en un dispositivo y recibir notificaciones en otro dispositivos mediante un sistema de sincronizacion. Este sistema se implementa utilizando la API de Google Calendar.

Una de las principales ventajas de este sistema es que, permite al usuario interactuar con el entorno supliendo esa falta de audición. Así, de esta manera, el usuario se sentirá reconfortado. Por ejemplo, si deja un dispositivo al lado de la puerta y otro con él, bien en un lugar visible o pegado a su cuerpo, para sentir la vibración del dispositivo, podrá saber cuando alguien está llamando a la puerta.

Palabras Clave

Ingeligencia Ambiente, Dispositivos móviles, Android, Google APIs, Deficiencia auditiva

Abstract

The boom of smart phones comes with an increase in the number of users who discover the usefulness of the applications that can be installed on these. From entertainment and leisure applications to applications that improve our lives or work.

This project emerges from the need to leverage technology to help people with a disability, in this case, hearing problems.

The system is a sensor/receiver application.You can active the sensor in a device and receive notifications in other devices by a synchronization system. This system is implemented using the Google Calendar API.

One of the main advantages of this system is that it allows the user to interact with the environment by supplying the lack of hearing. This way, the user will feel comforted. For example, if you leave a device next to the door and another device comes with you, attached to your body, you can know when someone is knocking at the door because of the device vibration.

Keywords

Ambient Intelligence, mobile devices, Android, Google APIs, hearing problems

Índice general

Índice de figuras

Capítulo 1

Introducción

El boom de los dispositivos móviles inteligentes o smart phones viene acompañado de un crecimiento en el número de usuarios que descubren la utilidad de las aplicaciones que se pueden instalar en éstos. Desde aplicaciones de entretenimiento y ocio hasta aplicaciones que nos facilitan la vida o el trabajo.

El objetivo de este proyecto es desarrollar una aplicación que capture y notifique eventos, dependiendo del modo elegido, y sincronice éstos entre diferentes dispositivos a través del Google Calendar.

1.1. Motivación del proyecto

Este proyecto surge de la necesidad de aprovechar la tecnología para ayudar a personas con una discapacidad, en este caso, una deficiencia auditiva. A parte de esto, podría servir también como sistema de seguridad o incluso como walkie talkie de bebé.

Una de las principales ventajas de este sistema es que, permite al usuario interactuar con el entorno supliendo esa falta de audición. Así, de esta manera, el usuario se sentirá reconfortado. Por ejemplo, si deja un dispositivo al lado de la puerta y otro con él, bien en un lugar visible o pegado a su cuerpo, para sentir la vibración del dispositivo, podrá saber cuando alguien está llamando a la puerta o si hay ruidos en el pasillo.

Por otro lado, serán más independientes. Por ejemplo, si viven solos, podrán irse al salón mientras que la comida se hace en el horno y cuando suene el aviso del temporizador, el sensor lo captará y aparecerá el aviso en el dispositivo que lleva consigo el usuario.

La motivación principal de este proyecto consiste en utilizar tecnologías como android y la API de Google para sincronizar dispositivos móviles ofreciendo un servicio al consumidor. Estos dispositivos pasan a formar parte de las actividades diarias en la vida de los usuarios, aportando un valor añadido y supliendo una necesidad.

1.2. Estructura de la documentación

Esta documentación se encuentra dividida en 14 secciones descritas brevemente a continuación.

1.2.1. Introducción

Es la sección presente. En ésta, se describe la motivación del proyecto y se presenta un breve resumen del contenido de la documentación.

1.2.2. Estado del arte

En esta sección se analizan soluciones existentes similares a la que se llevara a cabo a fin encontrar el punto diferenciador entre nuestra aplicación y el resto.

1.2.3. Análisis

Esta sección tiene como objetivo describir, de la manera más exhaustiva posible, qué se va a desarrollar. En esta sección se presenta el catálogo de requisitos de la aplicación.

1.2.4. Modelo de datos

Se describe cómo se almacenan y manejan los datos con los que trata la aplicación.

1.2.5. Diseño del sistema

En esta sección se detalla la forma en que se va a construir la aplicación. Al final de ésta, se presenta una lista en que se detalla en qué puntos del diseño se satisfacen los requisitos presentados en el capítulo de análisis.

1.2.6. Pruebas de software

En esta sección se describen las pruebas unitarias, de integración y de sistema realizadas en el proceso de desarrollo del proyecto.

1.2.7. Pruebas de usabilidad

Aquí se detallan las pruebas hechas con el fin de mejorar la usabilidad de la aplicación,así como los principales errores encontrados y soluciones llevadas a cabo.

1.2.8. Pruebas de accesibilidad

Aquí se detallan las pruebas hechas con el fin de conocer y asegurar el nivel de accesibilidad de la aplicación.

1.2.9. Presupuesto y Plan Temporal

En esta sección se incluye la planificación temporal de las tareas a realizar y el presupuesto.

1.2.10. Manual del programador

Este capítulo contiene información útil para un programador que pretenda ampliar o mejorar la aplicación.

1.2.11. Manual de internacionalización

En esta sección se indica a personal no informático como traducir la aplicación a otros idiomas.

1.2.12. Manual de instalación

En esta sección se indica como generar el instalable para poder utilizarlo en un dispositivo Android.

1.2.13. User's Manual

Este es el manual de usuario. Describe la forma de realizar las tareas en la aplicación.

1.2.14. Conclusiones y ampliaciones

Este capítulo contiene las conclusiones del desarrollo de este proyecto y algunas propuestas de futuras mejoras y ampliaciones al mismo.

Capítulo 2

Estado del arte

Este proyecto se engloba dentro del campo de la Inteligencia Ambiental o Ambient Intelligence (AmI). Éste, se basa en la obtención de información del entorno a través de diferentes sensores. A continuación se exponen diferentes proyectos similares a éste y se analizan las ventajas y desventajas.

2.1. A Support System for Context Awareness in a Group Home using Sound Cues

Se trata de un sistema de notificación mediante audio, que indica la situación en tiempo real de personas en una casa sin supervisión visual.

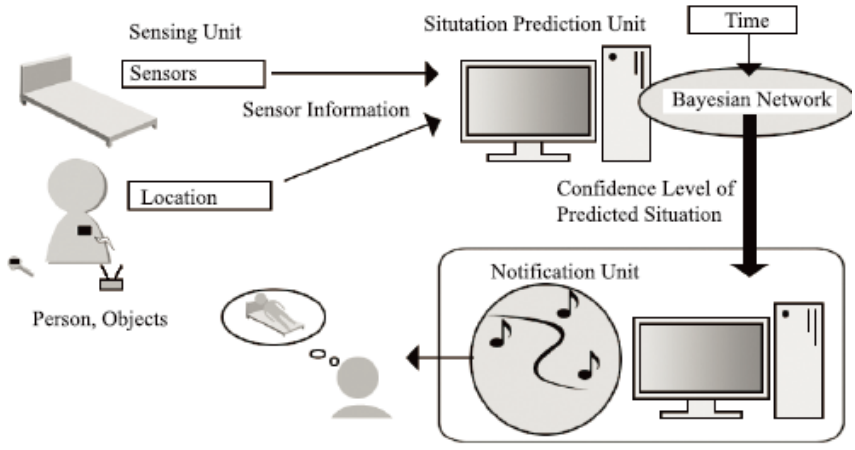

Figura 2.1: Componentes del sistema

Este proyecto se basa en la predicción de eventos en un entorno real y su notificación usando sonidos. La función de predicción utiliza una red Bayesiana e información de sensores. La función de notificación informa a los receptores de la situación predicha y del nivel de confianza de la predicción mediante pistas de sonido.

Los cuidadores de personas con discapacidad intentan entender los sonidos provenientes de éstas personas mientras están atendiendo a otra persona. El caso es que no siempre son capaces de diferenciar lo que es urgente de lo que no, sobre todo si son cuidadores inexpertos. Este sistema intenta ayudar en ese aspecto. El sistema está formado por 3 unidades: unidad *sensing*, unidad *situation prediction* y unidad *situation notification*.

La unidad *sensing* detecta el estado de una persona en tiempo real. La unidad *situation prediction* estima la situación de la persona usando una red Bayesiana basándose en la información de la *sensing*. La unidad *situation notification* informa al usuario de la situación a través de pistas de sonido.

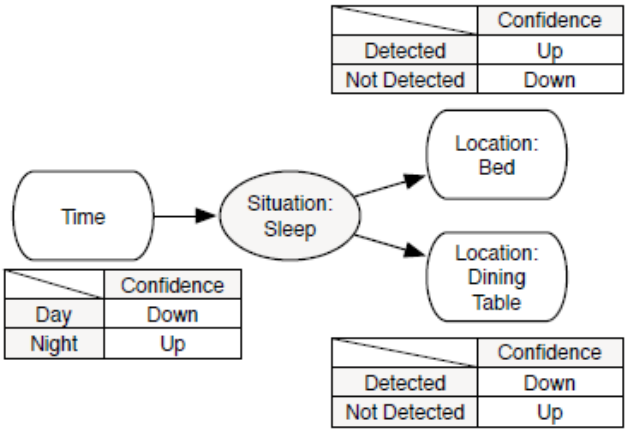

Figura 2.2: Ejemplo de red Bayesiana.

Se usa una red Bayesiana para predecir la situación de la persona. Es un tipo de modelo probabilístico gráfico. La red representa la probabilidad teniendo en cuenta un conjunto de variables independientes. Depende del comportamiento de las personas en el entorno de una vivienda. Se requieren algunas evidencias para poder predecir la situación de una persona. Como un ejemplo de *localización*: *tomarse un baño* está relacionado directamente con *una bañera*. Como ejemplo del *estado de un objeto*, *una persona tiene hambre o muestra interés en una comida* está directamente relacionado con *abrir y cerrar el frigorífico o un armario de la cocina*.

La figura 2.2 muestra una red Bayesiana para *Dormir*. Si el *Tiempo* es *Medianoche*, el nivel de confianza para *Dormir* aumentará. Además, si el individuo está en la cama, el nivel de confianza aumentará mucho más.

La unidad *situation notification* traduce la situación de una persona y su nivel de confianza en pistas de sonido. Provee información actual del contexto relacionada con la persona a los receptores. Los sonidos transmitidos están relacionados con eventos que ocurren en el mundo real. Por ejemplo, un timbre cuando la persona tiene una visita u otro tipo de alertas.

No es fácil para cuidadores con poca experiencia percibir el sonido deseado de una retransmisión de audio mientras llevan a cabo sus tareas.

Ventajas	Desventajas
▪ Ayuda a personas mayores. ▪ Ayuda a cuidadores a monitorizar el estado de las personas mayores y a desechar situaciones que no son de emergencia.	▪ No sirve de ayuda en la prevención de caídas. ▪ Algunas pistas de sonido pueden no ser adecuadamente interpretadas. ▪ Coste alto en la implantación del sistema. ▪ No dispone de otro tipo de aviso distinto al auditivo, como el visual. ▪ Sistema poco flexible para implantarlo en cualquier hogar o centro.

2.2. SWEET HOME

SWEET HOME es un proyecto de domótica basado en comandos de voz. El proyecto tiene como objetivo el diseño de una nueva casa inteligente, focalizándose en 3 aspectos principales: proveer asistencia via una interacción natural hombre - máquina(voz y comandos táctiles), para facilitar las relaciones sociales y para velar por la seguridad detectando situaciones desafortunadas.

La entrada del sistema está compuesta por información transmitida por el sistema de domótica mediante red local y la información de los micrófonos transmitida a través de canales de radiofrecuencia.

La información del sistema domótico es símbólica, mientra la de los micrófonos debe ser procesada para extraer información sobre lo que se dice y los sonidos. Esta extracción está basada en AUITHIS system. Este es un sistema en tiempo real y multihilo de procesamiento de audio para entornos ubicuos.

Las salidas del sistema incluyen ordenes de domótica, pero también la interacción con el usuario en caso de que una orden vocal no se hubiera entendido o en el caso de mensajes de alerta.

El sistema debe además facilitar la conectividad con los familiares, fisioterapeutas o cuidadores. Esto se consigue usando los sistemas e-lio y Visage.

La domótica permite a las personas mayores mantener un control sobre su entorno y sus actividades para mejorar su autonomía.

La tecnología basada en audio tiene un buen potencial para convertirse en una de las mejores modalidades en la interacción dentro de una casa inteligente.

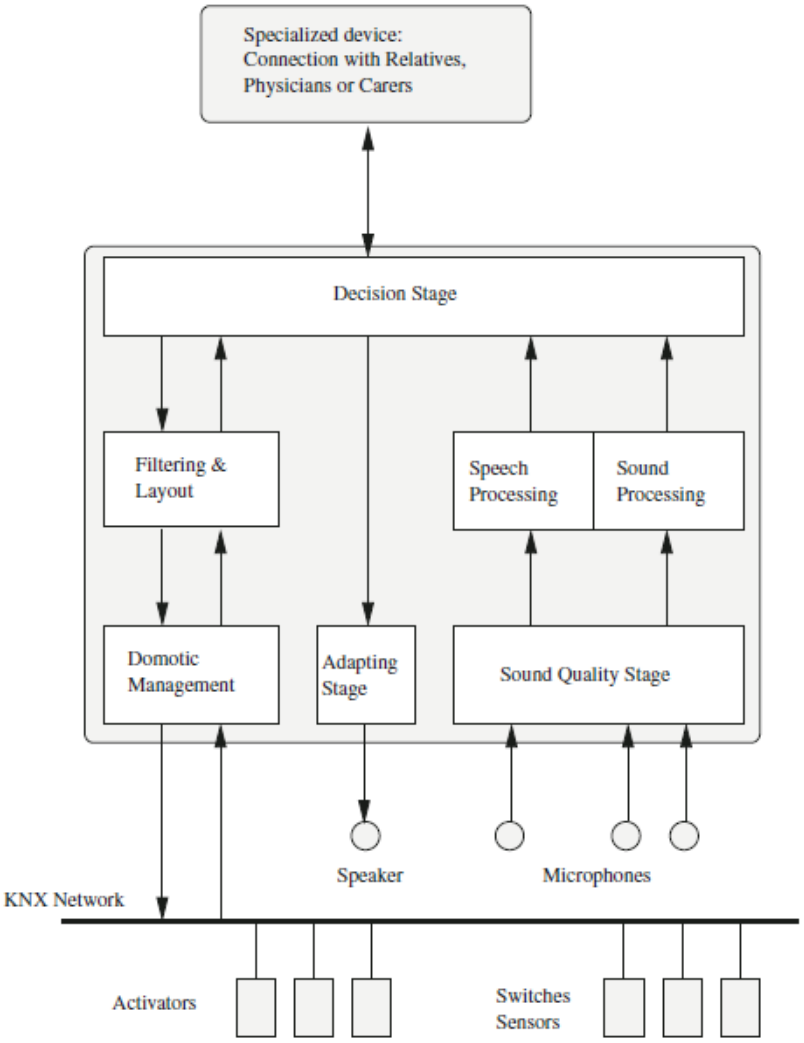

Figura 2.3: Diagrama de Sweet-Home

El sistema puede incluso capturar sonidos, tales como dar palmas para controlar la luz, un ejemplo muy conocido.

El sistema usa tecnologías y aplicaciones estandarizadas. Los standards aseguran la compatibilidad entre componentes y facilitan el mantenimiento a la vez que permiten soluciones de diseño más baratas.

Ventajas	Desventajas
▪ Control de la casa mediante bus KNX (luz, calefacción, etc). ▪ Prevención de accidentes ▪ Recordatorio de fechas importantes, horas de medicación, etc ▪ Video - conferencia ▪ Si ocurre un accidente, la persona puede llamar para pedir ayuda mediante un comando de voz (manos libres) ▪ Bastante adaptable gracias a la estandarización	▪ Las pruebas sólo se pueden realizar en una casa-prototipo. La flexibilidad de nuestro sistema permite probarlo en distintos escenarios ▪ Uno de los sistemas puede fallar, por ejemplo el de reconocimiento de voz ▪ Coste alto en la implantación del sistema. ▪ Sistema poco flexible para implantarlo en cualquier hogar o centro. ▪ El sistema no envía mensajes automáticos en posibles emergencias. ▪ Alertas sólo en modo voz, no existe otro tipo de interacción

2.3. CompanionAble

Se trata de un proyecto con los siguientes objetivos:

- combinar las capacidades de un robot con las funcionalidades estáticas de un entorno inteligente.
- integrar la información de los sensores con la capturada por el robot
- permitir la comunicación fácilmente de la persona con sus familiares y su médico
- mejorar la calidad de vida y la autonomía de las personas dependientes.

En este proyecto se lleva a cabo un sistema de reconocimiento de sonidos y de comandos de voz. Lo que pretende llevar a cabo nuestro prototipo cuando poseamos un servicio Web de reconocimiento de voz.

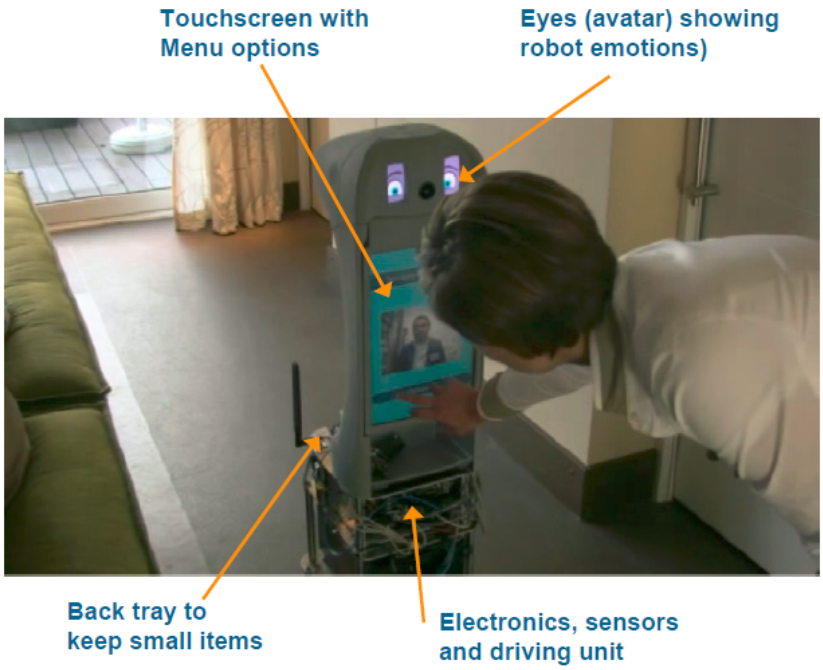

Figura 2.4: Robot CompanionAble

El sonido está siendo recogido continuamente por 2 sistemas paralelos: uno de ellos atribuye un tipo al sonido (palabras/sonido y tipo de sonido) y el otro transcribe las palabras en texto. Éstos utilizan el protocolo TCP/IP.

Los comandos de sonido reconocidos son transmitidos al servidor del proyecto via un protocolo SOAP. El reconocimiento vocal es filtrado por el módulo de reconocimiento para evitar las falsas alarmas.

Ventajas	**Desventajas**
▪ Asistencia mediante reconocimiento de sonidos ▪ Robot simula compañía	▪ Usa servidor propio (necesita mantenimiento, etc) ▪ Algunas personas podrían considerarlo un sistema intrusivo

2.4. ISISEMD

Este proyecto nació en marzo de 2009 con el objetivo de mejorar la calidad de vida de las personas mayores ofreciendoles servicios inteligentes.

De acuerdo al reporte de Alzheimer en el mundo, la demencia va a incrementar hasta en un 40 % dentro de 20 años dado el envejecimiento de la población.

Actualmente, éstos viven siendo cuidados por sus familias, lo que es muy estresante para éstas. Existen un montón de riesgos para estas personas, por eso los servicios inteligentes son muy importantes para ayudarles tanto a ellos como a sus familias.

Los servicios ISISEMD han sido diseñados inicialmente para 3 tipos de usuarios finales: las personas mayores con demencia o con un leve deterioro cognitivo, sus cuidadores informales (pareja, la familia cercana o vecinos) y los cuidadores formales.

La tecnología debe ser practicamente invisible si no invisible totalmente para los usuarios. Esto significa que será requerido el mínimo de interacción posible.

Un portal web funciona como un punto de entrada común para el sistema ISISEMD. Además es el componente responsable de la gestión de los usuarios así como de la asociación de usuarios con servicios y otros usuarios. Éste contiene subcompontentes funcionales: módulo *Gestión de usuarios*, módulo de Autenticación y autorización, módulo de *Logging* y módulo de *Informes*. Del lado del usuario, la unidad central es un ordenador con una pantalla táctil: *Carebox*.

2.4.1. Plataforma ISISEMD

La plataforma integra diferentes tecnologías: Hewlett Packard (Italia), Alcatel-Lucent (Italia), Converge Soluciones TIC (Grecia), Eltronic A / S (Dinamarca) y Socrate Medical (Italia).

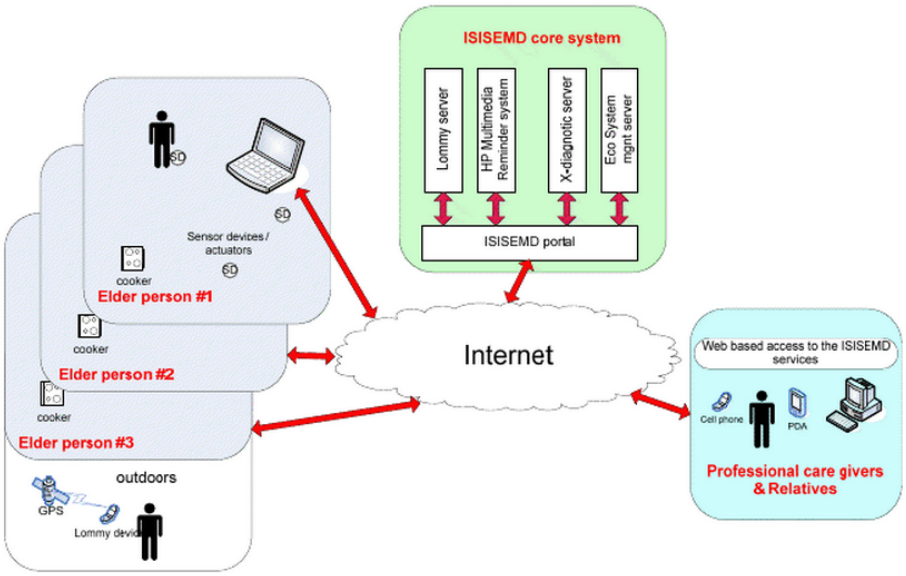

Figura 2.5: Diagrama de arquitectura de la plataforma ISISEMD

Uno de los objetivos de la plataforma ISISEMD es garantizar la integridad y extensibilidad de la misma en un futuro, por esto, se basa en una Arquitectura Orientada a Servicios (SOA).

Los servicios basados en SLA (service level agreement) son la clave de la automatización y dinamismo de la plataforma ISISEMD. Las especificaciones WSDM (Web Services Distributed Management) y la gestión de los WS (Servicios Web) sólo se centran en algunas propiedades de la calidad de servicio (QoS).

Las aplicaciones ISISEMD necesariamente necesitan una red de servicios para su implementación. Los flujos de datos o control pueden ser complejos y requieren un soporte extra para los aspectos e-care y las limitaciones.

ISISEMD hace un uso intensivo de los nuevas estándares de servicios Web, incluyendo las áreas de descripciones semánticas de servicios, SLAs y acuerdos, flujo de trabajo y la orquestación, la gestión, la confianza y la seguridad, transporte y mensajería. El uso de estándares tiene el potencial para mejorar la interoperabilidad, pero a menudo lo hace a expensas de comprometer la especificación.

Los servicios para las personas mayores que se incluyen son:

- seguridad para la casa y la persona
- recordatorios de tareas diarias
- servicios de estimulación cognitiva
- comunicación de video con cuidadores
- servicio de localización cuando la persona está fuera de la casa
- botón de contacto de emergencia en la pantalla táctil del Carebox
- botón de contacto de emergencia fuera de casa en el dispositivo GPS

Los servicios para los cuidadores formales e informales incluyen:

- alertas
- notificaciones
- servicios de alarma
- visión de actividades diarias en el portal
- servicio de video-llamada para comunicarse con las personas mayores
- información sobre patrones de estilo de vida
- servicio de doctor remoto

Esta tecnología ofrece una sensación de seguridad y además puede ser usada por cualquier tipo de usuario, por ejemplo una persona que cuida de su pareja puede ser también una persona mayor.

Para los cuidadores formales ofrece la ventaja de ahorrarse tiempo de viaje para realizar visatas innecesarias a los clientes. Ellos pueden comunicarse remotamente con los clientes y de esta manera tienen más tiempo para cuidar de más clientes.

Service type	Service name	Validation status
Home safety	Cooking monitor	Validated during the real-life pilot operation
	Smoke/ Fire detection alarm	
	Kitchen/Bathroom flood detection	
	Fridge door alarm	
	Leaving bed during night for long time	
	Wake up sensor	
	Intelligent front door	
Structure of the day and contact to informal care-givers in case of emergency	To Do List, Calendar, Time/Date	Validated during the real-life pilot operation
	Help/Contact request on the touch screen	
Cognitive stimulation	Brain Games	Validated during the real-life pilot operation
	Memory Lane	
Communication with care-givers	Videophone	Partially validated during the real-life pilot operation
Communication with health and care professionals	Remote Doctor	Demo evaluation
	Medication Manager	Not validated during the real-life pilot operation
Outdoor safety	Outdoor positioning	Validated during the real-life pilot operation
	Panic button with outdoor position	
	Fall alarm outdoor	
Professional care-givers support	Lifestyle Pattern	Partially validated during the real-life pilot operation

Figura 2.6: Lista de servicios de ISISEMD

2.4.2. Inteligencia en la plataforma

Los servicios son proactivos, de manera que pueden definir unas circunstancias críticas que podrían ocurrir antes de que ocurren y entonces intentar anticiparse a las consecuencias (por ejemplo, la familia de una persona recibirá una notificación si la persona está cocinando por más tiempo del esperado, anticipandose a un posible fuego en la casa). Esta inteligencia es posible de implementar gracias al sistema Ecosystem.

Servicios de seguridad en el hogar

La parte Ecosystem del Carebox tecnicamente consiste en una máquina virtual que corre un instancia local de un servidor *Ecosystem Domotics*, el cual conforma un conjunto reducido de servicios y procesos para unos requisitos específicos. Este servidor es responsable de monitorizar y responder a eventos. Un servicio especial instalado en cada módulo domótico es responsable de retransmitir estos eventos hacia el proceso concentrador central X-Server para ser evaluados.

Un ejemplo de servicio domótico es el control en la cocina, el evento es lanzado con la acción de encender la placa de la cocina. Esto activa un evento que activa los tempo-

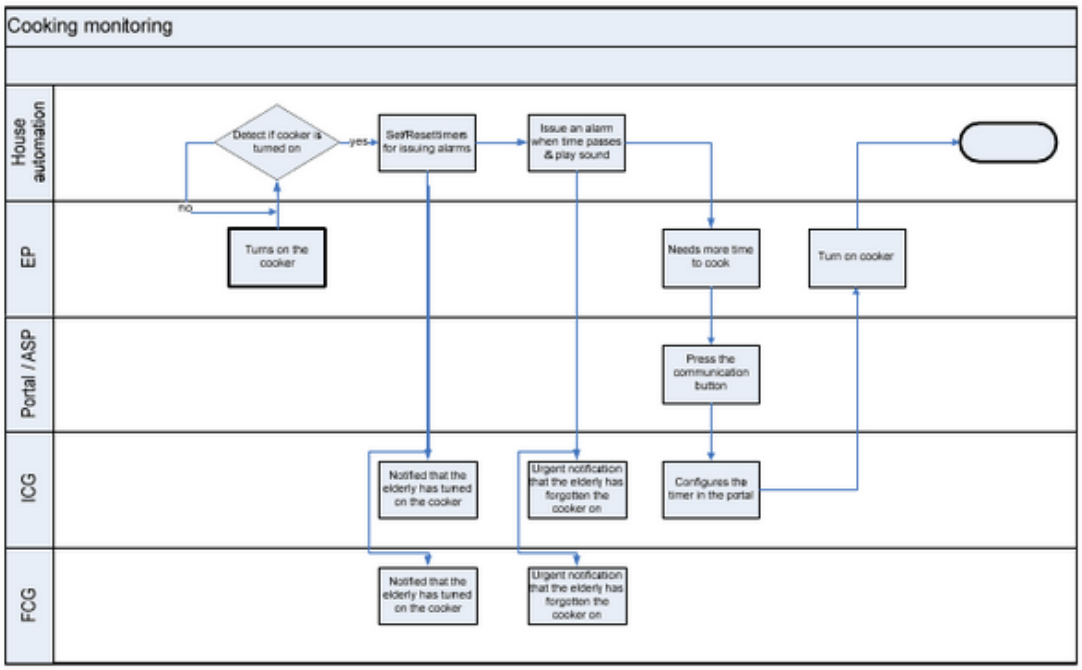

Figura 2.7: Workflow del servicio de cocina

rizadores apropiados que monitorizarán el tiempo que pasa la placa encendida.

Los servicios están configurados y corriendo sin irrumpir en la vida cotidiana. Cabe destacar que el cuidador podrá ver el estado de las diferentes tareas actualizado en el portal.

Si la cocina ha estado encendida por un buen rato el proceso será el siguiente:

1. El sistema enviará la información conteniendo un mensaje de alerta al cuidador (sms/email) - Al mismo tiempo, enviará un mensaje de voz a la pantalla del Carebox y un mensaje de alerta parpadeante.

2. Si el sistema detecta que no ha habido respuesta por parte de la persona mayor, se enviará una alarma al cuidador.

El resto de sistemas de seguridad funcionan de una manera similar. Por ejemplo el servicio del frigorífico detecta el tiempo que éste permanece abierto. La notificación se repite hasta que el frigorífico esté cerrado.

El servicio de *cama*, además de detectar que la persona permanece demasiado tiempo en la cama, ayuda a identificar patrones de sueño.

Carebox

Las motivaciones en la elección de este tipo de dispositivo son varias:

1. La fácil interacción. Es difícil para gente con problemas cognitivos aprender a usar nuevas tecnologías.

2. La necesidad de interacción debe ser reducida al mínimo.

3. Si las interacciones no pueden ser eliminadas, deben ser lo más imples e intuitivas posibles

Carebox cumple esos requisitos de esta manera:

1. La forma, dimensiones y aspecto de éste son idénticos a los de un mando de televisión. Esto significa que se puede introducir su uso de una manera natural.

2. El dispositivo puede proveer diferentes niveles de interacción dependiendo del usuario final.

3. La pantalla táctil es la tecnología más sencilla e intuitiva, requiere un muy bajo entrenamiento y bajas capacidades cognitivas.

Desde un punto de vista funcional, Carebox tiene los siguientes roles:

- Muestra mensajes al usuario. Decide cuando un mensaje debe ser mostrado basándose en los parámetros de configuración.
- Permite al usuario pedir ayuda a los cuidadores presionando un botón
- Estimula las capacidades de la persona mostrando información como la fecha y el día, fotos importantes para ellos, etc.

El módulo inteligente del dispositivo está constantemente en comunicación con el módulo del servidor central, considerando constantemente el contexto del usuario.

- Durante la noche el sitema entra modo ahorro de energía, desactivando la pantalla. Si se detecta movimiento, la pantalla se activa otra vez.
- Los mensajes de alarma multimedia relacionados con condiciones peligrosas en la casa son mostrados en el dispositivo. De esta manera la persona puede reaccionar ante la situación.
- En caso de condiciones externas que temporalmente previenen al módulo de proveer alguna funcionalidad, se cambia el layout mostrando los mensajes de notificación apropiados que informarán de la no disponibilidad de uno o varios servicios. Al mismo tiempo, un módulo del servidor es capaz de detectar el problema y avisar al equipo de soporte.

A través del portal se pueden habilitar o deshabilitar servicios. O sea, el sistema es flexible en cuando a funcionalidades.

Ventajas	Desventajas
▪ Diferentes niveles de interacción y características de fácil interacción para contactar ayuda, confirmar acciones, arrancar juegos de estimulación cognitiva y recibir video-llamadas ▪ Sistema altamente customizado y que tiene en cuenta a todos los posibles usuarios ▪ Arquitectura extensible en el futuro ▪ Atiende todas las necesidades de las personas con demencia, su seguridad, su independencia, etc ▪ Incrementa la calidad de vida de las personas con demencia y su familia ▪ Preparado para ofrecer nuevos servicios a medida que aumenta la enfermedad	▪ Usa servidor propio (necesita mantenimiento, etc) ▪ Se necesitaría servicio de helpdesk ▪ Alto coste de la instalación

2.5. Oficina inteligente

En este área es cabe destacar la importancia del ahorro energético. Una prueba de ello es la oficina *inteligente* de la empresa Conetate (http://www.conetate.com) , en la cual, las luces exteriores se encienden o apagan según la previsión meteorológica en ese momento y que tiene como fuente de información meteorológica la página web http://www.eltiempo.es.

2.6. Sound detector

Se trata de una aplicación Android muy similar a la desarrollada en este proyecto[8]. Sound Detector detecta sonidos en un dispositivo y se lo comunica a otro dispositivo

mediante una llamada o un sms. Además ofrece la posibilidad de realizar una foto en el momento de la captura del evento de sonido. Se puede configurar la sensibilidad y avisa con un sms si el dispositivo se está quedando sin batería.

2.7. Conclusiones

La aplicación desarrollada sirve de gran ayuda en el día a día de una persona con demencia o bien con una deficiencia auditiva y también se puede utilizar en otros escenarios de Inteligencia Ambiental.

La mejora destacable que ofrece respecto al resto de sistemas de domótica o detección de sonido, es la utilización de Google Calendar, un sistema de alta disponibilidad y seguridad, además sin gastos adicionales dentro de una tarifa de datos o con conexión vía wifi a internet.

Podría tenerse en cuenta para futuras ampliaciones el valor añadido de enviar una foto en el momento de captura del sonido.

Otra gran ventaja es la integración de varios sensores y características en un solo dispositivo. Esto aporta una gran flexibilidad y potencia a nuestro sistema en un futuro cuando sea un sistema completo.

En el caso de urgencias cuando la persona está fuera de casa, podemos obtener la localización del propio dispositivo, ya que dispone de GPS.

En un futuro esperamos utilizar un servicio Web para el tratamiento del sonido. Así se sabría también el grado de urgencia del evento. Ya que puede ser simplemente la mascota que tiene hambre, es decir, una falsa alarma.

Éste servicio aún no está disponible, sin embargo, ya en el prototipo se ven las ventajas de nuestro sistema, mucho más flexible, que evita tener que realizar una dura labor de obras en nuestro hogar o centro para personas mayores o discapacitadas. Hay que tener en cuenta además, el aporte económico que ello conlleva. De esta manera, actualmente no sería viable en la mayoría de los casos montar un sistema domótico como los explicados anteriormente en esta sección.

Capítulo 3

Análisis

En este capítulo se presentarán de la forma más concreta posible los objetivos de la aplicación que se va a construir.

3.1. Objetivos y alcance del sistema

Se pretende desarrollar una aplicación de notificación de eventos en entornos de Inteligencia Ambiente para dispositivos móviles android.

Entre los principales objetivos del proyecto está conseguir sincronizar varios dispositivos a modo sensor/receptor a fin de capturar/notificar eventos en diferentes lugares. Se utilizará como base el sistema Google Calendar. Los eventos capturados se almacenarán en el calendario del usuario. El modo receptor estará constantemente comprobando si hay eventos nuevos. Si existe un nuevo evento se notificará al usuario.

El sensor permitirá que los usuarios asignen un nombre y una localización al dispositivo. Además,una de las principales características del sensor, será la posibilidad de graduar la sensibilidad de detección de sonido.

Una vez detectado un evento, se sincronizará automáticamente y se notificará al dispositivo o dispositivos activados para esa misma cuenta de Google. Se mostrará una entrada con el nombre del sensor que ha detectado el evento, su localización y la hora. Además, existirá la opción de visualizar un registro de eventos por localización.

3.2. Catálogo de requisitos

R.1 Requisitos funcionales

R.1.1 Requisitos funcionales generales

R.1.1.1 *Los eventos se capturarán desde un dispositivo móvil:* el dispositivo móvil debe contar con sistema operativo android con versión 2.2 o superior.

R.1.1.2 *Sincronización de dispositivos:* los dispositivos han de poder sincronizarse para la notificación de eventos independiente del lugar. La mejor opción es usar Google Calendar, ya que es un sistema de Google, un sistema estándar.

R.1.1.3 *Los dos modos (sensor y receptor) accesibles desde una misma pantalla:* un usuario podrá poner en funcionamiento los 2 modos desde una misma pantalla.

R.1.2 Requisitos del sensor

R.1.2.1 *El usuario podrá configurar el sensor:* un usuario podrá establecer un nombre y una localización para el dispositivo. Así como modificar la sensibilidad.

R.1.2.2 *Existirá información visual de detección de sonido:* cuando se detecte un sonido se visualizará una animación en la pantalla a fin de indicar que el sensor está funcionando.

R.1.3 Requisitos del receptor

R.1.3.1 *La notificación se mostrará en un primer plano:* cuando llegue una nueva notificación, será totalmente visible en la pantalla.

R.1.3.2 *Debe notificarse mediante ruido, vibración y efectos luminosos.*

R.1.3.3 *Debe mostrar información relevante:*los datos del sensor y el instante en que sucedió.

R.1.3.4 *Visualización de registro de eventos:*se debe poder visualizar un registro de los eventos.

R.2 Requisitos no funcionales

R.2.1 Usabilidad: Se incorporarán evaluaciones de usabilidad al proceso de desarrollo de la aplicación.

R.2.2 Estandarización: El desarrollo del sistema deberá estar siempre, en la medida de lo posible, apegado a estándares.

R.2.3 Mantenibilidad

R.2.3.1 La arquitectura de la aplicación deberá permitir independencia, física y lógica, entre las capas de la misma.

R.2.3.2 Deberán facilitarse, en la medida de lo posible, las posteriores modificaciones a la aplicación.

R.2.4 Escalabilidad: El sistema deberá estar preparado, en la medida de lo posible, para un crecimiento posterior.

R.2.5 Desempeño: En condiciones normales, el sistema no debería tardar más de 30 segundos en responder.

R.2.6 Seguridad: Utilización de OAuth.

R.2.7 Tecnología: La aplicación para el dispositivo móvil deberá desarrollarse para la plataforma Android.

3.3. Descripción de los actores

En el sistema se distingue un único rol de usuario. Este actor puede realizar cualquier operación disponible con un dispositivo móvil que tenga configurada una cuenta de Gmail y que esté conectado a internet.

3.4. Diagramas de casos de uso

Como ya se ha comentado con anterioridad, la aplicación dispone de 2 sistemas. En primer lugar se mostrará un caso de uso para la aplicación en conjunto y a continuación se mostrarán los casos de uso de cada uno de los sistemas.

3.4.1. Aplicación.

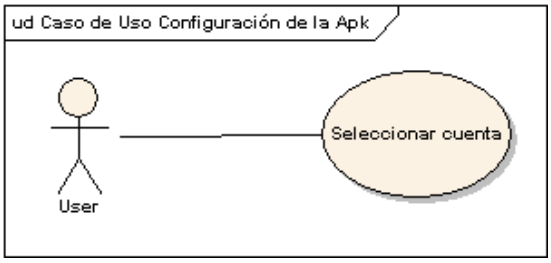

Figura 3.1: Diagrama de casos de uso de la configuración de la aplicación.

Casos de uso Configuración Aplicación		
Identificador	**Nombre**	**Requisitos cubiertos**
C.0.	Seleccionar cuenta	R.1.1.2

3.4.2. Sistema 1: sensor.

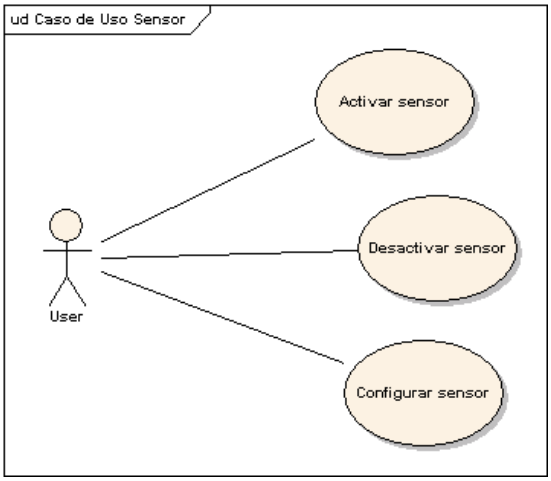

Figura 3.2: Diagrama de casos de uso del sensor.

Casos de uso sensor		
Identificador	**Nombre**	**Requisitos cubiertos**
C.1.1	Activar sensor	R.1.1.3, R.1.2.2
C.1.2	Desactivar sensor	R.1.1.3
C.1.3	Configurar sensor	R.1.2.1

3.4.3. Sistema 2: Receptor.

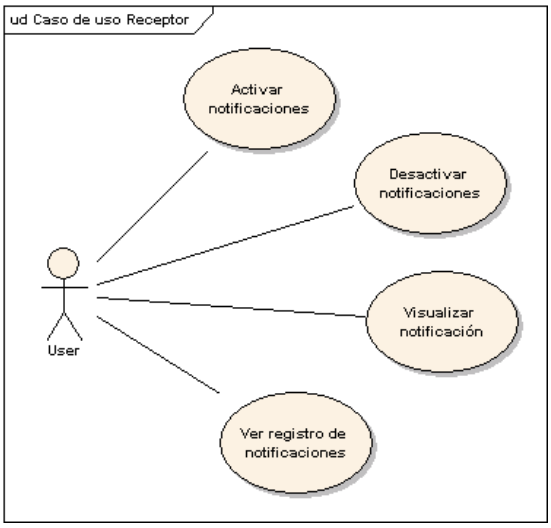

Figura 3.3: Diagrama de casos de uso del receptor.

Casos de uso receptor		
Identificador	**Nombre**	**Requisitos cubiertos**
C.2.1	Activar notificaciones	R.1.1.3
C.2.2	Desactivar notificaciones	R.1.1.3
C.2.3	Visualizar datos de la notificación	R.1.3.1, R.1.3.2, R.1.3.3
C.2.4	Ver registro de eventos	R.1.3.4

3.5. Descripción de escenarios

3.5.1. Configuración aplicación

Identificador E.0
Caso de uso C.0
Precondiciones Existir una cuenta en el dispositivo
Guión

1. Si existen varias cuentas de Gmail en el dispositivo se seleccionará una.
2. La cuenta seleccionada será la utilizada por la aplicación.

Excepciones Si sólo existe una cuenta, ésta será utilizada automáticamente por la aplicación.
Finalizado por Usuario.
Postcondiciones -

3.5.2. Sistema 1: Sensor

Activar sensor

Identificador E.1.1
Caso de uso C.1.1
Precondiciones Existir una cuenta en el dispositivo
Guión

1. Al seleccionar el botón ON, se activa la captura de sonido y el procesamiento de éste.
2. Si se detecta sonido se producirá una animación en la pantalla.

Excepciones -
Finalizado por Usuario: al desactivar el sensor.

Postcondiciones	-

Desactivar sensor

Identificador	E.1.2
Caso de uso	C.1.2
Precondiciones	Sensor activado
Guión	1. Seleccionar el botón de desactivado. 2. El sistema dejará de recoger el sonido.
Excepciones	-
Finalizado por	Sistema.
Postcondiciones	Sensor desactivado.

Configurar sensor

Identificador	E.1.3
Caso de uso	C.1.3
Precondiciones	-
Guión	1. Se hace clic en *Config.* 2. Primer opción, poner el nombre del sensor. 3. Segunda opción, poner el nombre de la localización. 4. Tercera opción, seleccionar la sensibilidad del sensor.
Excepciones	-
Finalizado por	Usuario.
Postcondiciones	Han cambiado los parámetros de configuración del sensor.

3.5.3. Sistema 2: receptor

Activar notificaciones

Identificador	E.2.1
Caso de uso	C.2.1
Precondiciones	Existir una cuenta en el dispositivo y tener un sensor activado
Guión	El sistema empieza a recoger los datos de Google Calendar.
Excepciones	-
Finalizado por	Usuario.
Postcondiciones	Si se detecta un ruido en el sensor correspondiente, deberá saltar una notificación.

Desactivar notificaciones

Identificador	E.2.2
Caso de uso	C.2.2
Precondiciones	-
Guión	El sistema deja de recoger los datos de Google Calendar.
Excepciones	-
Finalizado por	Sistema.
Postcondiciones	-

Visualizar notificación

Identificador	E.2.3
Caso de uso	C.2.3
Iniciado por	Aplicación.
Precondiciones	Hay evento en Google Calendar
Guión	Se muestra una notificación en la pantalla con sus datos.
Excepciones	-
Finalizado por	Usuario.
Postcondiciones	Nuevo evento en el registro.

Ver registro de notificaciones

Identificador	E.2.4
Caso de uso	C.2.4
Iniciado por	Usuario.
Precondiciones	Que existan eventos en Google Calendar
Guión	1. Se hace clic en el botón *registro*. 2. Visualización registro eventos para el día actual.
Excepciones	-
Finalizado por	Usuario.
Postcondiciones	Que los nuevos eventos se reflejen automáticamente.

Capítulo 4

Diseño del sistema

En este capítulo se detallará la forma en que se va a construir la aplicación descrita anteriormente.

4.1. Estructura de sistemas

Como se ha visto en el capítulo de análisis, la aplicación se compone de 2 sistemas, el sensor, que es el encargado de capturar los eventos y el receptor o la visualización de las notificaciones, dónde veremos la información de éstas y podremos acceder al Google Calendar. Ambos utilizarán la API de Google Calendar para la sincronización de lo múltiples dispositivos que pueden utilizarse, cada uno con su localización, etc. La utilización de Google Calendar supone un sistema estándar y de excelente nivel de servicio para la sincronización asegurando la rápida recepción de nuevos eventos y la intregridad de los datos. En la figura 4.1 puede verse el diagrama del sistema. La aplicación estará formada por tres capas, la capa de acceso a datos o de acceso a Google Calenar, la lógica de la aplicación y la capa de presentación.

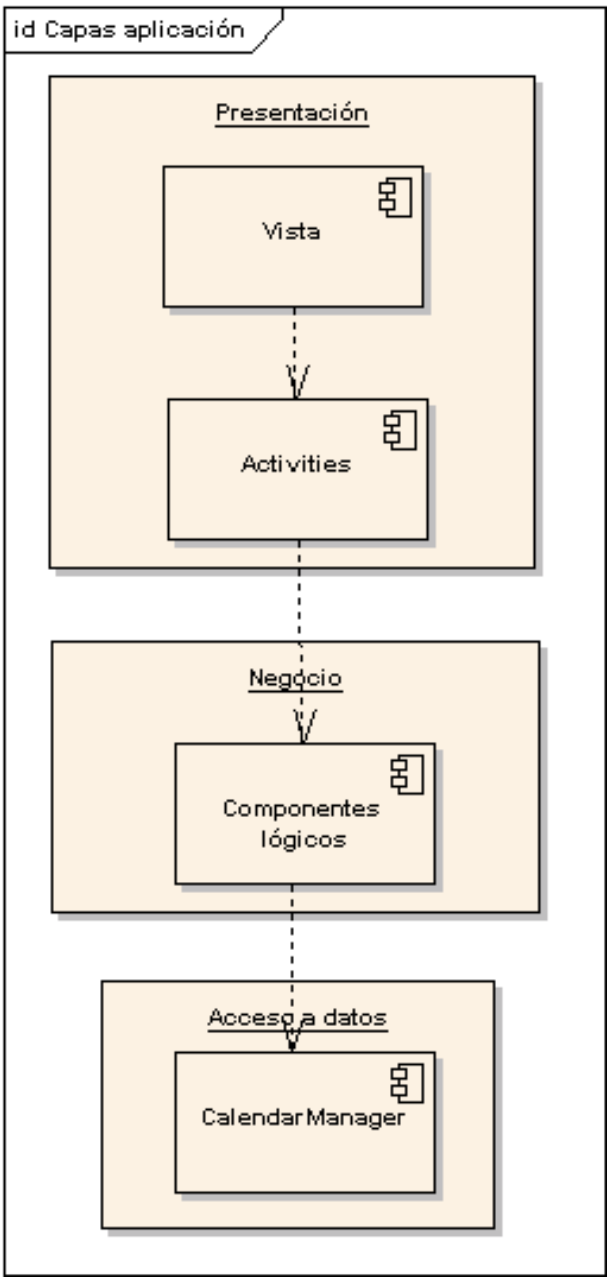

Figura 4.1: Diagrama de arquitectura del sistema.

4.1.1. Patrones arquitectónicos

En la implementación del sistema se basa en el siguiente patrón arquitectónico:

Modelo - Vista - Controlador: la aplicación va a estar dividida en 3 componentes. El modelo contiene la funcionalidad de la aplicación y el acceso a los datos. La vista despliega la información al usuario. En android, se compone de archivos XML (layouts). El controlador maneja las entradas del usuario. Los dos últimos se encuentran dentro de la capa de presentación.

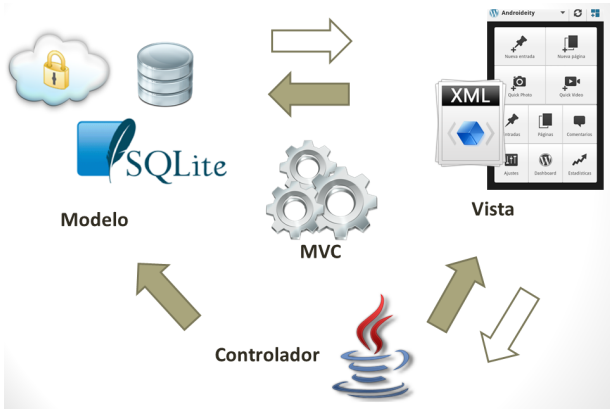

Figura 4.2: Diagrama del patrón MVC en Android.

4.2. Estructura de componentes y conectores.

A continuación podemos ver la imagen del diagrama de despliegue. Nuestra aplicación android consume los servicios de Google Calendar.

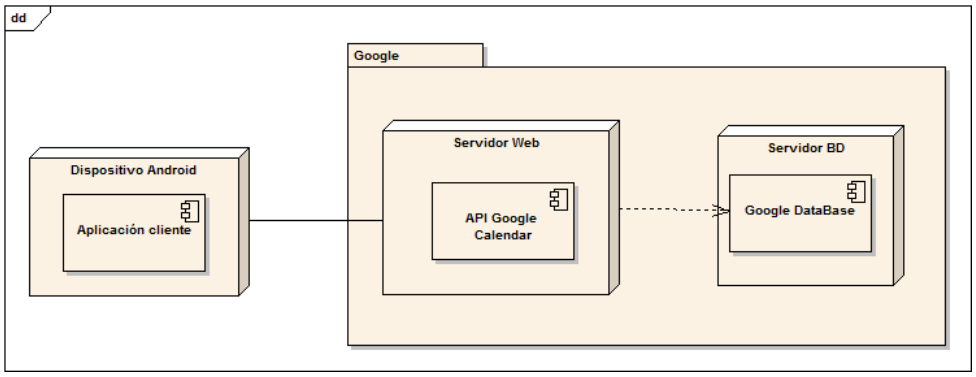

Figura 4.3: Diagrama de despliegue.

4.3. Diseño de clases

Como ya se ha estudiado en capítulos anteriores la aplicación está formada por 2 sistemas principalmente, que se comunican entre ellos mediante la sincronización de Google Calendar. En esta sección se describe la estructura de la aplicación y de cada uno de estos 2 sistemas.

4.3.1. Aplicación

Aquí se muestra el diagrama de paquetes de la aplicación. Además, se muestran las clases "manager"de cada paquete.

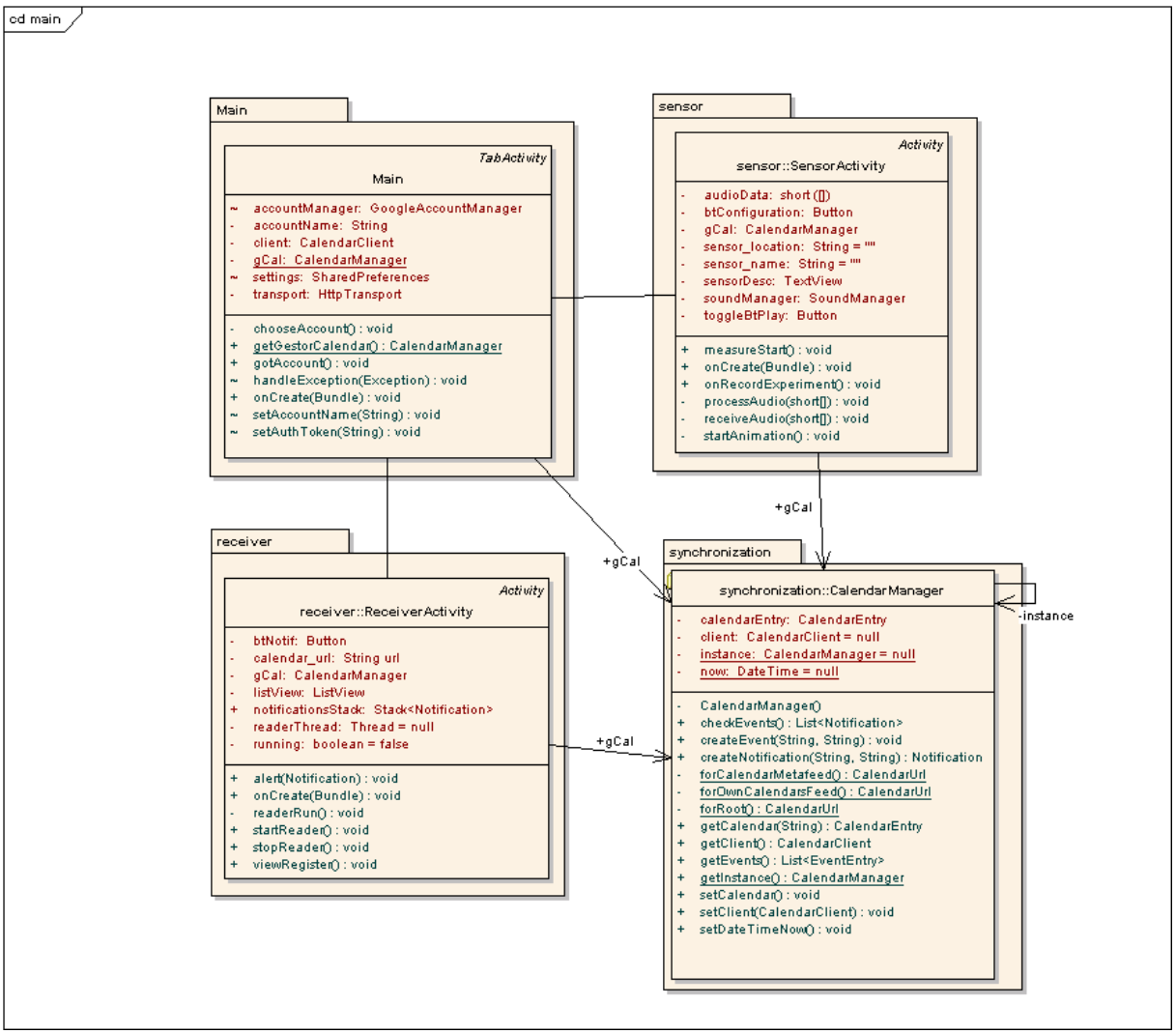

Figura 4.4: Diagrama de paquetes de la aplicación.

4.3.2. Sistema sensor

Este sistema es el encargado de capturar los eventos de sonido y almacenarlos en el Google Calenlar.

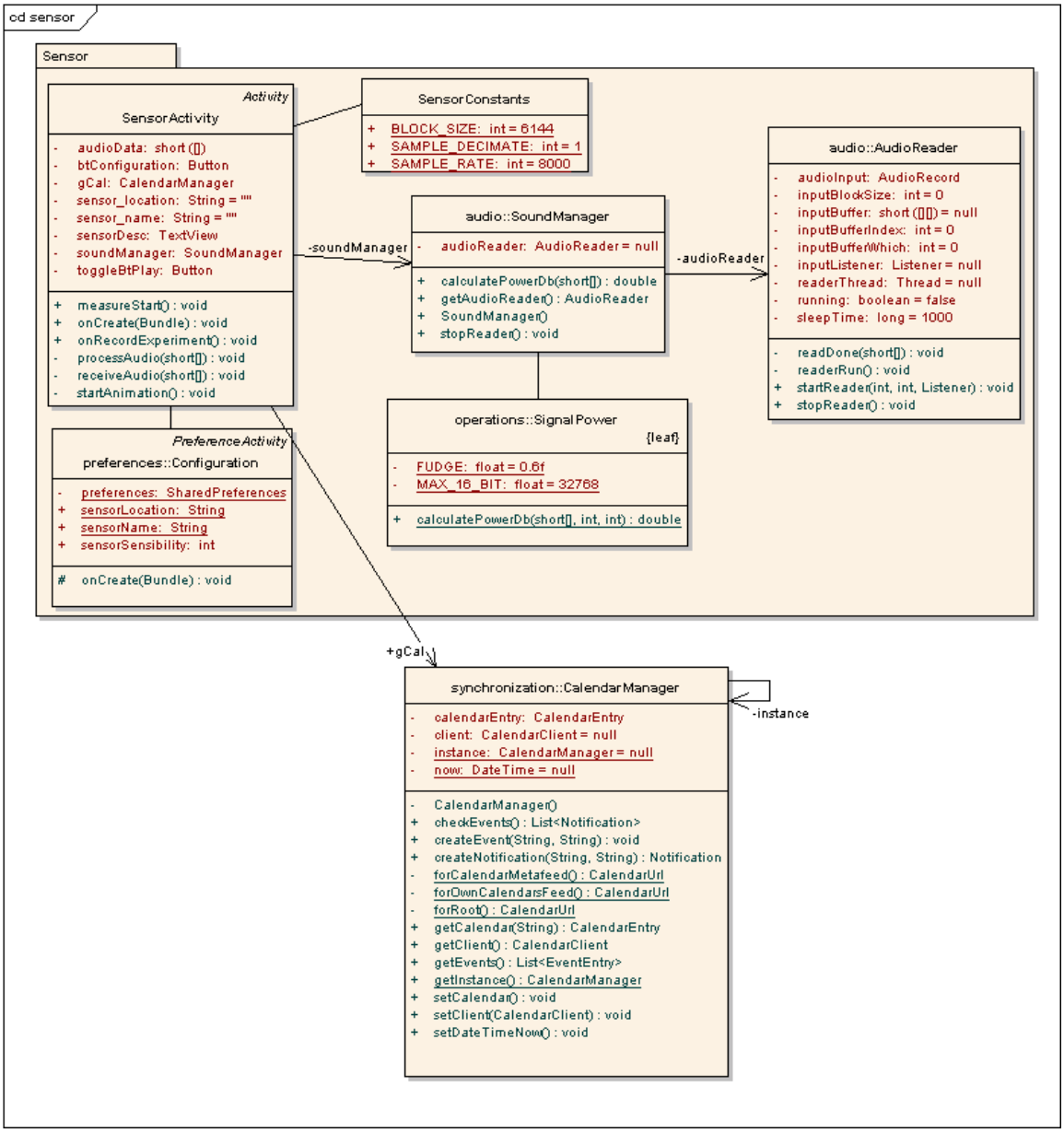

Figura 4.5: Diagrama de clases del sensor.

4.3.3. Sistema de notificaciones

Este sistema es el encargado de notificar al usuario los eventos nuevos registrados en Google Calendar.

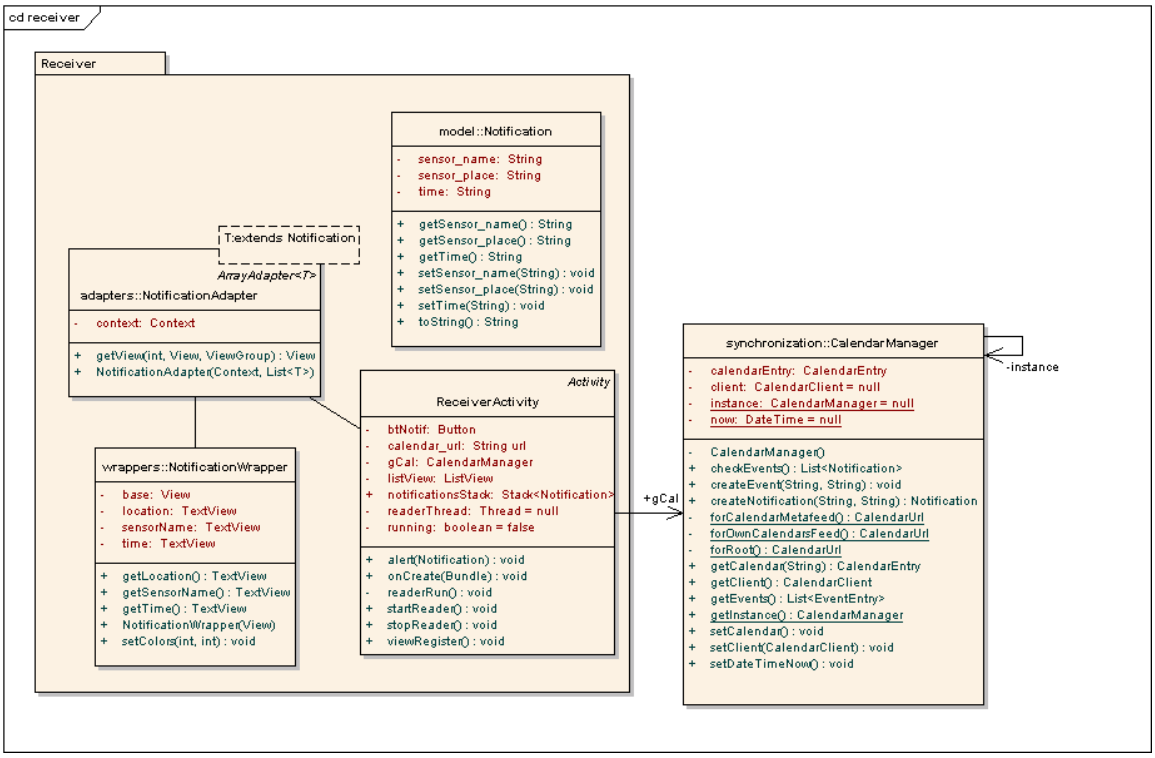

Figura 4.6: Diagrama de clases del receptor de notificaciones.

4.4. Descripción del modelo dinámico

A continuación se describe el comportamiento general del sistema en aspectos dinámicos, especificando su comportamiento a través de diferentes diagramas.

Diagrama de actividades

En la figura 4.7 se muestra un diagrama general que representa todos los flujos de navegación de la aplicación. Al iniciar la aplicación, el usuario tendrá que elegir la cuenta de Gmail a utilizar, si es que posee más de una. Una vez fijada la cuenta deseada, el usuario accede siempre a la pantalla del sensor, en la cual puede activar éste o cambiar su configuración. Una vez activado el sensor podremos observar cómo se genera la animación de onda al detectar ruido y cómo cambia el número de decibelios reflejado en esta misma pantalla dependiendo de la intensidad del ruido ambiental.

La aplicación utiliza un layout de pestañas. Posee 2 pestañas, una la del sensor, la primera, y la segunda, la de las notificaciones. Se trata de una solución de diseño para poder interactuar fácilmente con los 2 modos de funcionamiento. Al hacer click en la

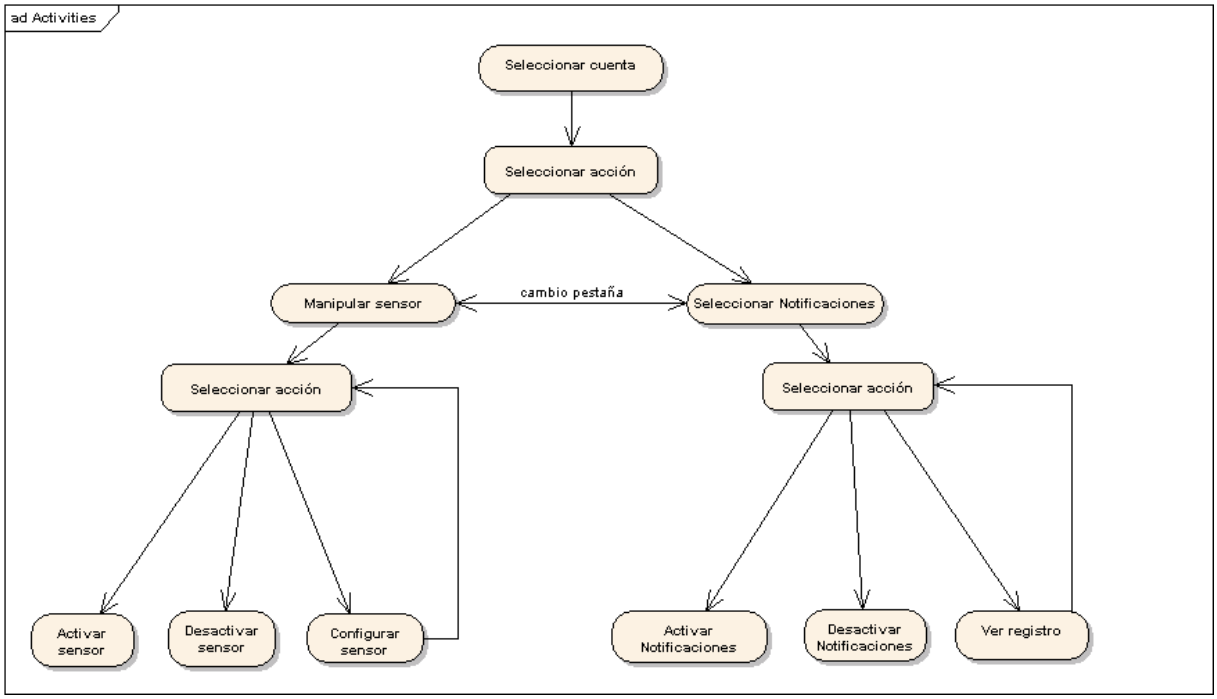

Figura 4.7: Diagrama de actividades.

pestaña contigua se accede al modo receptor. Aquí podremos activar el las notificaciones o ver el registro de eventos.

4.4.1. Diagramas de secuencia

En este apartado aparecen los diagramas de secuencia asociados a los casos de uso descritos en la sección del análisis. Éstos nos ayudarán a comprender mejor el funcionamiento programático de cada caso de uso.

Escenario 0. Configuración de la aplicación

En la fig. 4.8 se muestra el diagrama de secuencia para el escenario 0 Configuración de la aplicación.

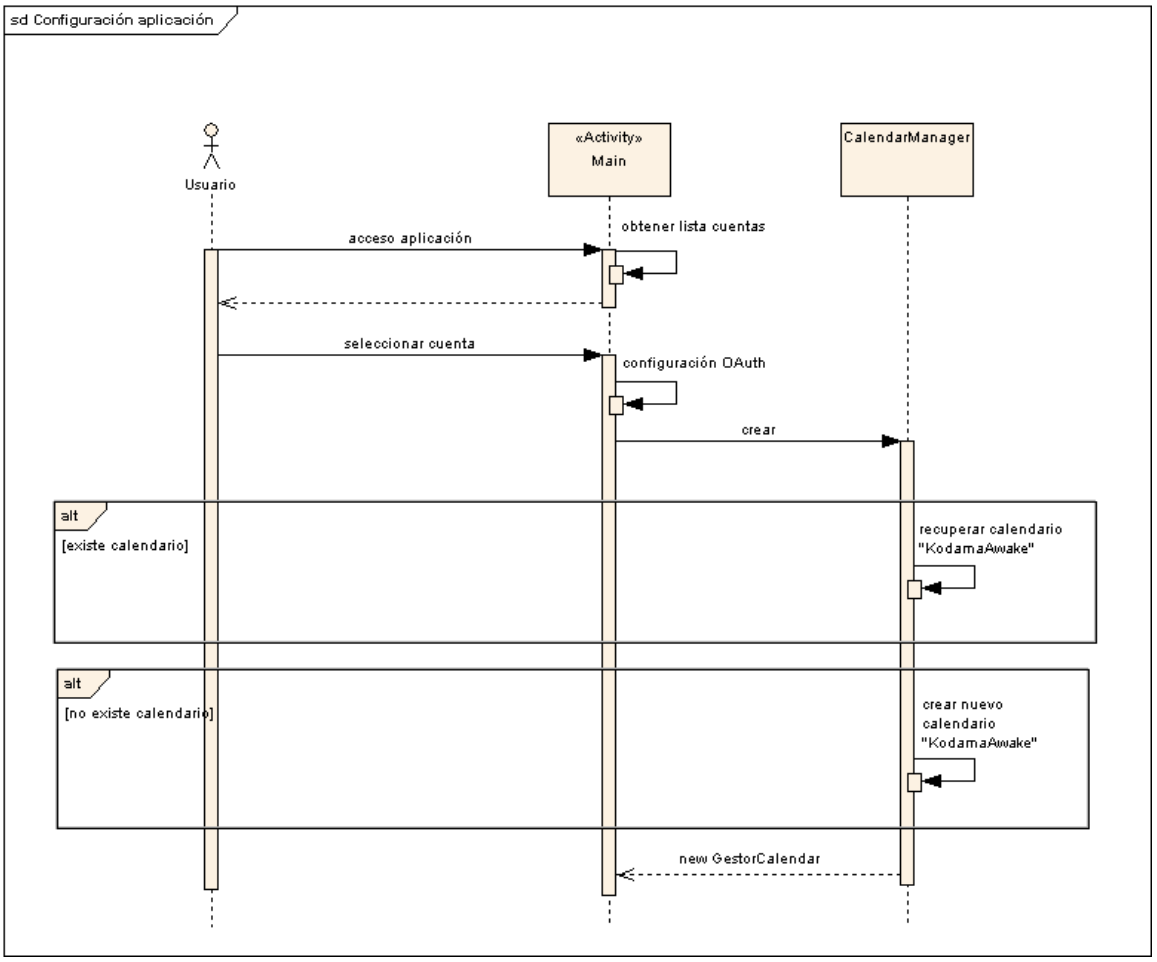

Figura 4.8: Diagrama de secuencia de configuración de la aplicación.

Escenario 1.1. Activar sensor

En la fig. 4.9 se muestra el diagrama de secuencia para el escenario 1.1 Activar sensor.

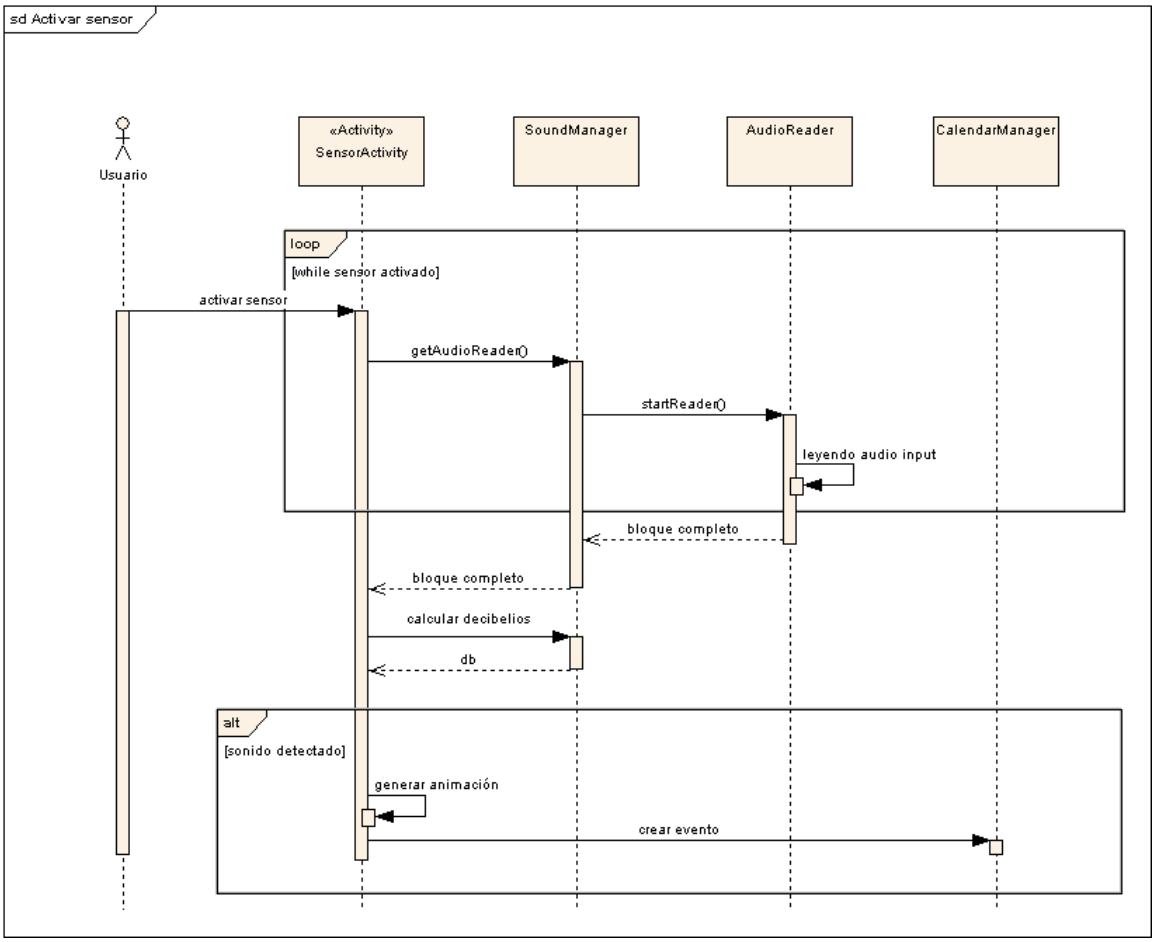

Figura 4.9: Diagrama de secuencia Activar Sensor.

Escenario 1.2 Desactivar sensor

En la fig. 4.10 se muestra el diagrama de secuencia para el escenario 1.2 Desactivar sensor.

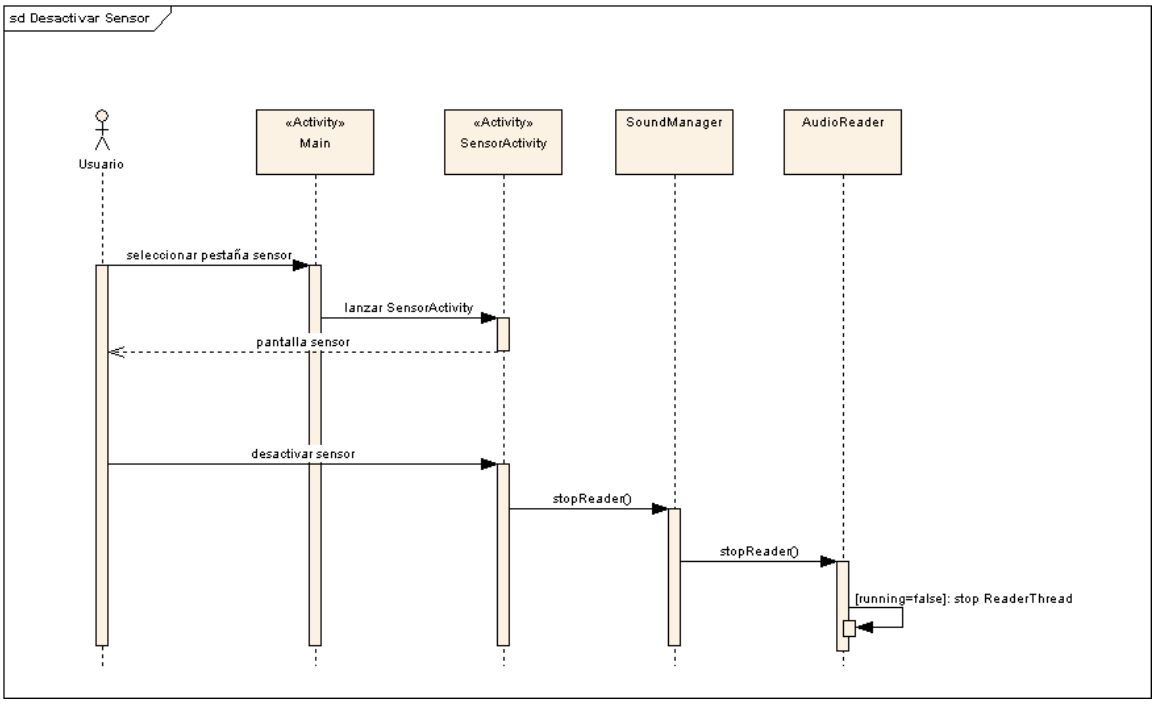

Figura 4.10: Diagrama de secuencia Desactivar Sensor.

Escenario 1.3. Configurar sensor

En la fig. 4.11 se muestra el diagrama de secuencia para el cambio de la configuración del nombre del sensor el escenario 1.3 Configurar sensor.

También se muestra en la figura 4.12 cómo se carga la configuración establecida previamente en la aplicación para su uso.

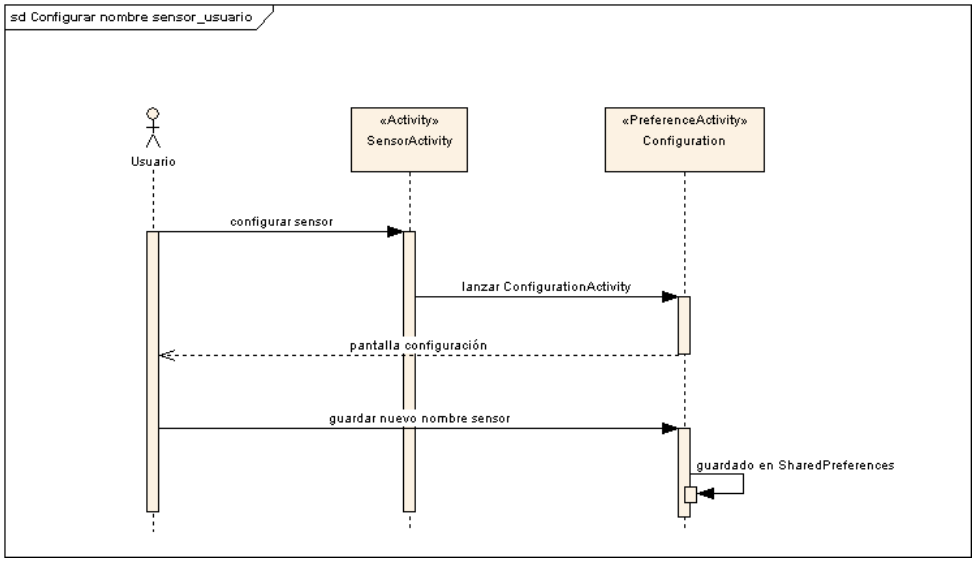

Figura 4.11: Diagrama de secuencia Configurar nombre del Sensor.

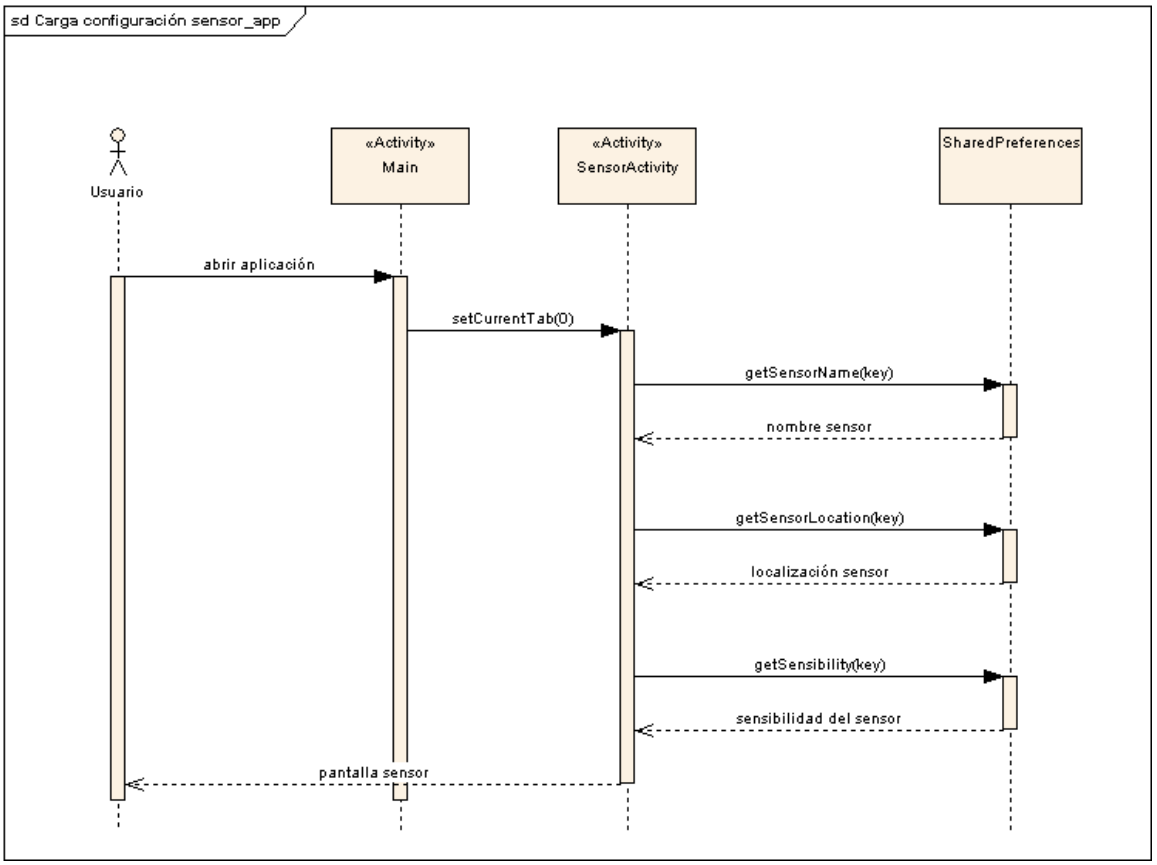

Figura 4.12: Diagrama de secuencia Configurar Sensor.

Escenario 2.1. Activar notificaciones

En la fig. 4.13 se muestra el diagrama de secuencia para el escenario 2.1 Activar notificaciones.

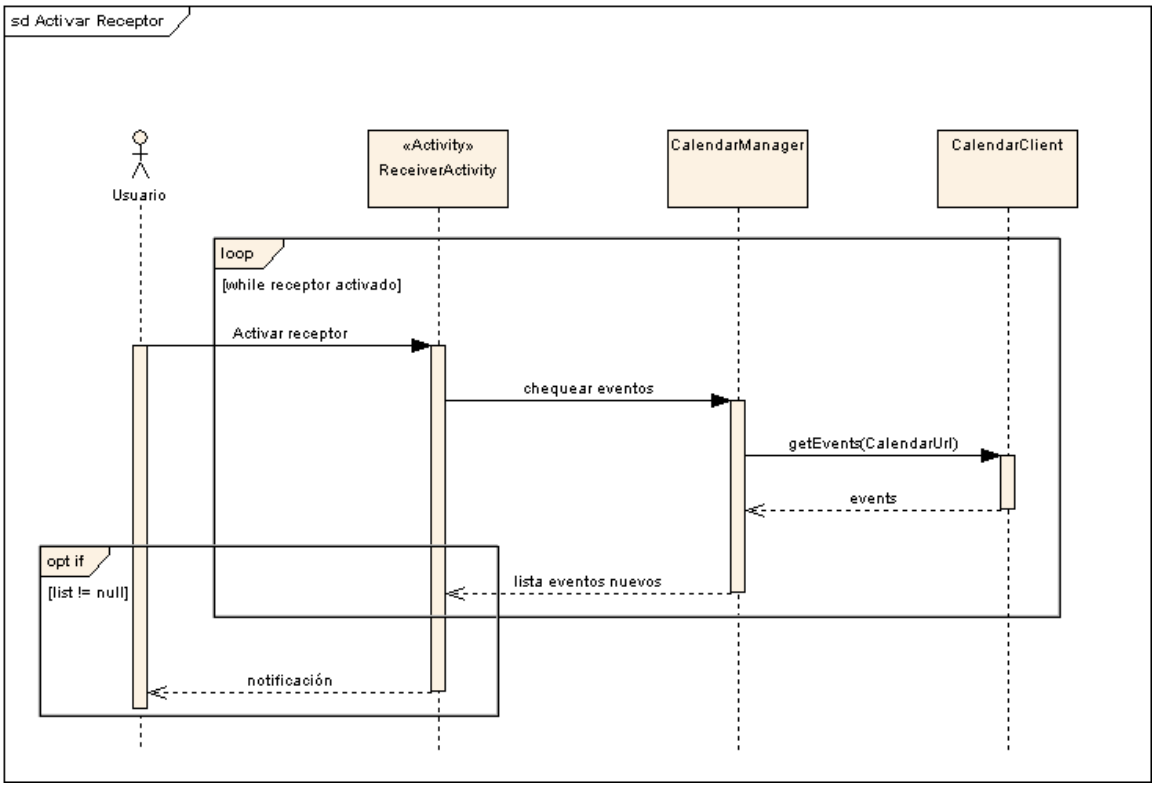

Figura 4.13: Diagrama de secuencia Activar Notificaciones.

Escenario 2.2. Desactivar notificaciones

En la fig. 4.7 se muestra el diagrama de secuencia para el escenario 2.2 Desactivar notificaciones.

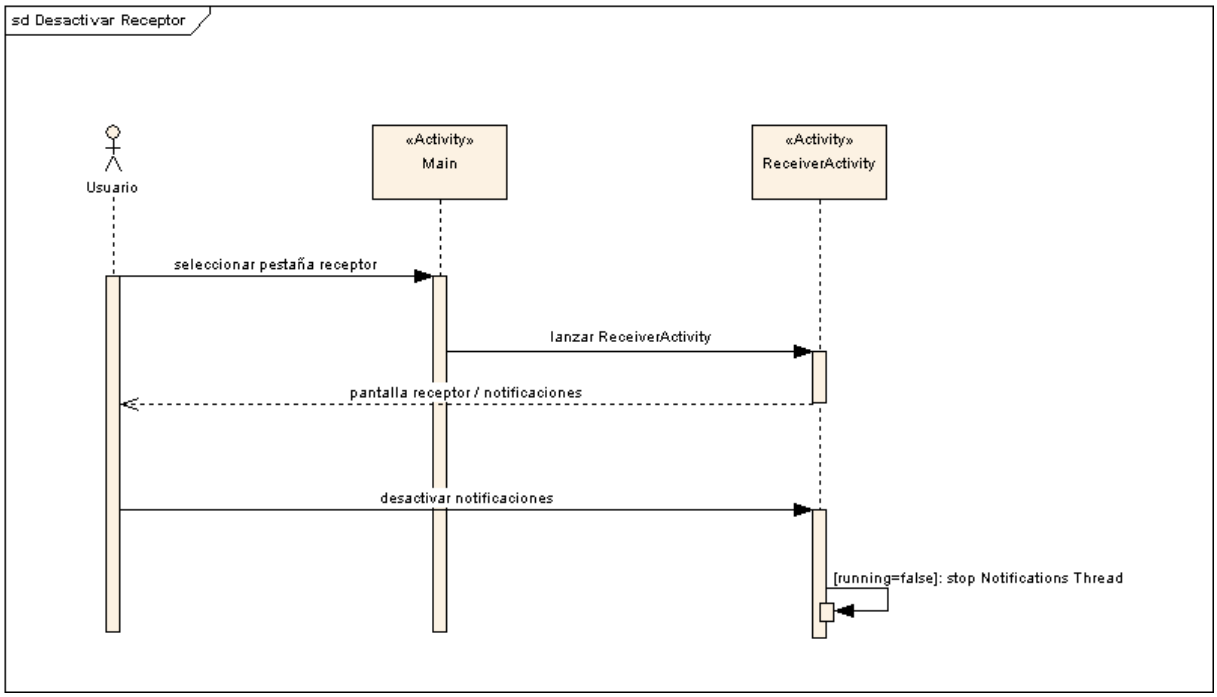

Figura 4.14: Diagrama de secuencia Desactivar Notificaciones.

Escenario 2.3. Visualizar notificación

Este se engloba dentro del caso de uso activar notificaciones, ya que se muestra la notificación tras haber activado las notificaciones cuando se detecta una nueva en el Google Calendar. Ver figura 4.9.

Escenario 2.4. Ver registro de notificaciones

En la fig. 4.15 se muestra el diagrama de secuencia para el escenario 2.4 Ver registro de notificaciones.

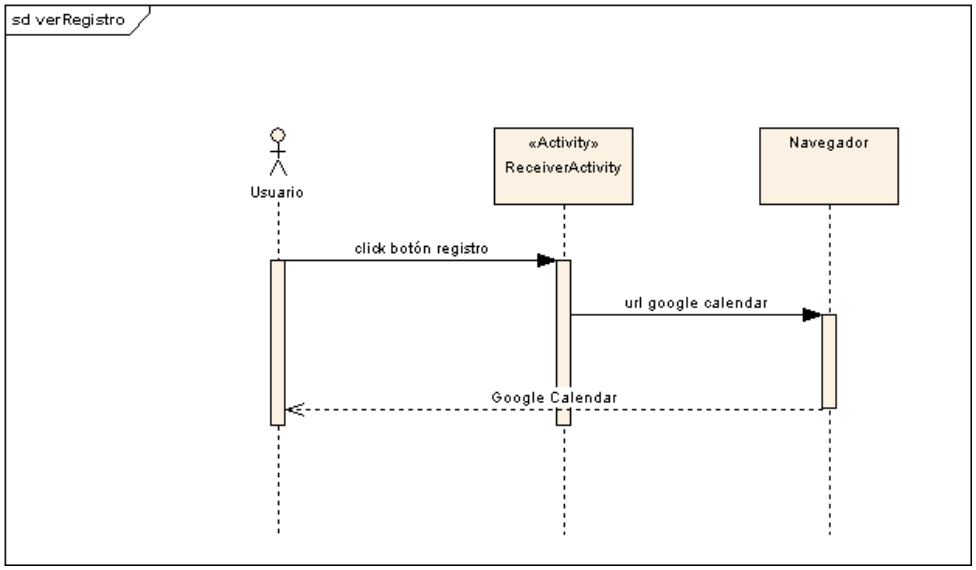

Figura 4.15: Diagrama de secuencia Ver registro de notificaciones.

4.5. Patrones de diseño

A continuación se explicarán los patrones de diseño a utilizar en el sistema.

4.5.1. Patrón Adapter

El patrón Adapter (Adaptador) se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda.

Se utiliza este patrón para mostrar un objeto Notificación como un ítem de la lista de la pantalla de Notificaciones.

4.5.2. Patrón Singleton

Utilizamos este patrón en la clase CalendarManager.

4.6. Layouts

A continuación podremos ver el diseño de layouts que se establece para esta aplicación. Cabe notar que se siguen los estándares de Android para asegurar un buen rendimiento.

4.6.1. Main layout

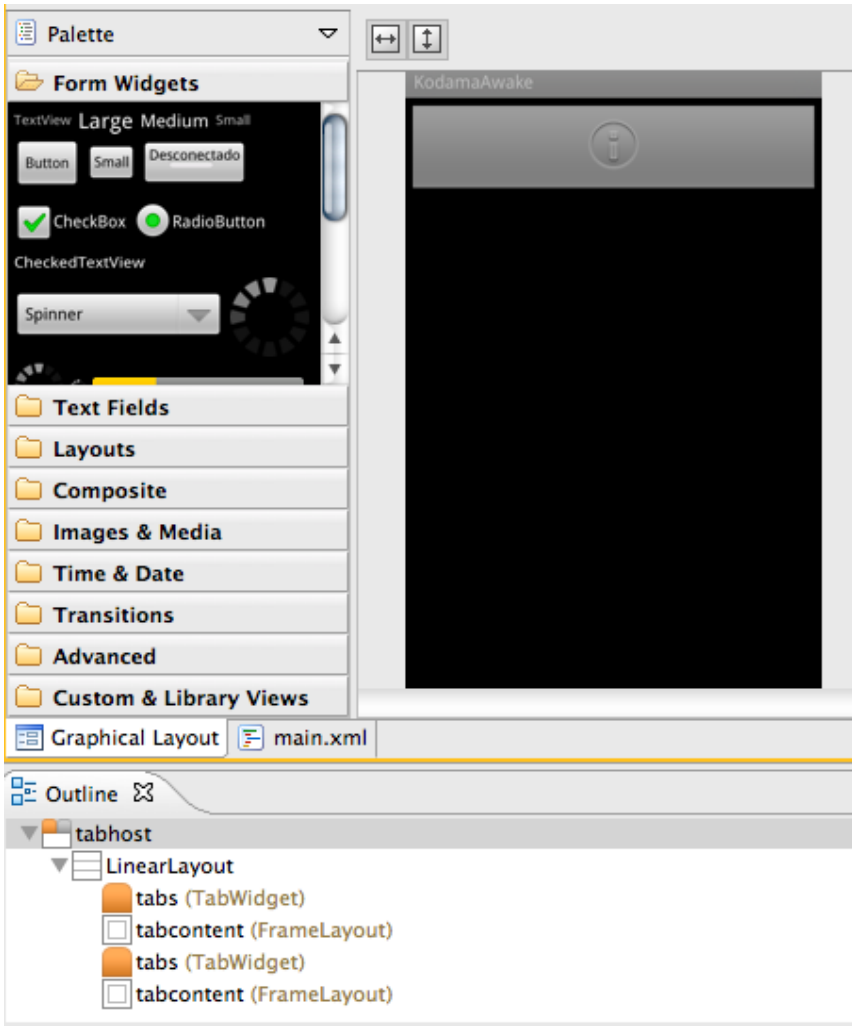

Figura 4.16: Main layout.

Listing 4.1: layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical"
        android:padding="5dp" >

        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
```

```
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
    </LinearLayout>

</TabHost>
```

Sensor layout

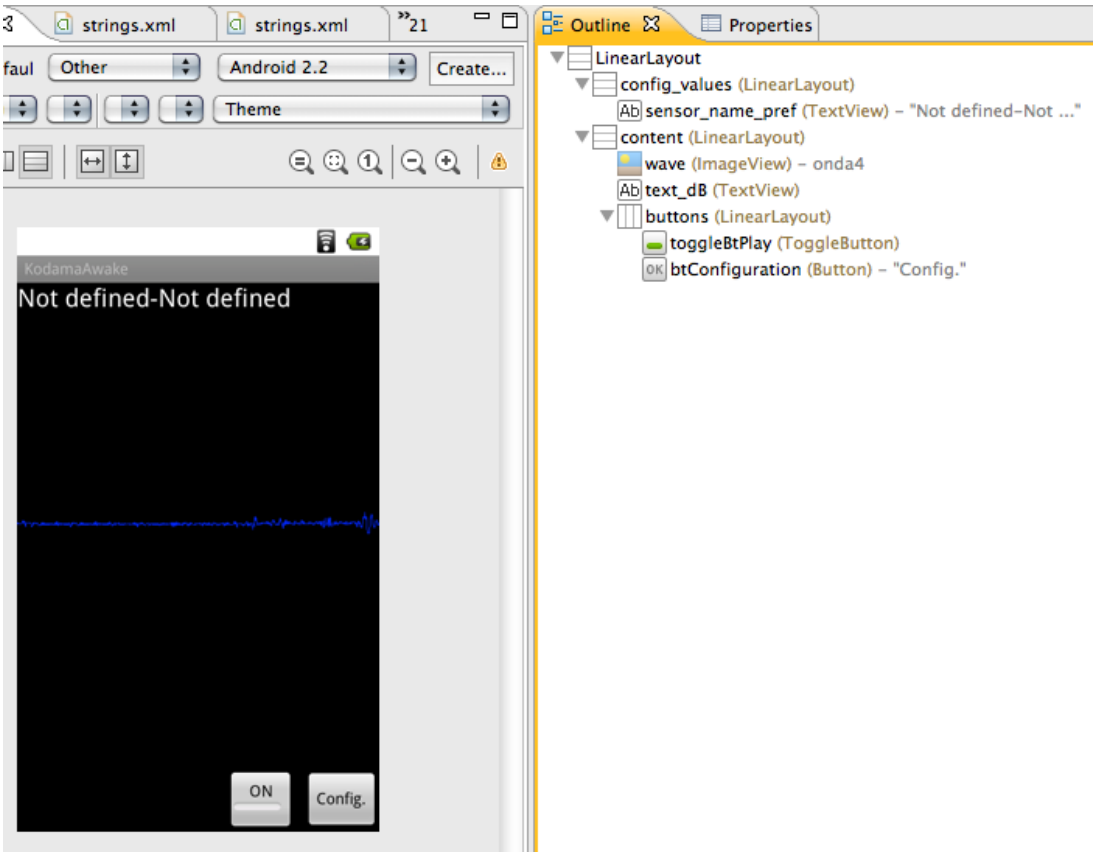

Figura 4.17: Sensor layout.

Listing 4.2: layout/recording.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="right"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/config_values"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
```

```
            android:id="@+id/sensor_name_pref"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/not_defined_not_defined"
            android:textAppearance="?android:attr/textAppearanceLarge" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="0.76"
        android:gravity="right"
        android:orientation="vertical" >

        <ImageView
            android:id="@+id/wave"
            android:layout_width="wrap_content"
            android:layout_height="0dip"
            android:layout_weight="0.46"
            android:src="@drawable/onda4"
            android:contentDescription="@string/wave"/>

        <TextView
            android:id="@+id/text_dB"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dip"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <LinearLayout
            android:id="@+id/buttons"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="right" >

            <ToggleButton
                android:id="@+id/toggleBtPlay"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="center_vertical"
                android:gravity="center"
                android:textOff="ON"
                android:textOn="ON" />

            <Button
                android:id="@+id/btConfiguration"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="right"
                android:layout_marginLeft="10dp"
                android:text="@string/configuration" />
        </LinearLayout>

    </LinearLayout>

</LinearLayout>
```

Notifications layout

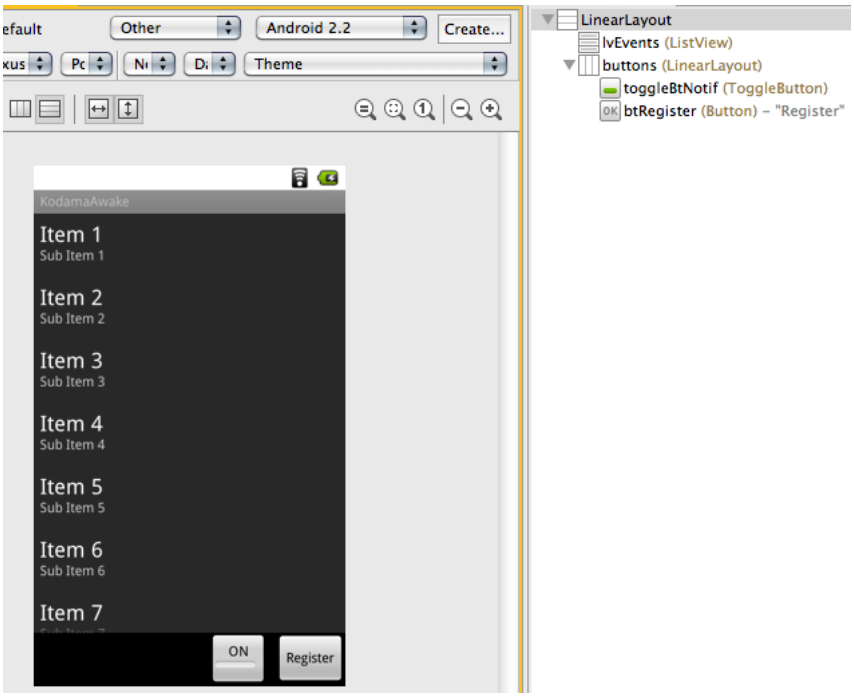

Figura 4.18: Notifications layout.

Listing 4.3: layout/notifications.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:keepScreenOn="true">

    <ListView
        android:id="@+id/lvEvents"
        android:layout_width="fill_parent"
        android:layout_height="364dp"
        android:layout_weight="2.43" >
    </ListView>

    <LinearLayout
        android:id="@+id/buttons"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right" >

        <ToggleButton
            android:id="@+id/toggleBtNotif"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:textOff="ON"
            android:textOn="ON" />

        <Button
            android:id="@+id/btRegister"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="right"
            android:layout_marginLeft="10dp"
            android:text="@string/register" />
    </LinearLayout>

</LinearLayout>
```

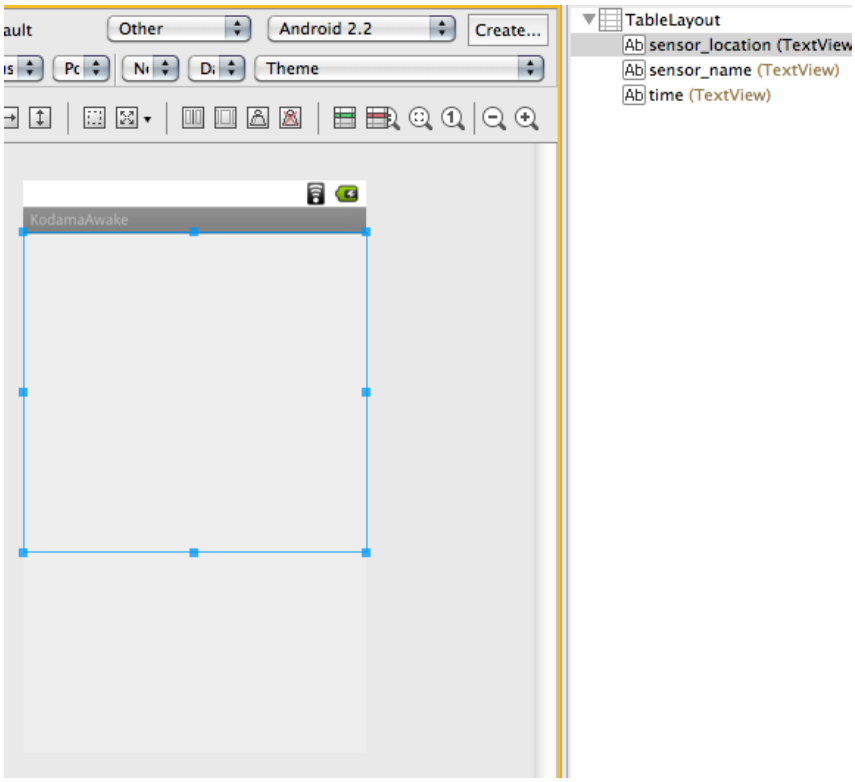

Figura 4.19: Notification item layout.

Listing 4.4: layout/notificationRow.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#EEEEEE"
    android:stretchColumns="*" >

    <TextView
        android:id="@+id/sensor_location"
        android:layout_width="0dip"
        android:layout_weight="2"
        android:gravity="center"
        android:textColor="#000000"
        android:textSize="18dip"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/sensor_name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#000000"
        android:textSize="18dip"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/time"
        android:layout_width="0dip"
        android:layout_margin="1dip"
        android:layout_weight="1"
        android:gravity="center"
        android:textColor="#000000"
        android:textSize="18dip" />

</TableLayout>
```

Configuration layout

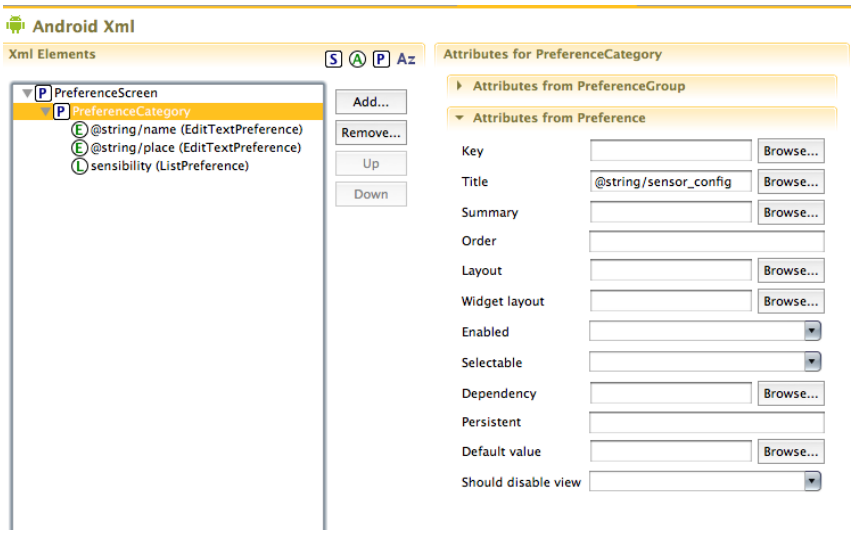

Figura 4.20: Configuration layout.

Listing 4.5: xml/preferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory android:title="@string/sensor_config" >
        <EditTextPreference
            android:name="@string/name"
            android:defaultValue="@string/not_defined"
            android:key="sensor_name"
            android:summary="@string/sensor_name_desc"
            android:title="@string/sensor_name" />
        <EditTextPreference
            android:name="@string/place"
            android:defaultValue="@string/not_defined"
            android:key="sensor_location"
            android:summary="@string/sensor_location_desc"
            android:title="@string/sensor_location" />
            <ListPreference android:summary="@string/sensibility_desc" android:title="@string/
                sensibility" android:dialogTitle="@string/sensibility" android:key="sensibility"
                android:entryValues="@array/sensibility_array_values" android:entries="@array/
                sensibility_array"/>
    </PreferenceCategory>

</PreferenceScreen>
```

4.7. Validación de requisitos

R.1 Requisitos funcionales

R.1.1 Requisitos funcionales generales

R.1.1.1 *Los eventos se capturarán desde un dispositivo móvil:* el dispositivo móvil debe contar con sistema operativo android con versión 2.2 o superior.

- **Cumplido en 4.3.1, 4.3.2, 4.3.3**

R.1.2 *Sincronización de dispositivos:* los dispositivos han de poder sincronizarse para la notificación de eventos independiente del lugar. La mejor opción es usar Google Calendar, ya que es un sistema de Google, un sistema estándar.

- **Cumplido en 4.3.2, 4.3.3**

R.1.3 *Los dos modos (sensor y receptor) accesibles desde una misma pantalla:* un usuario podrá poner en funcionamiento los 2 modos desde una misma pantalla.

- **Cumplido en 4.3.1**

R.2 Requisitos del sensor

R.2.1 *El usuario podrá configurar el sensor:* un usuario podrá establecer un nombre y una localización para el dispositivo. Así como modificar la sensibilidad.

- **Cumplido en 4.3.2**

R.2.2 *Existirá información visual de detección de sonido:* cuando se detecte un sonido se visualizará una animación en la pantalla a fin de indicar que el sensor está funcionando.

- **Cumplido en 4.3.2**

R.3 Requisitos del receptor

R.3.1 *La notificación se mostrará en un primer plano:* cuando llegue una nueva notificación, será totalmente visible en la pantalla.

- **Cumplido en 4.3.3**

R.3.2 *Debe notificarse mediante ruido, vibración y efectos luminosos.*

- **Cumplido en 4.3.3**

R.3.3 *Debe mostrar información relevante:*los datos del sensor y el instante en que sucedió.

- **Cumplido en 4.3.3**

R.3.4 *Visualización de registro de eventos:*se debe poder visualizar un registro de los eventos.

- **Cumplido en 4.3.3**

Capítulo 5

Modelo de datos

En este capítulo se explica como se manejan los datos en la aplicación.

En esta aplicación utilizamos la API de Google Calendar como un Servicio Web, no se utiliza otro tipo de sistema de bases de datos. Los datos de los eventos capturados por el sensor se almacenan en el Google Calendar, y, desde la pantalla de Notificaciones podemos acceder a éste a través del botón Registro.

5.1. Plataforma de Google

Al arrancar la aplicación se crea un nuevo calendario, si no existe, llamado KodamaAwake.

Cuando un evento es capturado se almacena la información del sensor, la fecha y hora de captura en este calendario. Se utiliza para ello el atributo *title* del evento y el atributo *when*.

La información se almacena en el título del evento siguiendo esta estructura: *Localización sensor;Nombre sensor*. Para recoger cada parte se trocea la cadena mediante un *split* que la rompe por los *;*.

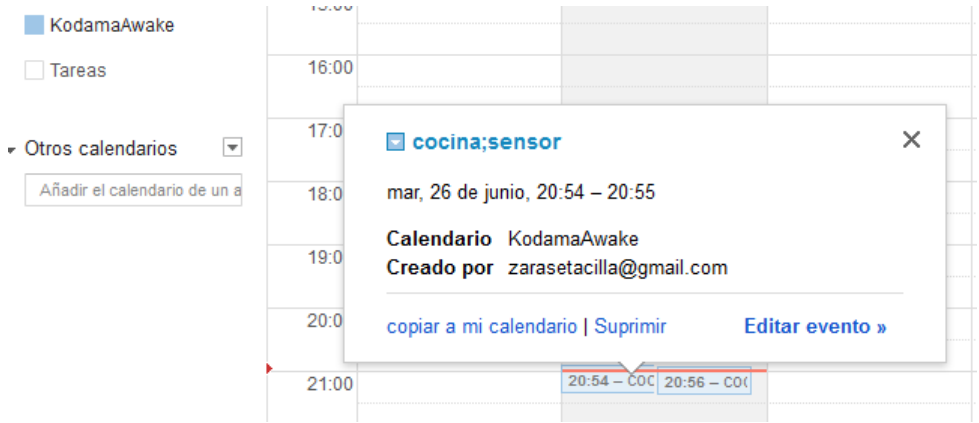

Figura 5.1: Google Calendar screenshot

Por otro lado los datos de tiempo se almacenan en el atributo *When* del objeto *EventEntry* que nos proporciona la API.

La clase encargada de comunicarse con Google Calendar es CalendarManager.java. En está tenemos los siguientes métodos:

- Establecer calendario: crea un nuevo calendario y lo establece para su uso o bien, comprueba que ya existe y directamente lo establece para su uso.
- Crear evento: Crea un nuevo evento con los datos correspondientes
- Chequear eventos: retorna una lista de notificaciones. Ésta se genera a partir de la información de los eventos capturados en el Google Calendar desde un momento dado. Este momento de tiempo se define por la hora en que se activaron las notificaciones o bien, por la hora del anterior chequeo para comprobar si había nuevos eventos.

El sensor crea un evento al ser capturado. El receptor tendrá que estar llamando de manera continua a *Chequear eventos* para comprobar si hay eventos nuevos. En condiciones normales siempre tendríamos un único evento en la lista, el último. Sin embargo, podría darse un caso en el que tengamos 2 si hay algún problema de comunicación. Asi, de esta manera, nunca perderemos datos. Por otro lado, la API no nos ofrece la opción de *obtener último evento introducido*.

La API nos devuelve los eventos ordenados, del más actual al más antiguo. Nos devuelve los 25 últimos eventos y de ahí extraemos los que nos interesan. Hemos observado que esto no repercute en el rendimiento. Si así fuera, habría que explorar otra solución para ampliar las posibilidades que nos ofrece la API.

La API también nos ofrece las opciones de borrar o actualizar evento, sin embargo no se necesitan en este caso.

5.2. Visualización de los eventos

Los eventos registrados se almacenan en el Google Calendar. Se puede acceder a ellos desde la pantalla de notificaciones haciendo click en el botón de Registro. Al hacer click, se abrirá un ventana de explorador que nos lleva a la versión de Google Calendar para móviles. Tendremos que loguearnos la primera vez. En la pantalla se mostrarán entonces un listado de eventos para el día actual.

5.3. API de Google Calendar

Las librerías utilizadas son las siguientes:

- Google API Java client library: google-api-client-1.6.0-beta.jar
- google-api-client-extensions-android2-1.6.0-beta.jar
- google-api-services-calendar-v3-1.3.1-beta.jar

Figura 5.2: Eventos en Google Calendar

- google-oauth-client-1.6.0-beta.jar

Utilizamos Http Request Factory de la librería *google-api-client* para inicializar todas las peticiones y reintentar las peticiones fallidas.

Tenemos que usar AccountManager (google-api-client-extensions-android2) para obtener las cuentas. y para pedir los tokens de autorización necesitamos añadir los siguientes permisos en el fichero Android application manifest:

android.permission.GET_ACCOUNTS

android.permission.USE_CREDENTIALS

Capítulo 6

Pruebas de software

6.1. Pruebas unitarias

Estas pruebas fueron útiles para detectar errores en la codificación y un error de mala configuración de un método de la API de Google calendar.

Para realizar las pruebas unitarias con Android, hubo que hacer extender las clases que necesitaban del calendario de Google de ActivityInstrumentationTestCase2 y fijar la configuración para las pruebas en el fichero AndroidManifest.xml.

A continuación se muestra el código de las pruebas unitarias.

SoundsTest.java

Paquete kodama.testing 6.1: SoundsTest.java

```
package kodama.testing;

import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.UnsupportedAudioFileException;

import kodama.sensor.audio.SoundManager;

/**
 *
 * @author saragarciaperez
 *
 *         Testing sound input
 */
public class SoundsTest {
        private SoundManager soundManager;
        private ByteArrayOutputStream out = new ByteArrayOutputStream();
        private AudioInputStream in;

        public void setTestEnvironment() {
                soundManager = new SoundManager();
        }

        // Testing different dB input
        public void testSounds(String file) throws FileNotFoundException,
                        UnsupportedAudioFileException, IOException {

                in = AudioSystem.getAudioInputStream(new FileInputStream(file));

                int read;
                byte[] buff = new byte[1024];
                while ((read = in.read(buff)) > 0) {
                        out.write(buff, 0, read);
                }
                out.flush();
```

```
                byte[] audioBytes = out.toByteArray();

                double decibelios = soundManager.calculatePowerDb(audioBytes);

                System.out.println("Decibelios: " + decibelios);
                out.reset();
        }

        // The sound files were extracted from www.audiotest.net
        public static void main(String[] args) {

                SoundsTest soundsTest = new SoundsTest();
                soundsTest.setTestEnvironment();
                try {
                        soundsTest.testSounds("src/test/36db.wav");
                        soundsTest.testSounds("src/test/42db.wav");
                        soundsTest.testSounds("src/test/48db.wav");
                        soundsTest.testSounds("src/test/54db.wav");
                        soundsTest.testSounds("src/test/60db.wav");
                        soundsTest.testSounds("src/test/66db.wav");
                        soundsTest.testSounds("src/test/72db.wav");
                        soundsTest.testSounds("src/test/78db.wav");

                        // (-54.993771537882466,-57.627165366241364)

                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (UnsupportedAudioFileException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

CalendarAPITest.java

Paquete kodama.testing 6.2: CalendarAPITest.java

```
package kodama.testing;

import java.io.IOException;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.TimeZone;

import kodama.main.Main;
import kodama.synchronization.CalendarManager;
import kodama.synchronization.calendar.CalendarUrl;
import kodama.synchronization.calendar.model.CalendarEntry;
import kodama.synchronization.calendar.model.EventEntry;
import kodama.synchronization.calendar.model.When;

import org.junit.Test;

import android.test.ActivityInstrumentationTestCase2;

import com.android.ddmlib.Log;
import com.google.api.client.util.DateTime;

/**
 *
 * @author saragarciaperez
 *         <p>
 *          Unit Test for Calendar API
 *         </p>
 */
public class CalendarAPITest extends ActivityInstrumentationTestCase2<Main> {

        public CalendarAPITest() throws ClassNotFoundException {
                super("kodama.main.Main", Main.class);
        }

        DateTime date = null;
        private When when = null;
        private CalendarEntry calendarEntry = null;
        private static CalendarManager gCal = null;

        protected void setUp() {
                getActivity();

                // It's neccesary wait here when debug, because we need time to set the
                // calendar and to start to work with it.
                // Then we have to select the account of gmail and one the app is
                // loaded, continue (Step over)
                gCal = CalendarManager.getInstance();

                try {
```

```
                super.setUp();

        } catch (Exception e) {
                e.printStackTrace();
        }

        // Calendar is set in setAccount (Main)
        assertEquals("KodamaAwake", calendarEntry.title);

        //the text used is extracted from Don Quijote (Cervantes) and Campos de Castilla (
            Antonio Machado)
        testLenthName();
        testLenthDescription();
        testLenthDescriptionLonger();
}

// Testing the lenth of the attribute title
@Test
public void testLenthName() {

        StringBuilder cad = new StringBuilder();

        cad.append("Dichosa edad y siglos dichosos aquéllos a quien los antiguos pusieron");
        cad.append(" nombre de dorados, y no porque en ellos el oro, que en esta nuestra edad")
            ;
        cad.append(" de hierro tanto se estima, se alcanzase en aquella venturosa sin fatiga");
        cad.append(" alguna, sino porque entonces los que en ella vivían ignoraban estas");
        cad.append(" dos palabras de tuyo y mío. Eran en aquella santa edad todas las cosas");
        cad.append(" comunes. A nadie le era necesario para alcanzar su ordinario sustento");
        cad.append(" tomar otro trabajo que alzar la mano y alcanzarle de las robustas encinas,
            ");
        cad.append(" que liberalmente les estaban convidando con su dulce y sazonado fruto.");
        cad.append(" Las claras fuentes y corrientes ríos, en magnífica abundancia,");
        cad.append(" sabrosas y transparentes aguas les ofrecían. En las quiebras");
        cad.append(" de las peñas y en lo hueco de los árboles formaban su república las");
        cad.append(" solícitas y discretas abejas, ofreciendo a cualquiera mano,");
        cad.append(" sin interés alguno, la fértil cosecha de su dulcísimo trabajo.");
        cad.append(" Los valientes alcornoques despedían de sí, sin otro artificio que");
        cad.append(" el de su cortesía, sus anchas y livianas cortezas, con que se");
        cad.append(" comenzaron a cubrir las casas, sobre rústicas estacas sustentadas,");
        cad.append(" no más que para la defensa de las inclemencias del cielo.");
        cad.append(" Todo era paz entonces, todo amistad, todo concordia: aún no se había");
        cad.append(" atrevido la pesada reja del corvo arado a abrir ni visitar las entrañas");
        cad.append(" piadosas de nuestra primera madre; que ella, sin ser forzada, ofrecía,");
        cad.append(" por todas las partes de su fértil y espacioso seno, lo que pudiese");
        cad.append(" hartar, sustentar y deleitar a los hijos que entonces la poseían.");
        cad.append(" Entonces sí que andaban las simples y hermosas zagalejas de valle");
        cad.append(" en valle y de otero en otero, en trenza y en cabello, sin más vestidos");
        cad.append(" de aquéllos que eran menester para cubrir honestamente lo que la ");
        cad.append(" honestidad quiere y ha querido siempre que se cubra, y no eran sus");
        cad.append(" adornos de los que ahora se usan, a quien la púrpura de Tiro y la");
        cad.append(" por tantos modos martirizada seda encarecen, sino de algunas hojas");
        cad.append(" verdes de lampazos y yedra, entretejidas, con lo que quizá iban tan");
        cad.append(" pomposas y compuestas como van agora nuestras cortesanas con las raras");
        cad.append(" y peregrinas invenciones que la curiosidad ociosa les ha mostrado.");

        Log.i("Lenth string prueba", String.valueOf(cad.length()));

        crearEvento(cad);

        EventEntry event = chequearEventos().get(0);

        String nameRead = event.title;

        Log.i("Max. Length", String.valueOf(nameRead.length()));
}

// Testing the lenth of the attribute summary or description
@Test
public void testLenthDescription() {

        StringBuilder cad = new StringBuilder();

        cad.append("He vuelto a ver los álamos dorados,\n");
        cad.append("álamos del camino en la ribera\n");
        cad.append("del Duero, entre San Polo y San Saturio,\n");
        cad.append("tras las murallas viejas\n");
        cad.append("de Soria barbacana\n");
        cad.append("hacia Aragón, en castellana tierra.\n");
        cad.append("Estos chopos del río, que acompañan\n");
        cad.append("con el sonido de sus hojas secas\n");
        cad.append("el son del agua, cuando el viento sopla,\n");
        cad.append("tienen en sus cortezas\n");
        cad.append("grabadas iniciales que son nombres\n");
        cad.append("de enamorados, cifras que son fechas.\n");
        cad.append("¡Álamos del amor que ayer tuvisteis\n");
        cad.append("de ruiseñores vuestras ramas llenas;\n");
        cad.append("álamos que seréis mañana liras\n");
        cad.append("del viento perfumado en primavera;\n");
        cad.append("álamos del amor cerca del agua\n");
        cad.append("que corre y pasa y sueña,\n");
        cad.append("álamos de las márgenes del Duero,\n");
        cad.append("conmigo vais, mi corazón os lleva!\n");
```

```
        cad.append("¡Oh!, sí, conmigo vais, campos de Soria,\n");
        cad.append("tardes tranquilas, montes de violeta,\n");
        cad.append("alamedas del río, verde sueño\n");
        cad.append("del suelo gris y de la parda tierra,\n");
        cad.append("agria melancolía\n");
        cad.append("de la ciudad decrépita,\n");
        cad.append("me habéis llegado al alma,\n");
        cad.append("¿o acaso estabais en el fondo de ella?\n");
        cad.append("¡Gentes del alto llano numantino\n");
        cad.append("que a Dios guardáis como cristianas viejas,\n");
        cad.append("que el sol de España os llene\n");
        cad.append("de alegría, del luz y de riqueza!\n");

        try {
                calendarEntry = gCal.getCalendar("KodamaAwake");
        } catch (IOException e1) {
                e1.printStackTrace();
        }

        EventEntry eventEntry = new EventEntry();
        eventEntry.summary = cad.toString();

        when = new When();

        GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

        when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

        cal.add(GregorianCalendar.MINUTE, 1);

        when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

        eventEntry.when = when;
        try {
                gCal.getClient()
                                .eventFeed()
                                .insert()
                                .execute(new CalendarUrl(calendarEntry.getEventFeedLink()),
                                                eventEntry);
        } catch (IOException e) {
                e.printStackTrace();
        }

        EventEntry event = chequearEventos().get(0);

        String descRead = event.summary;

        Log.i("Max. Length", String.valueOf(descRead.length()));

}

// Testing the lenth of the attribute summary or description with a longer text input
@Test
public void testLenthDescriptionLonger() {

        StringBuilder cad = new StringBuilder();
        cad.append("Antonio Machado\n");
        cad.append("¡Soria fría, Soria pura,\n");
        cad.append("cabeza de Extremadura,\n");
        cad.append("con su castillo guerrero\n");
        cad.append("arruinado, sobre el Duero;\n");
        cad.append("con sus murallas roídas\n");
        cad.append("y sus casas denegridas!\n");
        cad.append("¡Muerta ciudad de señores\n");
        cad.append("soldados o cazadores;\n");
        cad.append("de portales con escudos\n");
        cad.append("de cien linajes hidalgos,\n");
        cad.append("y de famélicos galgos,\n");
        cad.append("de galgos flacos y agudos,\n");
        cad.append("que pululan\n");
        cad.append("por las sórdidas callejas,\n");
        cad.append("y a la medianoche ululan,\n");
        cad.append("cuando graznan las cornejas!\n");
        cad.append("¡Soria fría! La campana\n");
        cad.append("Soria, ciudad castellana\n");
        cad.append("¡tan bella! bajo la luna.\n");
        cad.append("VII\n");
        cad.append("¡Colinas plateadas,\n");
        cad.append("grises alcores, cárdenas roquedas\n");
        cad.append("por donde traza el Duero\n");
        cad.append("su curva de ballesta\n");
        cad.append("en torno a Soria, obscuros encinares,\n");
        cad.append("ariscos pedregales, calvas sierras,\n");
        cad.append("caminos blancos y álamos del río,\n");
        cad.append("tardes de Soria, mística y guerrera,\n");
        cad.append("hoy siento por vosotros, en el fondo\n");
        cad.append("del corazón, tristeza,\n");
        cad.append("tristeza que es amor! ¡Campos de Soria\n");
        cad.append("donde parece que las rocas sueñan,\n");
        cad.append("conmigo vais! ¡Colinas plateadas,\n");
        cad.append("grises alcores, cárdenas roquedas!..\n");
        cad.append("He vuelto a ver los álamos dorados,\n");
        cad.append("álamos del camino en la ribera\n");
        cad.append("del Duero, entre San Polo y San Saturio,\n");
```

```
cad.append("tras las murallas viejas\n");
cad.append("de Soria barbacana\n");
cad.append("hacia Aragón, en castellana tierra.\n");
cad.append("Estos chopos del río, que acompañan\n");
cad.append("con el sonido de sus hojas secas\n");
cad.append("el son del agua, cuando el viento sopla,\n");
cad.append("tienen en sus cortezas\n");
cad.append("grabadas iniciales que son nombres\n");
cad.append("de enamorados, cifras que son fechas.\n");
cad.append("¡Álamos del amor que ayer tuvisteis\n");
cad.append("de ruiseñores vuestras ramas llenas;\n");
cad.append("álamos que seréis mañana liras\n");
cad.append("del viento perfumado en primavera;\n");
cad.append("álamos del amor cerca del agua\n");
cad.append("que corre y pasa y sueña,\n");
cad.append("álamos de las márgenes del Duero,\n");
cad.append("conmigo vais, mi corazón os lleva!\n");
cad.append("¡Oh!, sí, conmigo vais, campos de Soria,\n");
cad.append("tardes tranquilas, montes de violeta,\n");
cad.append("alamedas del río, verde sueño\n");
cad.append("del suelo gris y de la parda tierra,\n");
cad.append("agria melancolía\n");
cad.append("de la ciudad decrépita,\n");
cad.append("me habéis llegado al alma,\n");
cad.append("¿o acaso estabais en el fondo de ella?\n");
cad.append("¡Gentes del alto llano numantino\n");
cad.append("que a Dios guardáis como cristianas viejas,\n");
cad.append("que el sol de España os llene\n");
cad.append("de alegría, del luz y de riqueza!\n");
cad.append("\nCervantes\n");
cad.append("Dichosa edad y siglos dichosos aquéllos a quien los antiguos pusieron");
cad.append(" nombre de dorados, y no porque en ellos el oro, que en esta nuestra edad")
    ;
cad.append(" de hierro tanto se estima, se alcanzase en aquella venturosa sin fatiga");
cad.append(" alguna, sino porque entonces los que en ella vivían ignoraban estas");
cad.append(" dos palabras de tuyo y mío. Eran en aquella santa edad todas las cosas");
cad.append(" comunes. A nadie le era necesario para alcanzar su ordinario sustento");
cad.append(" tomar otro trabajo que alzar la mano y alcanzarle de las robustas encinas,
    ");
cad.append(" que liberalmente les estaban convidando con su dulce y sazonado fruto.");
cad.append(" Las claras fuentes y corrientes ríos, en magnífica abundancia,");
cad.append(" sabrosas y transparentes aguas les ofrecían. En las quiebras");
cad.append(" de las peñas y en lo hueco de los árboles formaban su república las");
cad.append(" solícitas y discretas abejas, ofreciendo a cualquiera mano,");
cad.append(" sin interés alguno, la fértil cosecha de su dulcísimo trabajo.");
cad.append(" Los valientes alcornoques despedían de sí, sin otro artificio que");
cad.append(" el de su cortesía, sus anchas y livianas cortezas, con que se");
cad.append(" comenzaron a cubrir las casas, sobre rústicas estacas sustentadas,");
cad.append(" no más que para la defensa de las inclemencias del cielo.");
cad.append(" Todo era paz entonces, todo amistad, todo concordia: aún no se había");
cad.append(" atrevido la pesada reja del corvo arado a abrir ni visitar las entrañas");
cad.append(" piadosas de nuestra primera madre; que ella, sin ser forzada, ofrecía,");
cad.append(" por todas las partes de su fértil y espacioso seno, lo que pudiese");
cad.append(" hartar, sustentar y deleitar a los hijos que entonces la poseían.");
cad.append(" Entonces sí que andaban las simples y hermosas zagalejas de valle");
cad.append(" en valle y de otero en otero, en trenza y en cabello, sin más vestidos");
cad.append(" de aquéllos que eran menester para cubrir honestamente lo que la ");
cad.append(" honestidad quiere y ha querido siempre que se cubra, y no eran sus");
cad.append(" adornos de los que ahora se usan, a quien la púrpura de Tiro y la");
cad.append(" por tantos modos martirizada seda encarecen, sino de algunas hojas");
cad.append(" verdes de lampazos y yedra, entretejidas, con lo que quizá iban tan");
cad.append(" pomposas y compuestas como van agora nuestras cortesanas con las raras");
cad.append(" y peregrinas invenciones que la curiosidad ociosa les ha mostrado.");

try {
        calendarEntry = gCal.getCalendar("KodamaAwake");
} catch (IOException e1) {
        e1.printStackTrace();
}

EventEntry eventEntry = new EventEntry();
eventEntry.summary = cad.toString();

when = new When();

GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

cal.add(GregorianCalendar.MINUTE, 1);

when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

eventEntry.when = when;
try {
        gCal.getClient()
                        .eventFeed()
                        .insert()
                        .execute(new CalendarUrl(calendarEntry.getEventFeedLink()),
                                        eventEntry);
} catch (IOException e) {
        e.printStackTrace();
}
```

```
                EventEntry event = chequearEventos().get(0);

                String descRead = event.summary;

                Log.i("Max. Length", String.valueOf(descRead.length()));

        }

        private void crearEvento(StringBuilder cad) {

                try {
                        calendarEntry = gCal.getCalendar("KodamaAwake");
                } catch (IOException e1) {
                        e1.printStackTrace();
                }
                EventEntry eventEntry = new EventEntry();
                eventEntry.title = "testLenthName" + ";" + cad.toString();

                When when = new When();

                GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

                when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

                cal.add(GregorianCalendar.MINUTE, 1);

                when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

                eventEntry.when = when;

                String eventFeedLink = calendarEntry.getEventFeedLink();

                CalendarUrl calUrl = new CalendarUrl(eventFeedLink);

                try {
                        gCal.getClient().eventFeed().insert().execute(calUrl, eventEntry);
                } catch (IOException e) {
                        e.printStackTrace();
                }

        }

        private List<EventEntry> chequearEventos() {

                List<EventEntry> events = null;
                try {
                        events = gCal.getClient().eventFeed().list()
                                        .execute(new CalendarUrl(calendarEntry.getEventFeedLink())).
                                            events;
                } catch (IOException e) {
                        e.printStackTrace();
                }

                return events;
        }

}
```

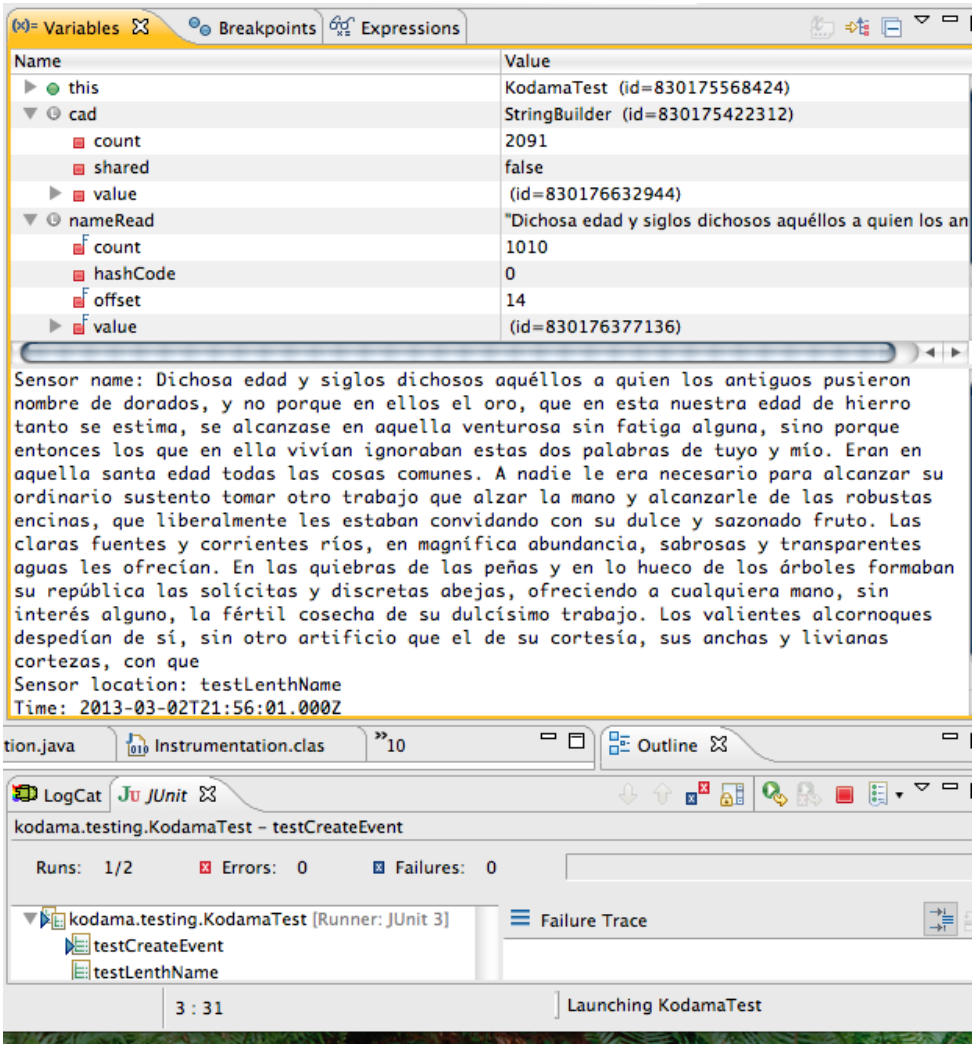

Figura 6.1: Longitud del atributo title 1010 caracteres

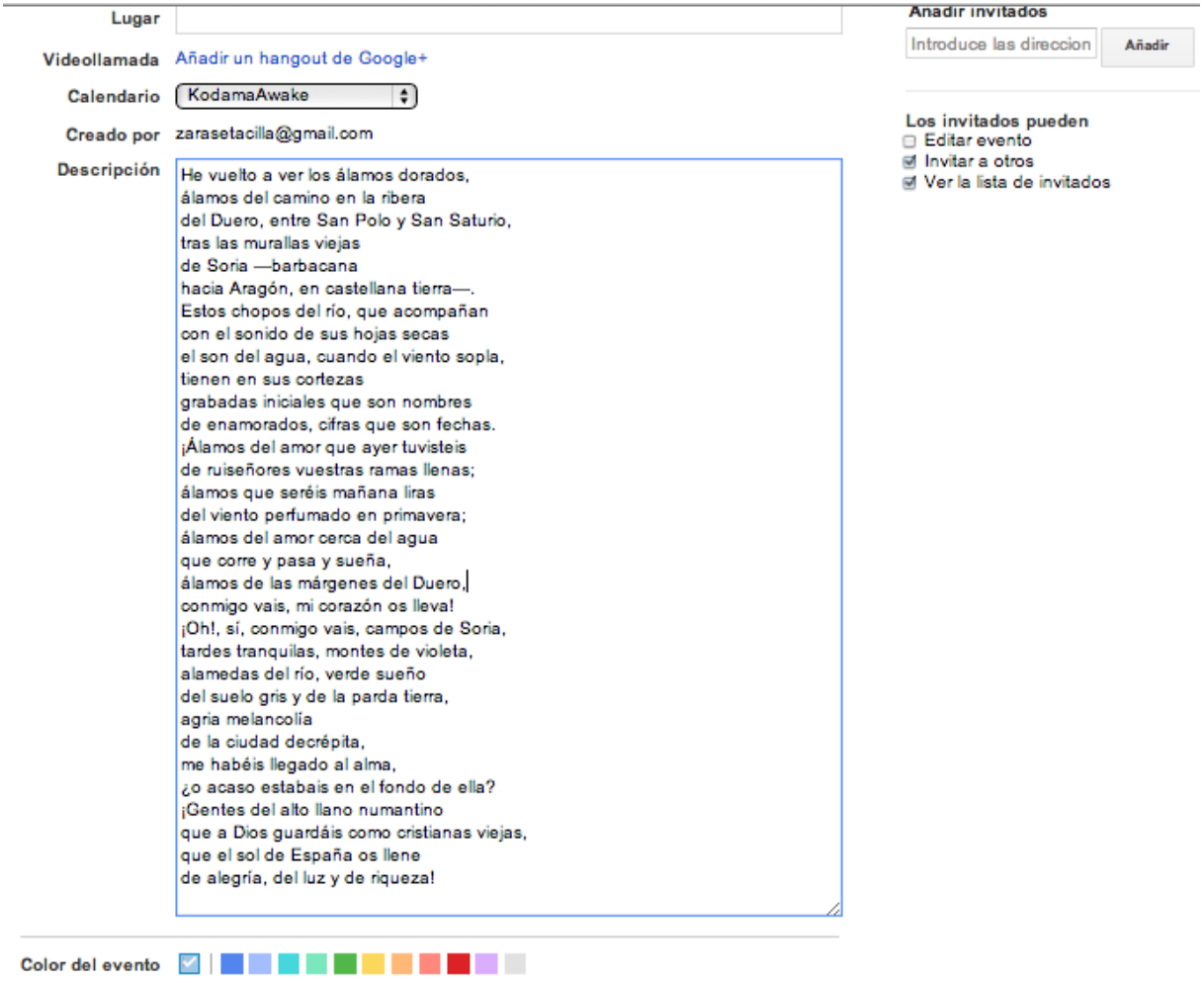

Figura 6.2: Captura de evento con descripción

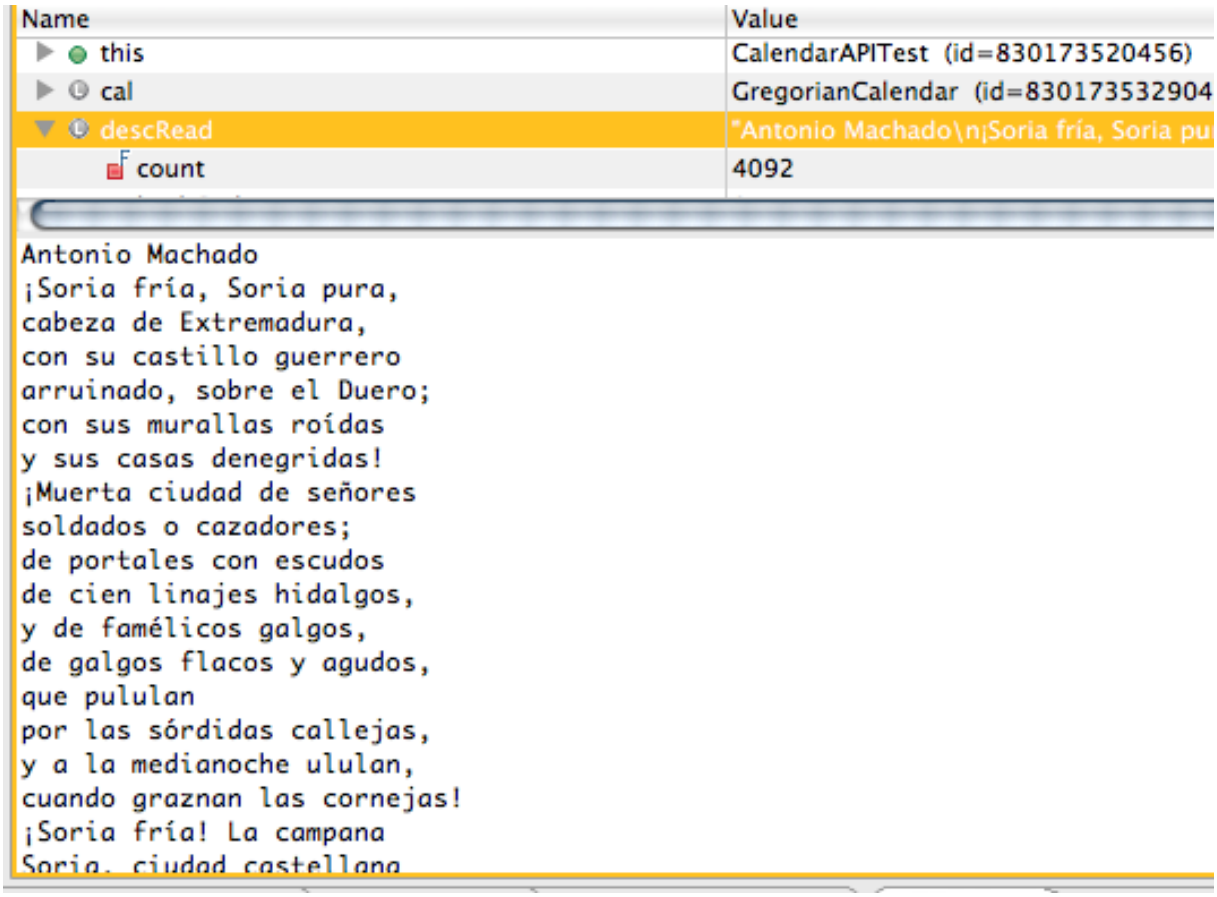

Figura 6.3: Longitud aceptada por la descripción: 4092 caracteres

NotificationTest.java

Paquete kodama.testing 6.3: NotificationTest.java

```
package kodama.testing;

import java.util.List;

import kodama.main.Main;
import kodama.receiver.model.Notification;
import kodama.synchronization.CalendarManager;

import org.junit.Test;

import android.test.ActivityInstrumentationTestCase2;

import com.android.ddmlib.Log;
import com.google.api.client.util.DateTime;

/**
 *
 * @author saragarciaperez
 *         <p>
 *          Unit Test for Notifications
 *          </p>
 */
public class NotificationTest extends ActivityInstrumentationTestCase2<Main> {

        public NotificationTest() throws ClassNotFoundException {
                super("kodama.main.Main", Main.class);
        }

        CalendarManager calManager = null;
        DateTime date = null;
```

```
        protected void setUp() {

                try {

                        super.setUp();

                } catch (Exception e) {
                        e.printStackTrace();
                }

                getActivity();

                calManager = Main.getGestorCalendar();

                // It's neccesary wait here when debug, because we need time to set the
                // calendar and to start to work with it.
                // Then we have to select the account of gmail and one the app is
                // loaded, continue (Step over)

                testCheckEvents(0);

                testCheckEvents(0);
                testCreateEvent();
                testCheckEvents(1);
                testCreateEvent();
                testCreateEvent();
                testCheckEvents(2);

                long ms1 = System.currentTimeMillis();

                // testing also the performance where we have a lot of events

                int cont = 10;
                do {
                        testCreateEvent();
                        cont--;
                } while (cont > 0);

                long ms2 = System.currentTimeMillis();

                Log.i("Create 10 events", String.valueOf(ms2 - ms1));

                long ms3 = System.currentTimeMillis();

                testCheckEvents(10);

                long ms4 = System.currentTimeMillis();

                Log.i("Check new 10 events", String.valueOf(ms4 - ms3));

                testCheckFormatedDate();
        }

        @Test
        public void testCreateEvent() {

                calManager.setCalendar();

                calManager.createEvent("here", "test");
        }

        @Test
        public void testCheckEvents(int num) {

                if (calManager.checkEvents() != null)
                        assertEquals(num, calManager.checkEvents().size());
                else
                        assertEquals(num, 0);
        }

        //We use a simple reg Exp to check the format of the date
        public void testCheckFormatedDate() {

                calManager.createEvent("NotificationTest", "testCheckFormatTime");

                try {
                        Thread.sleep(2000);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }

                List<Notification> notifs = calManager.checkEvents();

                String time = notifs.get(0).getTime();

                assertEquals(true,
                                time.matches("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}"));

        }
}
```

SensorSensibilityTest.java

Paquete kodama.testing 6.4: SensorSensibilityTest.java

```
package kodama.testing;

import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Calendar;
import java.util.GregorianCalendar;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.UnsupportedAudioFileException;
import org.junit.Test;


import kodama.sensor.audio.SoundManager;

/**
 *
 * @author saragarciaperez
 *
 *
 *          Testing the sensibility of the sensor
 */
public class SensorSensibilityTest {

        public SensorSensibilityTest() {
                setTestEnvironment();
        }

        private SoundManager soundManager;
        private ByteArrayOutputStream out = new ByteArrayOutputStream();
        private AudioInputStream in;

        protected void setUp() {
                System.out
                                .println("TEST-SENSIBILITY --> dB Full Scale [Min: -100, Max: 0]\n");

                try {

                        // desde micro -55.0 -diferencia => 0.73
                        testSounds("src/test/grifocerca.wav");
                        testSounds("src/test/grifolejos.wav");
                        // With the sensor near the principal door, portero in kitchen
                        testSounds("src/test/portero.wav");
                        testSounds("src/test/toctoc.wav");
                        testSounds("src/test/comedor.wav");
                        testSounds("src/test/tele.wav");
                        testSounds("src/test/oficinahoracomida.wav");
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (UnsupportedAudioFileException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        public void setTestEnvironment() {
                soundManager = new SoundManager();
        }

        @Test
        public void testSounds(String file) throws FileNotFoundException,
                        UnsupportedAudioFileException, IOException {

                in = AudioSystem.getAudioInputStream(new FileInputStream(file));

                int read;
                byte[] buff = new byte[1024];
                while ((read = in.read(buff)) > 0) {
                        out.write(buff, 0, read);
                }
                out.flush();

                byte[] audioBytes = out.toByteArray();

                // double decibelios = soundManager.calculatePowerDb(audioBytes);

                // System.out.println("Decibelios: "+ String.valueOf(decibelios));

                processAudio(audioBytes, file);
                out.reset();
        }

        // ********************************************************************* //
        // Main Loop of sensor activity (Testing)
        // ********************************************************************* //
```

```
        /**
         * Processes audio, calculates dB and calls the CalendarManager to create a
         * new event if it takes place
         *
         * @param buffer
         */
        private void processAudio(byte[] buffer, String file) {

                double dBTransformation = 0;
                double currentPower = soundManager.calculatePowerDb(buffer);

                dBTransformation = currentPower + 0.73;

                /*
                 * <string-array name="sensibility_array"> <item>@string/high</item>
                 * <item>@string/medium</item> <item>@string/low</item> </string-array>
                 *
                 *
                 * <string-array name="sensibility_array_values"> <item>-80</item> <!--
                 * fijar limite inferior --> <item>-58</item> <!-- mayor que -58 ,
                 * limite sensibiliad media--> <item>-55</item> <!-- es mayor
                 * que -55.8 => sensibilidad baja --> </string-array>
                 */

                System.out.println(file + " Current power: "
                                + String.valueOf(dBTransformation));

                if (currentPower > -80) {

                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        System.out.println("HIGH > -80");
                }

                if (currentPower > -58) {

                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        System.out.println("MEDIUM > -58");
                }

                if (currentPower > -55) {

                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        System.out.println("LOW > -55");
                }
                System.out.println("\n");
        }

        public static void main(String[] args) {
                SensorSensibilityTest sst = new SensorSensibilityTest();
                sst.setUp();
        }
}
```

6.1.1. Resultados de los test de sensibilidad

Los decibelios se miden en Full Scale y no tienen una correspondencia directa con los decibelios de presión, que son de los que hablamos habitualmente para referirnos a la intensidad del sonido. Por esto, se han realizado pruebas de sensibilidad, para fijar los umbrales.

```
TEST-SENSIBILITY --> dB Full Scale [Min: -100, Max: 0]

src/test/grifocerca.wav Current power: -55.73390946311129
HIGH > -80
MEDIUM > -58
LOW > -55.8


src/test/grifolejos.wav Current power: -55.91180307185827
HIGH > -80
MEDIUM > -58


src/test/portero.wav Current power: -58.081443659486226
HIGH > -80


src/test/toctoc.wav Current power: -56.84969339484929
HIGH > -80
MEDIUM > -58


src/test/comedor.wav Current power: -55.71921198904462
HIGH > -80
MEDIUM > -58
LOW > -55.8


src/test/tele.wav Current power: -55.93249314433159
HIGH > -80
MEDIUM > -58


src/test/oficinahoracomida.wav Current power: -55.90987364572298
HIGH > -80
MEDIUM > -58
```

Figura 6.4: Resultados de los tests de sensibilidad

6.2. Pruebas de integración

Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes. El objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes.

6.2.1. Planificación

A continuación se describe el plan de pruebas de integración para el proyecto.

Caso de uso C.0.: Configuración de la Apk

Escenario E.0.: seleccionar cuenta	
Caso de prueba CP.0.1.	
Entrada	**Resultado esperado**
Seleccionamos cuenta (si hay más de una en el dispositivo)	Los eventos se almacenan en el calendario de la cuenta correspondiente

Caso de uso C.1.1: Activar sensor

Escenario E.1.1: activar sensor	
Caso de prueba CP.1.1.1	
Entrada	**Resultado esperado**
Evento detectado	Se almacena en Google Calendar con la hora correcta y los datos de configuración

Caso de uso C.1.3: Configurar sensor

Escenario E.1.3: configurar sensor	
Caso de prueba CP.1.3.2	
Entrada	**Resultado esperado**
Se cambian la sensibilidad del sensor	Se observa, con la ayuda de puntos de interrupción, que el sensor se ajusta a la sensibilidad de la nueva configuración

Caso de uso C.2.1: Activar notificaciones

Escenario E.2.1: activar notificaciones	
Caso de prueba CP.2.1.1	
Entrada	**Resultado esperado**
Se activa el sensor y el receptor	Recibimos notificación con datos correctos si hay ruido
Caso de prueba CP.2.1.2	
Entrada	**Resultado esperado**

Se recibe una notificación de evento, ya que existe una nueva entrada en Google Calendar	El sensor se para temporalmente (si está activo) para evitar capturar el sonido de la notificación

Caso de uso C.2.3: Visualizar notificaciones

Escenario E.1.3: visualizar notificaciones	
Caso de prueba CP.2.3.1	
Entrada	**Resultado esperado**
Nos llega una notificación y tocamos la pantalla para avisar de que la hemos recibido	Se elimina de la pantalla, pero quedará almacenada en el registro

Caso de uso C.2.4: Ver registro

Escenario E.2.4: ver registro de notificaciones	
Caso de prueba CP.2.4.1	
Entrada	**Resultado esperado**
Nos llega una notificación y vamos a ver registro	Aparece en la última línea del registro

6.2.2. Desarrollo

Las pruebas de integración se desarrollaron en forma paralela a la implementación del sistema. A lo largo de este proceso las pruebas fueron útiles para detectar errores en la codificación que se corrigieron en el momento.

6.3. Pruebas de sistema

Las pruebas del sistema son pruebas de integración del sistema construido completo, que permiten probar el conjunto de todo el sistema y que sus relaciones con otros sistemas que necesite son correctas, verificando así que todas sus especificaciones funcionales y técnicas se cumplen.

6.3.1. Planificación

Manejo sensor

Entrada	**Resultado esperado**
Activación sensor	Se enciende el toggle button RECORD
Activación sensor	Se genera la visualización de los decibelios del ambiente y se genera un movimiento de la onda de la pantalla al detectar sonido

Cambiamos el nombre y la localización del sensor	El sistema muestra los nuevos datos en la pantalla
Desactivamos el sensor	El sistema deja de mostrar la variación de decibelios y la onda no se mueve
Desactivación sensor	Se apaga el toggle button RECORD

Manejo del receptor: mientras tengamos un sensor activado

Entrada	**Resultado esperado**
Activación de notificaciones	Se enciende el toggle button ON
Activar notificaciones	Mostrar notificación cuando se detecta sonido
Ver registro	El registro muestra datos correctos y completos de los eventos detectados
Desactivación de notificaciones	Se apaga el toggle button ON

6.3.2. Desarrollo

En el desarrollo de las pruebas de sistema se detectó la necesidad de programar el sensor para que no capturara un evento justo al principio, si no un tiempo después. Ya que justo al principio a veces al activarlo se detecta sonido y este no es el comportamiento deseado. Así evitamos almacenar en el google calendar falsos positivos.

Capítulo 7

Pruebas usabilidad

En esta sección se documentan las pruebas de usabilidad llevadas a cabo antes y durante el desarrollo de la aplicación.

La realización de pruebas de usabilidad en este proyecto es muy importante puesto que va a ser utilizado principalmente por personas mayores o discapacitadas. También va a ser usado por sus familiares o cuidadores que pueden pertenecer a cualquier segmento de la población. Por lo tanto, ha de ser comprensible para usuarios que no están familiarizados con este tipo de tecnología.

7.1. Etapas del proyecto y pruebas de usabilidad

El proyecto ha pasado por varias etapas según iba tomando forma. En cada una de ellas se han realizado distintas pruebas de usuario, tales como *Cognitive Walkthrough*, *Think Aloud*.

7.2. Etapa 1, etapa inicial

Se hizo un diseño de las pantallas para realizar una evaluación de usabilidad mediante la técnica de *cognitive walkthrought* o *paseo cognitivo*.

Se comienza evaluando el sistema en términos de las tareas que los usuarios realizarán con ese sistema. Ayudará identificar las metas de los usuarios y sus propósitos en cada tarea. En este caso, los objetivos son: activar el sensor y activar el receptor de notificaciones. Por lo tanto, estos controles tienen que estar fácilmente accesibles e identificables.

Esta técnica ha sido aplicada antes de empezar a desarrollar la aplicación, y antes de realiza su diseño de clases, etc. Ha servido como punto de partida para la fase de diseño y posteriormente para el desarrollo.

A continuación podemos ver el diseño de las pantallas en papel:

Figura 7.1: Pantalla inicial de la aplicación

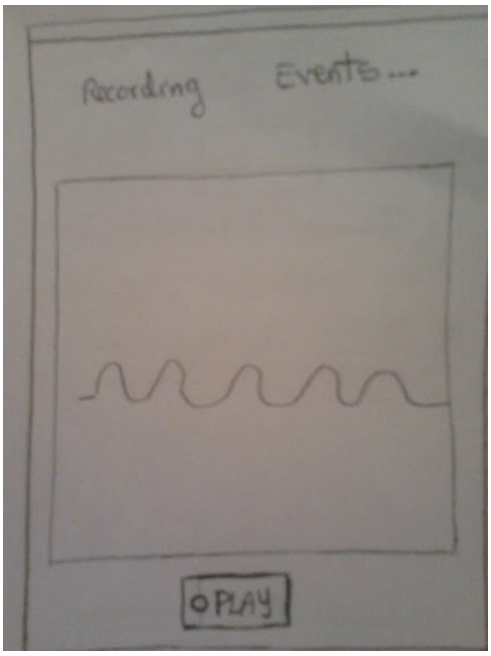

Figura 7.2: Pantalla del modo sensor

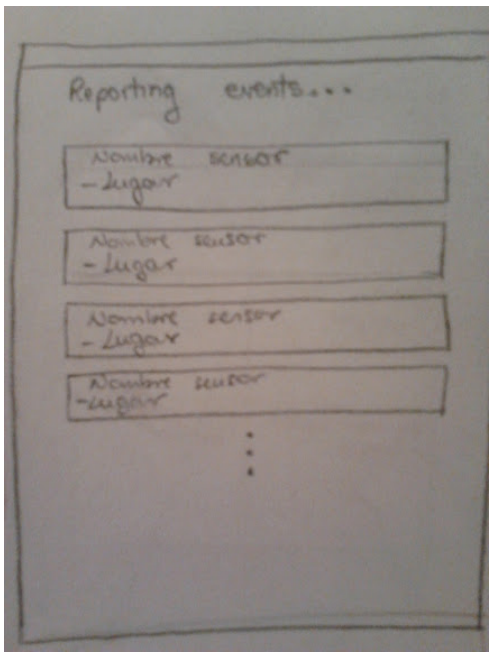

Figura 7.3: Pantalla del modo receptor

Figura 7.4: Pantalla de configuración

7.2.1. Resultados y conclusiones

Pantalla inicial

Figura 7.5: Pantalla inicial de la aplicación

Problemas encontrados en la pantalla inicial	
Problema	**Solución**
Disposición del acceso a las funciones no adecuada	Uso de pestañas para los dos modos. SENSOR - WARNINGS. La pestaña Warning debería parpadear, cambiar de color, etc cuando existan eventos pendientes de notificación.
Texto de botones confuso	Se cambian por nombres más descriptivos por consejo de los usuarios.

Sensor

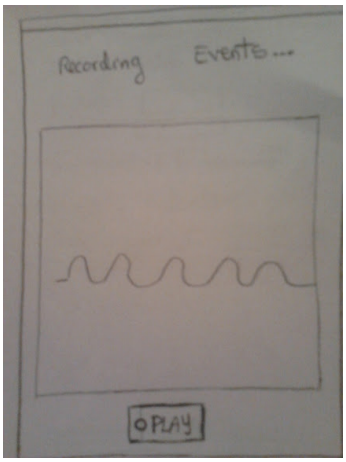

Figura 7.6: Pantalla del modo sensor

Problemas encontrados en la pantalla del sensor	
Problema	**Solución**
Configuración sensor pensada para ser incluida en la configuracion general	Incluir botón de configuración en la pantalla del sensor, es decir, la configuración del sensor, no de la aplicación.

Receptor

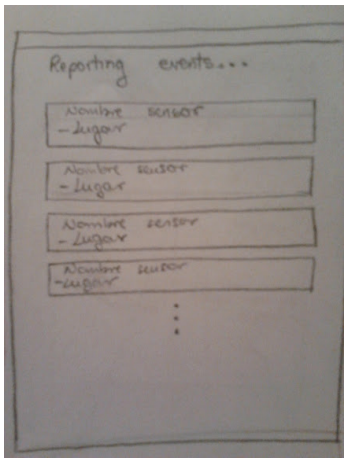

Figura 7.7: Pantalla del modo receptor

Problemas encontrados en la pantalla del receptor	
Problema	**Solución**
Falta acceso a registro de eventos	Se requiere un acceso al registro de eventos pudiendo acceder desde esta pantalla.

Configuración

Figura 7.8: Pantalla de configuración

Problemas encontrados en la pantalla dela configuración	
Problema	**Solución**
No se diseñó una pantalla de preferencias estándar de Android	Un usuario habitual de android resaltó que era más adecuado usar una pantalla estándar de preferencias de android.

7.3. Etapa 2, comienza el diseño y desarrollo

Se llevaron a cabo pruebas de think aloud sobre la parte de la aplicación desarrollada hasta el momento.

Tras analizar las pruebas de usabilidad realizadas al diseño de las pantallas, se empezó a realizar el diseño del sistema.

En la pantalla de Notificaciones de momento sólo aparecerá un mensaje indicando que se ha detectado ruido.

La tarea que tiene que llevar a cabo el usuario es activar el sensor.

En la siguiente imagen vemos la pantalla del sensor correspondiente a esta etapa número 1:

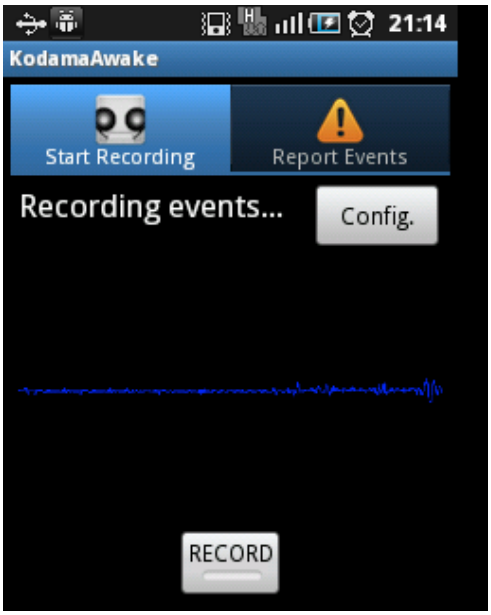

Figura 7.9: Pantalla sensor etapa 1

7.3.1. Problemas encontrados y soluciones

Problemas encontrados en la pantalla del sensor	
Problema	**Solución**
Se detectó que el texto de las pestañas no era apropiado	▪ Start Recording= Sensor ▪ Report Events = Notifications
Botones no están como habitualmente se disponen éstos	los 2 juntos alineados a la derecha
Texto de pantalla de sensor inapropiado	Que indique el nombre y la localización del sensor
Imágenes de las pestañas pueden llevar a confusión	Suprimir imágenes.

7.4. Etapa 3

En esta versión se contemplan todas las funcionalidades salvo la de visualizar el registro de eventos. Al igual que en la etapa anterior, se han realizado pruebas siguiendo la técnica de Think Aloud. Las tareas a realizar eran:

- Activar el sensor
- Cambiar la configuración del sensor
- Activar las notificaciones

7.4.1. Pantallas

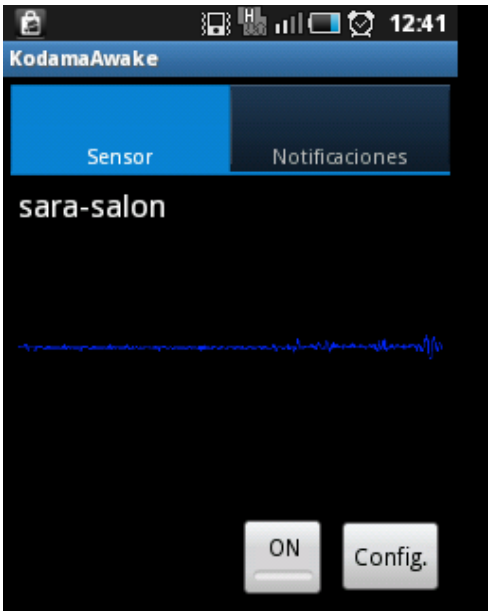

Figura 7.10: Pantalla sensor

Figura 7.11: Pantalla configuración sensor

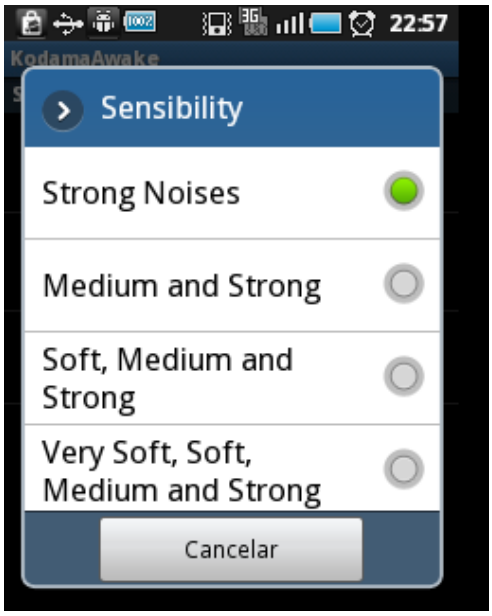

Figura 7.12: Pantalla configuración sensibilidad sensor

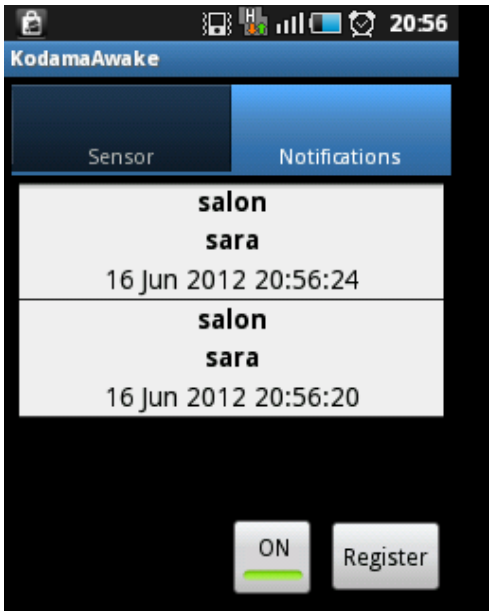

Figura 7.13: Pantalla receptor

Problemas encontrados y soluciones

Problemas encontrados	
Problema	**Solución**
Informacion de la pantalla del sensor incompleta.	En el sensor, uno de los usuarios expresa que desearía poder ver los decibelios detectados en la pantalla, ya que la onda es una animación y no expresa los valores reales.
Fallos en la configuración	▪ Quitar la palabra *Edit* de Edit Sensor Name y Edit Sensor Place. ▪ Cambiar la descripción de Sensibilidad, que es incorrecta. ▪ Cambiar el menú de sensibilidad: high, medium, low.

7.5. Etapa 4, prototipo

Esta versión final que contiene el 100 % de la funcionalidad pensada para este prototipo. Se incluyen las pantallas a continuación para que se puedan observar los cambios efectuados a lo largo de las pruebas de usabilidad.

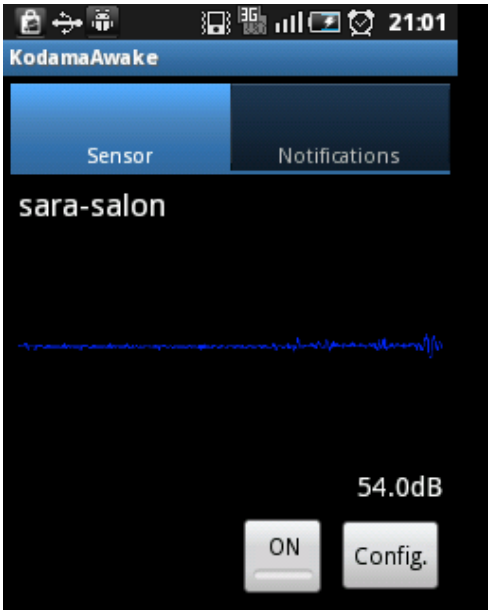

Figura 7.14: Pantalla final del sensor

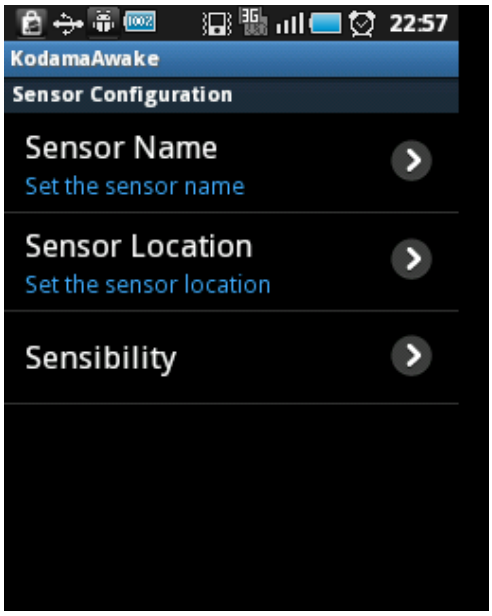

Figura 7.15: Pantalla final de la configuración del sensor

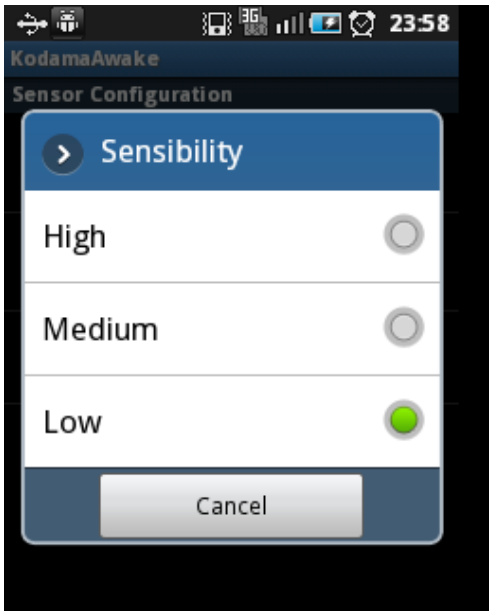

Figura 7.16: Pantalla final de la configuración de la sensibilidad del sensor

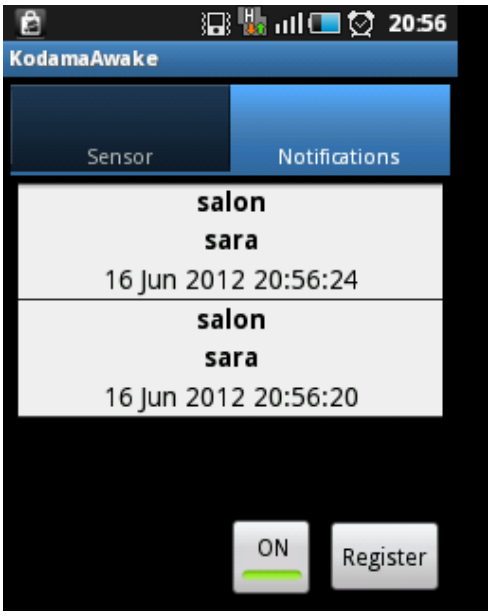

Figura 7.17: Pantalla final del receptor

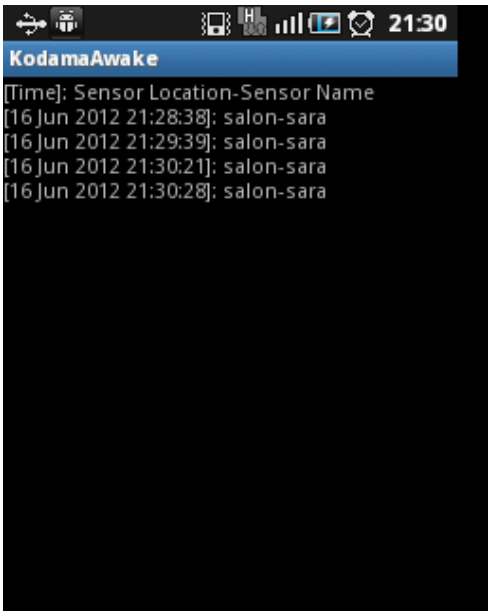

Figura 7.18: Pantalla final del registro de eventos

7.6. Pruebas en escenarios

Aquí se documentarán las pruebas realizadas tratando de simular un entorno real. Se utilizan para ello varios dispositivos sincronizados con un mismo calendario.

Cocina

Se deja activado un sensor en la cocina desde la noche de un sábado hasta la noche del domingo siguiente. Vamos a observar los eventos detectados. Las horas a las que se han guardado eventos nos indican cuando ha habido actividad en la cocina además de aportarnos información adicional cómo el tiempo durante el cual se ha prolongado la actividad o si ha sido un evento puntual.

Con esta información diaria podríamos nutrir un posible servicio Web (creado en el futuro) que nos ayudara a establecer patrones de estilo de vida e hiciera de nuestro sistema de detección de eventos un sistema con una inteligencia más autónoma.

A continuación podemos ver los resultados de nuestros tests para el escenario *Cocina*

Tenemos algunos eventos aislados de madrugada 7.19. El gato debe de haber hecho ruido al despertarse. El resto de sonidos se corresponderían con la actividad normal de la mañana. El ruido al cerrar el frigorífico sería uno de los eventos registrados.

Hay ruido a la hora de la comida 7.20.

Hay ruido sobre todo a la hora de la cena 7.21.

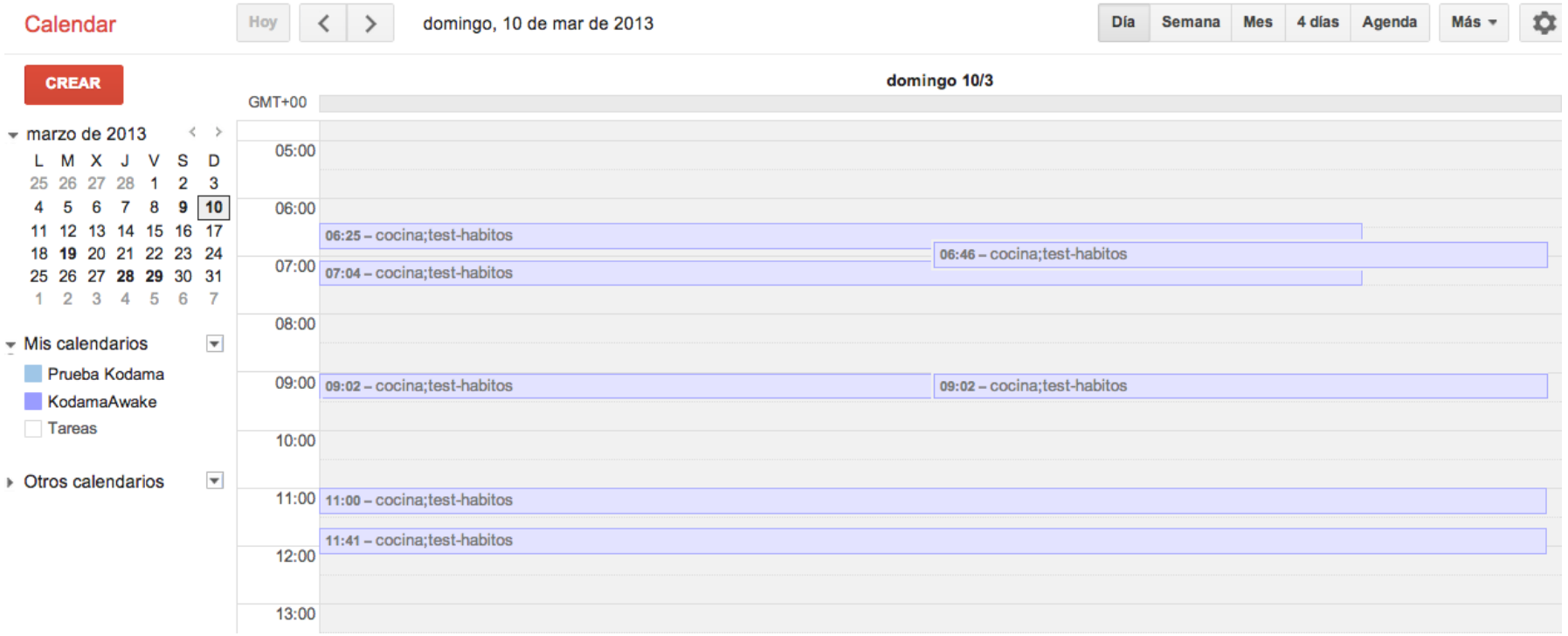

Figura 7.19: Eventos en la cocina por la mañana

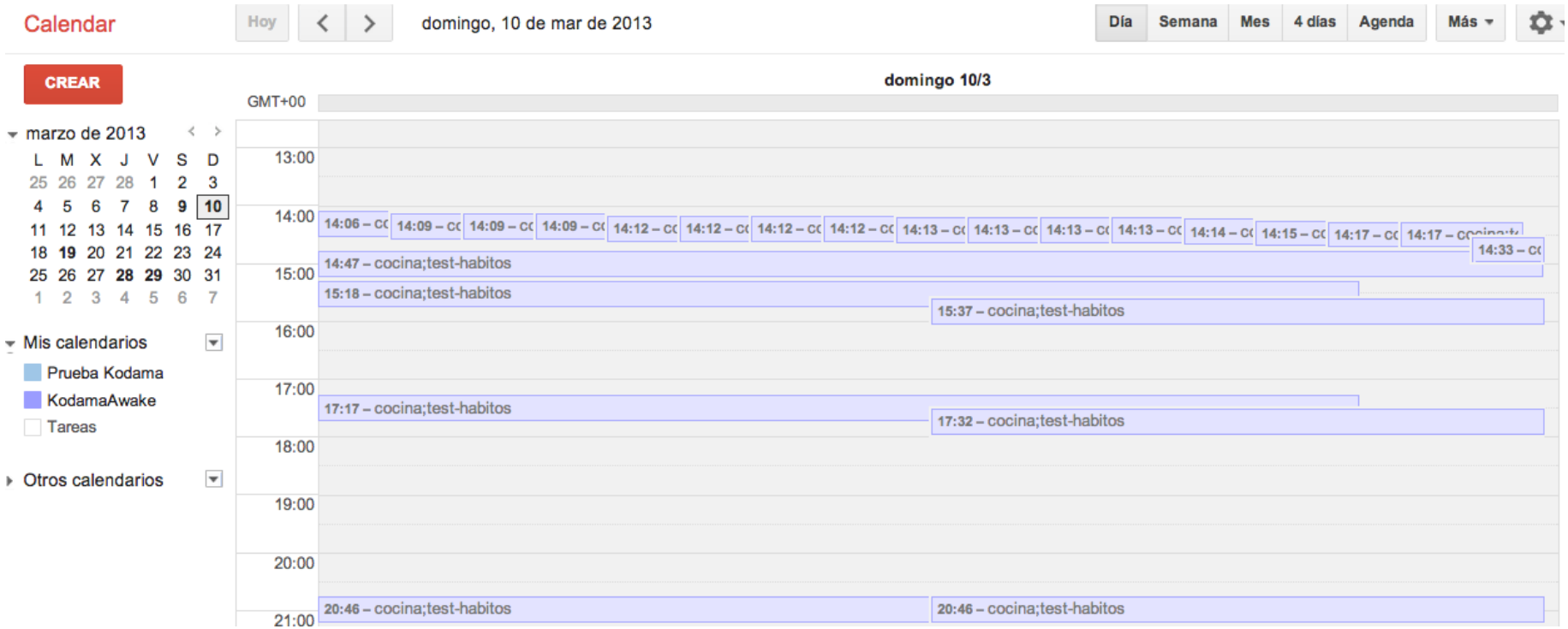

Figura 7.20: Eventos en la cocina por la tarde

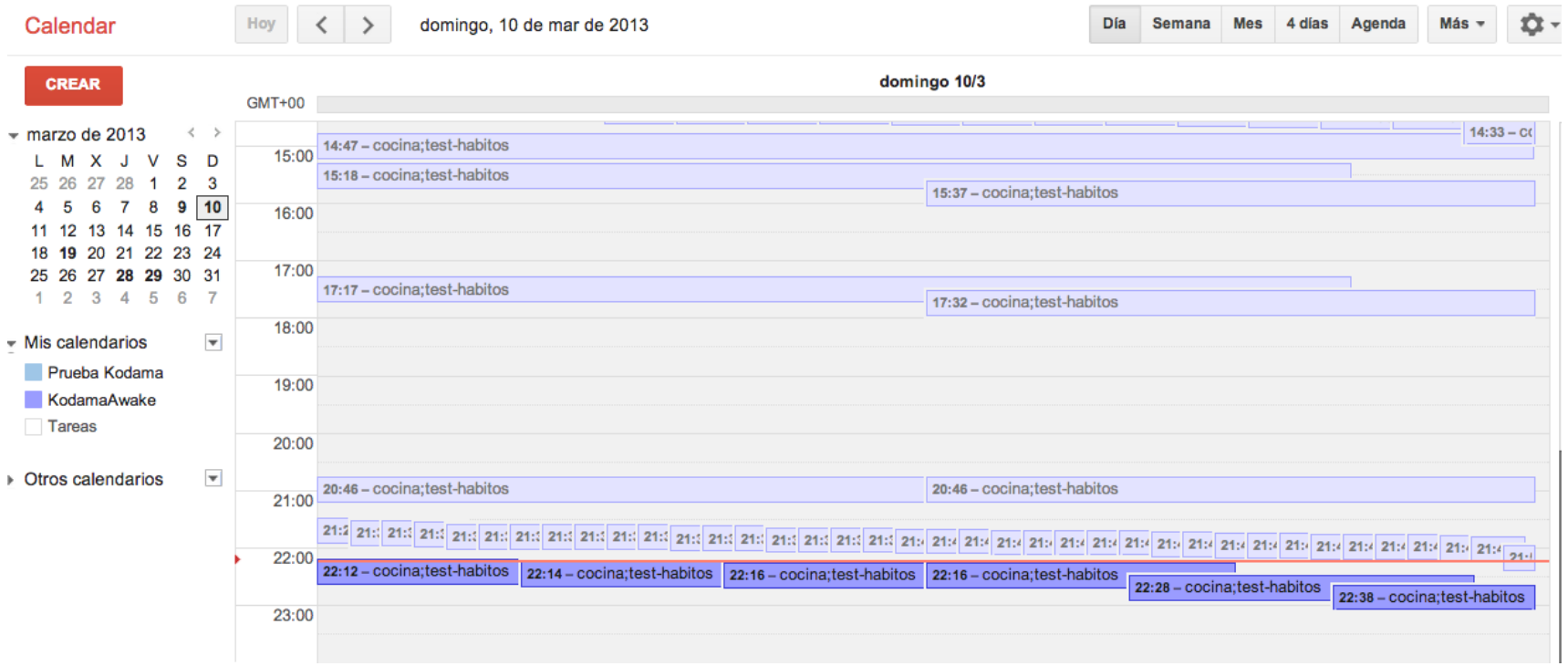

Figura 7.21: Eventos en la cocina por la noche

Capítulo 8

Pruebas accesibilidad

En esta sección se documentan las pruebas de accesibilidad llevadas a cabo antes. Éstas han sido realizadas mediante la comprobación de los ítems de una lista de revisión. Ésta la extraemos de *qweos.net*. Se trata de una traducción amigable de las normas WCAG 2.0 oficiales. Para facilitar la lectura no incluimos aquí aquellos ítems no aplicables (N/A) a este prototipo. Se podría considerar que estamos ante un nivel de accesibilidd WAI 2, aunque no se cumplan puntos de este nivel porque no aplican realmente.

8.1. Contenido perceptible

El contenido debe estar disponible (ser perceptible) para los sentidos: vista, audición, y/o tacto.

Pauta 1.1. Alternativas textuales para todo el contenido no textual.	
Recomendación (Nivel A)	**Cumplimiento**
Las imágenes que no transmitan contenidos, sean decorativas o con el contenido ya presente como texto se ofrecerán con el texto alternativo vacío (alt=) o aplicadas como fondos de imagen CSS. Todas las imágenes enlazadas contarán con un texto descriptivo alternativo.	Sí
Los botones de los formularios tendrán nombres (value) descriptivos.	Sí

- Cuando detectamos un sonido se mueve la imagen de onda y cambian los decibelios pero no hay ningún mensaje de texto que se pueda pasar a "text to speech"para indicar este evento en el sensor.
- Los textos de los botones de la aplicación son descriptivos.

Pauta 1.3. Contenido presentado de diferentes maneras.	
Recomendación (Nivel A)	**Cumplimiento**
El orden de navegación y lectura (determinado por el orden en el código fuente) será lógico e intuitivo.	Sí

Las instrucciones no dependerán de la forma, tamaño o ubicación visual (por ejemplo, "Haga clic en el icono cuadrado para continuar.º "Las instrucciones están en la columna de la derecha").	Sí
Las instrucciones no dependerán del sonido (por ejemplo, Ün sonido beep le indica que puede continuar").	Sí

Pauta 1.4. Facilite a los usuarios el acceso al contenido.	
Recomendación (Nivel A)	**Cumplimiento**
No use el color como el único método para transmitir el contenido o distinguir elementos visuales.	Sí
Los enlaces deben distinguirse de los elementos y texto que les rodean. Si utiliza el color para diferenciar los enlaces, use una forma adicional para distinguirlos. (por ejemplo, se subrayan cuando reciben el foco).	Sí

8.2. Aplicación operable

Pauta 2.3. No diseñe contenidos que provoquen convulsiones	
Recomendación (Nivel A)	**Cumplimiento**
No deberá crear contenidos que destellen más de tres veces por segundo a menos que el parpadeo sea lo suficientemente pequeño, los destellos sean de bajo contraste y no contengan demasiado rojo.	Sí
Recomendación (Nivel AAA)	**Cumplimiento**
No deberá crear contenidos que destellen más de tres veces por segundo.	Sí

Pauta 2.4 Ofrezca ayuda al usuario para encontrar el contenido	
Recomendación (Nivel A)	**Cumplimiento**
El orden de la navegación por los enlaces, elementos de los formularios, etc. deberá ser lógico e intuitivo.	Sí

- Los 2 modos de la aplicación están accesibles desde la misma pantalla (layout de pestañas).
- En la propia pantalla del sensor disponemos de un botón para acceder a la configuración de éste.
- En la propia pantalla de las notificaciones podemos acceder al Google Calendar para verlas todas.

8.3. Aplicación comprensible

El contenido y la interfaz deben poder entenderse fácilmente.

Pauta 3.1 Cree contenidos legibles y fáciles de entender	
Recomendación (Nivel A)	**Cumplimiento**
El idioma principal de la página deberá estar identificado utilizando el atributo lang de HTML (por ejemplo, ¡HTML lang=.es»).	Sí

- El idioma principal viene indicado en la página de Google calendar.
- Se trata de una aplicación con internacionalización: inglés, español.
- Si la configuración del dispositivo tiene fijado el idioma *español*, éste será seleccionado automáticamente. Si no, se usará por defecto el inglés.

8.4. Sistema robusto

El contenido debe ser lo suficientemente consistente y fiable como para permitir su uso con una amplia variedad de agentes de usuario, ayudas técnicas y preparado para las tecnologías venideras.

Pauta 4.1. Mejore la compatibilidad con los agentes.	
Recomendación (Nivel A)	**Cumplimiento**
Se deberán evitar los errores de sintaxis de HTML/X-HTML. El código puede comprobarse, analizarse y validarse a través de http://validator.w3.org/	Sí
Deberá utilizar el marcado de tal forma que se facilite la accesibilidad. Esto incluye el seguir las especificaciones oficiales de HTML/XHTML, utilizando la gramática formal de forma apropiada.	Sí

8.5. Otras observaciones

- Notificación mediante sonido, texto y vibración.
- Accesibilidad implícita que ofrece la pantalla táctil.
- Se utilizan los estándares de Android.
- Compatibilidad con versiones venideras.

Capítulo 9

Presupuesto y Plan Temporal

En este capítulo se muestra el plan temporal y el presupuesto del proyecto.

En las siguientes figuras 9.1, 9.2, 9.3, 9.4 se muestra el diagrama Gantt de la planificación del proyecto. Debido a su tamaño, el diagrama aparece dividido en cuatro. En el diagrama se muestran las tareas y las actividades que componen cada una. El proyecto será realizado durante los fines de semana y abarca desde finales de diciembre hasta junio.

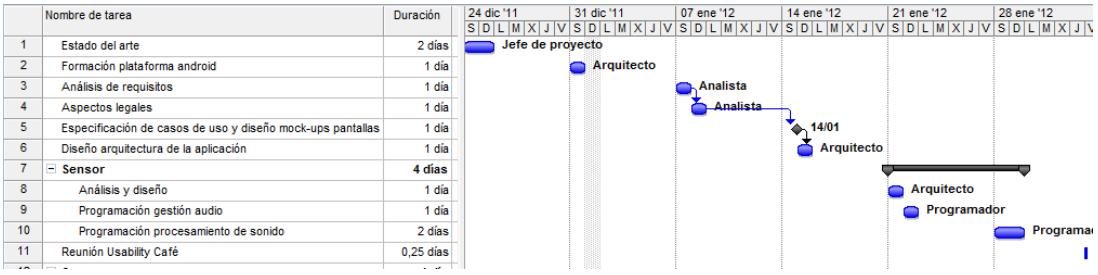

Figura 9.1: Planificación del proyecto (1)

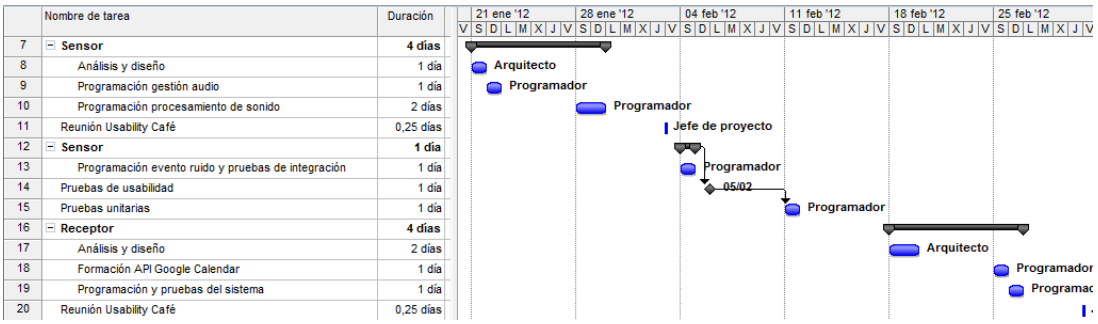

Figura 9.2: Planificación del proyecto (2)

A continuación, en la figura 9.5 se indican los recursos y el tiempo de cada tarea, así como el beneficio, coste total y presupuesto final aplicando el 21 % de IVA.

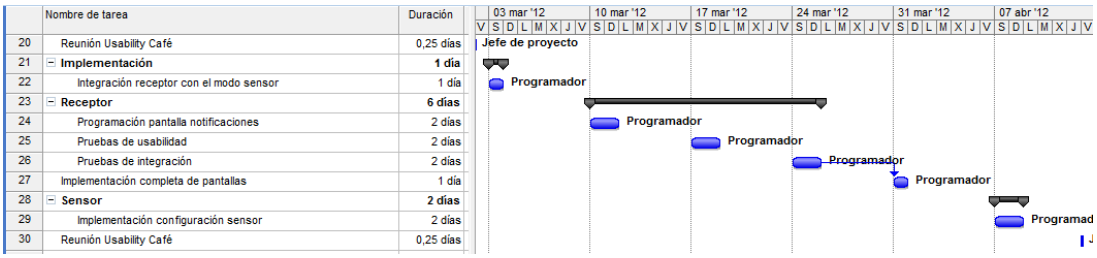

Figura 9.3: Planificación del proyecto (3)

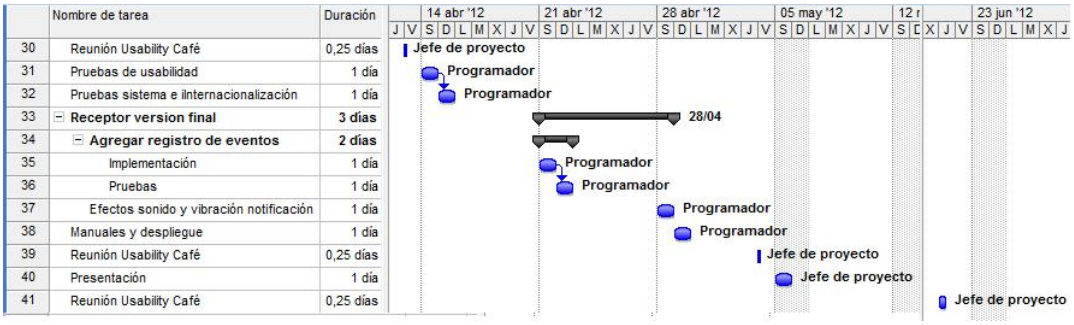

Figura 9.4: Planificación del proyecto (4)

Recurso		Tiempo (horas)	Tarifa	Coste			
Jefe de proyecto			*30,00 €*	*18,00 €*			
Reuniones Usability Café		10	300,00 €	180,00 €			
Estado del arte		16	480,00 €	288,00 €			
Presentación		8	240,00 €	144,00 €			
	TOTAL	**34**	1.050,00 €	630,00 €	**Beneficio**	420,00 €	
Arquitecto			30,00 €	12,00 €			
Formación plataforma android		8	240,00 €	96,00 €			
Diseño arquitectura de la aplicación		8	240,00 €	96,00 €			
Análisis y diseño módulos		24	720,00 €	288,00 €			
	TOTAL	**40**	1.200,00 €	480,00 €	**Beneficio**	720,00 €	
Analista			25,00 €	14,00 €			
Análisis de requisitos		8	200,00 €	112,00 €			
Aspectos legales		8	200,00 €	112,00 €			
Especificación casos de uso y mock-ups pantallas		8	200,00 €	112,00 €			
	TOTAL	**24**	600,00 €	336,00 €	**Beneficio**	264,00 €	
Programador			20,00 €	12,00 €			
Formación		8	160,00 €	96,00 €			
Implementación		98	1.960,00 €	1.176,00 €			
Pruebas		78	1.560,00 €	936,00 €			
Despliegue		8	160,00 €	96,00 €			
	TOTAL	**192**	3.840,00 €	2.304,00 €	**Beneficio**	1.536,00 €	
							% beneficios:
	Total horas	290		**BENEFICIO TOTAL:**		2.940,00 €	**36,32 %**
				COSTE TOTAL:		3.750,00 €	IVA 21%
					PRESUPUESTO	**6.690,00 €**	1.404,90 €
				PRESUPUESTO + IVA		**8.094,90 €**	

Figura 9.5: Presupuesto del personal del proyecto.

9.0.1. Hardware y software

No se añaden costes por amortización de software ni hardware, ya que el personal utiliza sus propias tecnologías (estrategia de BYOT, *Bring Your Own Technology*).

Capítulo 10

Manual del programador

El objetivo de este manual es ayudar, a que un programador que pretenda ampliar o mejorar la aplicación pueda entender, en poco tiempo, la estructura del código.

A continuación se describen las clases de la aplicación agrupadas por paquetes.

10.0.2. Paquete kodama.main

Main.java

Se trata de la Activity principal. En caso de querer modificar la vista de pestañas, ésta es la clase que debería modificarse. Aquí se incluye también los métodos que facilitan la autenticación a través de Google Calendar. Se utiliza el protocolo OAuth 2.0.

10.0.3. Paquete kodama.receiver

ReceiverActivity.java

Activity que se corresponde con la pestaña de notificaciones. Se comunica con la clase CalendarManager para chequear si existen nuevos eventos. Para ello utiliza un hilo que está constantemente realizando el chequeo.

10.0.4. Paquete kodama.receiver.model

Notification.java

Esta clase contiene los atributos de una notificación y el método toString() correspondiente para mostrar estos datos. Si se desea añadir nuevos atributos a una *Notificación* ha de modificarse esta clase.

10.0.5. Paquete kodama.receiver.adapters

NotificationAdapter.java

Clase adaptadora que permite mostrar una Notificación en la lista de *Notificaciones*.

10.0.6. Paquete kodama.receiver.wrappers

NotificationWrapper.java

Clase envoltorio utilizada por Notification Adapter. Si la clase Notification.java se modifica, debería modificarse esta también.

10.0.7. Paquete kodama.sensor

SensorActivity.java

Activity que se corresponde con la pestaña del sensor. Posee un hilo para la captura de sonido. Si se desean capturar nuevos eventos habría que añadir nueva funcionalidad a esta clase, con nuevos hilos para capturar otro tipo de eventos.

SensorConstants.java

Posee las constantes del sensor. En caso de añadirse otros tipos de eventos, éste será el lugar correcto para añadir las nuevas constantes asociadas.

10.0.8. Paquete kodama.sensor.audio

AudioReader.java

Esta clase se encarga de la lectura continuada de micrófono indicando cuándo un bloque ha sido leído y está listo para procesarlo.

SoundManager.java

Es la encargada de recibir el bloque leído y encargarse de llamar al método que lo va a procesar para obtener los decibelios.

En caso de ampliarse la funcionalidad, ésta clase sería la encargada de llamar a la clase o clases nuevas encargada de procesar los datos del evento capturado.

10.0.9. Paquete kodama.sensor.operations

SignalPower.java

Esta clase es la encargada de realizar el cálculo de los decibelios, para ello, recibe el bloque de la clase SoundManager.

10.0.10. Paquete kodama.sensor.preferences

Configuration.java

PreferenceActivity que se corresponde con el fichero de *res/xml*, preferences.xml. Posee métodos get static para obtener los datos de configuración.

10.0.11. Paquete kodama.synchronization

CalendarManager.java

Clase que utiliza los métodos de la API de google Calendar. Esta clase utiliza los métodos create y retrieve, pero no utiliza los método update y delete. Por no contemplarse esta funcionalidad en esta aplicación, no se ha probado el correcto funcionamiento del método update con la integración de la API de Google Calendar en nuestra aplicación. Podría no funcionar, ya que el método delete no funcionaba tal cual estaba la clase CalendarClient.java que forma parte del código proporcionado por la API. Para conseguir que el borrado de eventos funcionara se ha tenido que modificar el método *executeEventDelete*.

Se añadió esta línea: *request.getHeaders().put(Ïf-Match;̈ *);̈*

Y se comentó esta otra: *//request.getHeaders().setIfMatch(((EventEntry) entry).etag)*;

Parece ser que no se lograba identificar correctamente el elemento a borrar, a pesar de indicarse en la URL de la request.

10.0.12. Paquete kodama.synchronization.calendar

CalendarClient.java

Clase que forma parte de la API de google, encargada de realizar las peticiones. Se añade en este manual porque fue objeto de modificación.

10.0.13. Paquete kodama.testing

CalendarAPITest.java

Clase para las pruebas unitarias del Calendario de Google. Si se contempla alguna prueba de update o delete sería llevada a cabo aquí.

NotificationTest.java

Clase que prueba el funcionamiento del módulo de notificaciones utilizando para ello la clase CalendarManager.

SoundsTest.java

Clase para las pruebas unitarias del procesamiento de bloques de sonido.

SensorSensibilityTest.java

Clase para las pruebas unitarias que nos han ayudado a fijar los umbrales de sensibilidad del sensor. Se han utilizado 3 niveles: bajo, medio, alto. Podría haberse utilizado un spinner, se contempló en su momento, pero la falta de conocimiento de Android y la falta de tiempo nos ha llevado a elegir la opción de los 3 niveles.

10.0.14. Paquete kodama.util

Constants.java

Clase que contiene las constantes usadas en la aplicación. Para el sensor, tenemos una clase de constantes específica, la clase SensorConstants.

10.0.15. Paquetes principales

En la siguiente figura se incluye una imagen con los paquetes que poseen el mayor peso en cuanto a funcionalidad.

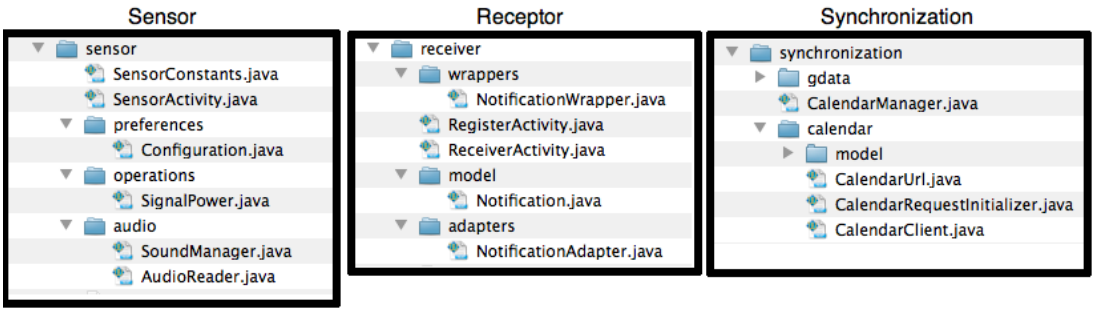

Figura 10.1: Paquetes principales de la aplicación.

10.0.16. Ficheros XML

En esta sección se listan los ficheros xml de Android. Se puede ver el contenido de estos ficheros en el apéndice que incluye el código de la aplicación.

10.0.17. Fichero AndroidManifest.xml

Paquete kodama.testing 10.1: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kodamaAwake.audio"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="8" />

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.USE_CREDENTIALS" >
    </uses-permission>
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />

    <instrumentation
        android:name="android.test.InstrumentationTestRunner"
        android:targetPackage="kodamaAwake.audio" >
    </instrumentation>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
```

```
        <uses-library android:name="android.test.runner" />

        <activity
             android:name="kodama.main.Main"
             android:label="@string/app_name" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.DEFAULT" />

                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.receiver.ReceiverActivity" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.sensor.SensorActivity" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.sensor.preferences.Configuration" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.PREFERENCE" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.receiver.RegisterActivity" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

En este se incluye la información para el flujo de ejecución de las Activities, los permisos necesarios para realizar ciertas tareas sobre el dispositivo y librerías como la de JUnit.

10.0.18. Layouts

Fichero main.xml

Pantalla principal.

Fichero notifications.xml

Pantalla donde se visualizan las notificaciones.

Fichero notificationrow.xml

Es el layout que conforma un ítem de la lista del layout notifications.xml.

Fichero recording.xml

Es la pantalla del sensor.

Fichero register.xml

Layout que muestra el registro de eventos.

Fichero preferences.xml

Layout de las preferencias del sensor.

10.0.19. Otros ficheros de recursos

Fichero arrays.xml

Contiene array con los niveles de sensibilidad a seleccionar en la pantalla preferences.xml.

Fichero strings.xml

Contiene las cadenas utilizadas en la aplicación. Se separan los textos que se muestran en las pantallas para facilitar los cambios y permitir la internacionalización.

10.0.20. Diagrama de navegación de pantallas

Este diagrama que se incluye a continuación refleja la comunicación de pantallas. Cada elemento representa un layout de la aplicación.

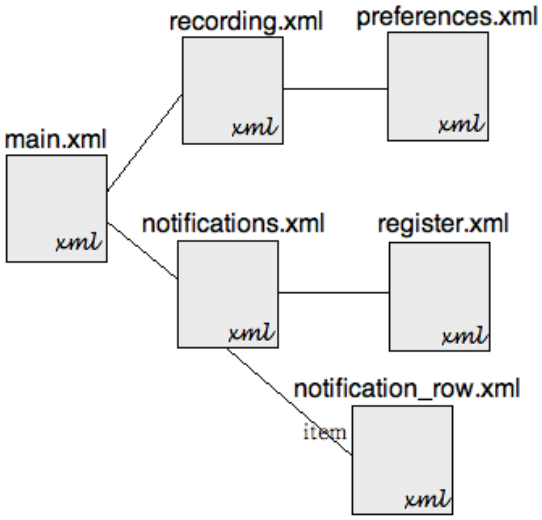

Figura 10.2: Navegación pantallas.

Capítulo 11

Manual de internacionalización

El objetivo de este manual es facilitar la tarea de internacionalización.

Los archivos de texto se encuentran en la carpeta values y se organizan de acuerdo al idioma, indicandolo al final del nombre de archivo. Se sigue este formato: values_-(identificador de idioma).

Podemos observar el ejemplo siguiente y veremos que tenemos una carpeta values, que será la que utiliza por defecto la aplicación si no encuentra el idioma. Para la configuración de idioma Español, utilizaría la carpeta values_es.

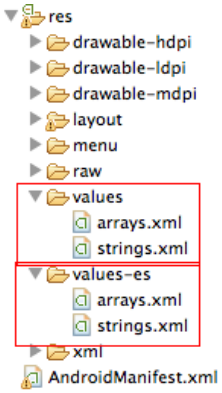

Figura 11.1: Ubicación de archivos xml de texto para la internacionalización

Los códigos de internacionalización se pueden consultar en `http://www.webtutoriales.com/articulos/codigos-de-paises-e-idiomas-i18n`.

A continuación vemos un extracto de strings.xml dentro de values_es. Cuando en la aplicación aparezca la referencia @string/name, si el idioma del dispositivo es español, cogerá el valor *Nombre*.

$< stringname = "name" > Nombre < /string >$
$< stringname = "place" > Localizacion < /string >$
$< stringname = "sensibility" > Sensibilidad < /string >$

Por ejemplo en *preferences.xml*
$< EditTextPreference$
$android : name = "@string/name"$
$android : defaultValue = "@string/not - defined"$
$android : key = "sensor - name"$
$android : summary = "@string/sensor - name - desc"$
$android : title = "@string/sensor - name" / >$

Las instruccione a seguir si se desea realizar la internacionalización para un nuevo idioma, por ejemplo, francés, serán:

- Crear una nueva carpeta values_fr
- Copiar el fichero strings.xml de values (el que se selecciona por defecto) en esta nueva carpeta
- Traducir las palabras o textos al idioma francés

$< stringname = "name" > Nom < /string >$
$< stringname = "place" > Endroit < /string >$
$< stringname = "sensibility" > Sensibilite < /string >$

Capítulo 12

Manual de instalación

En este capítulo hablaremos del procedimiento seguido para la instalación de la aplicación. Como se trata de una aplicación Android, nos centraremos en el proceso a seguir para generar el .apk.

12.1. Exportación de aplicación Android

En el eclipse, hacemos click derecho sobre el proyecto y seleccionamos **Export...** 12.1. Entonces se nos abrirá la ventana de la figura 12.2.

Seleccionamos el proyecto, y a continuación seguimos los pasos en orden.

Una vez generado el .apk, se podrá instalar éste, tras ser descargado en el dispositivo. Habrá que aceptar los permisos necesarios (Internet, etc).

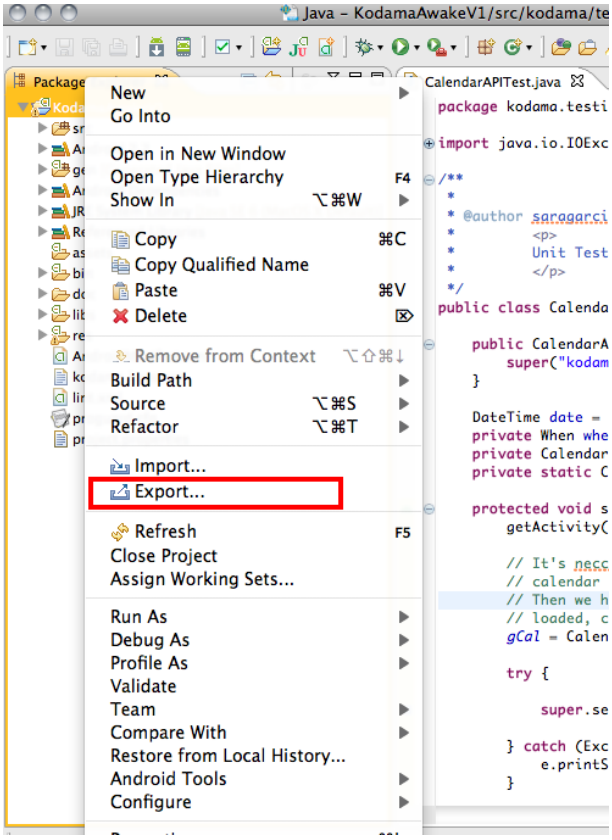

Figura 12.1: Menú del proyecto.

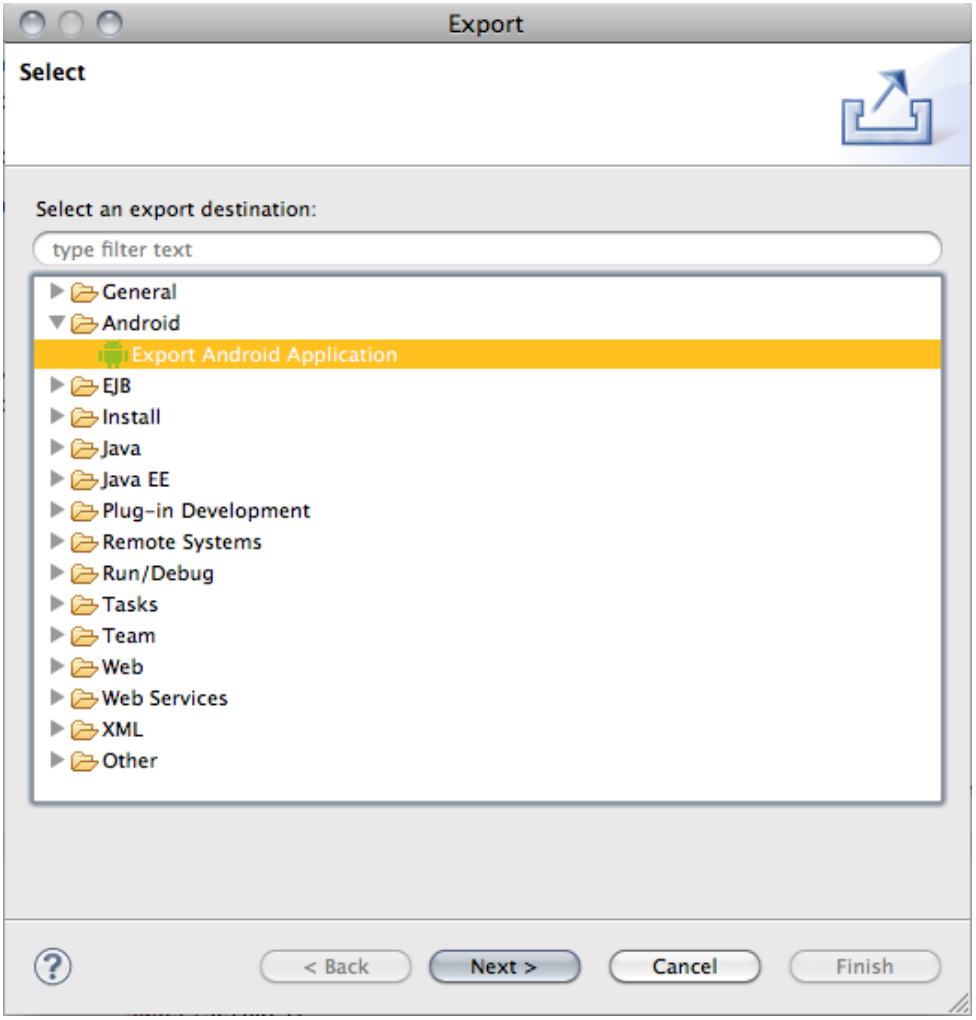

Figura 12.2: Ventana exportación

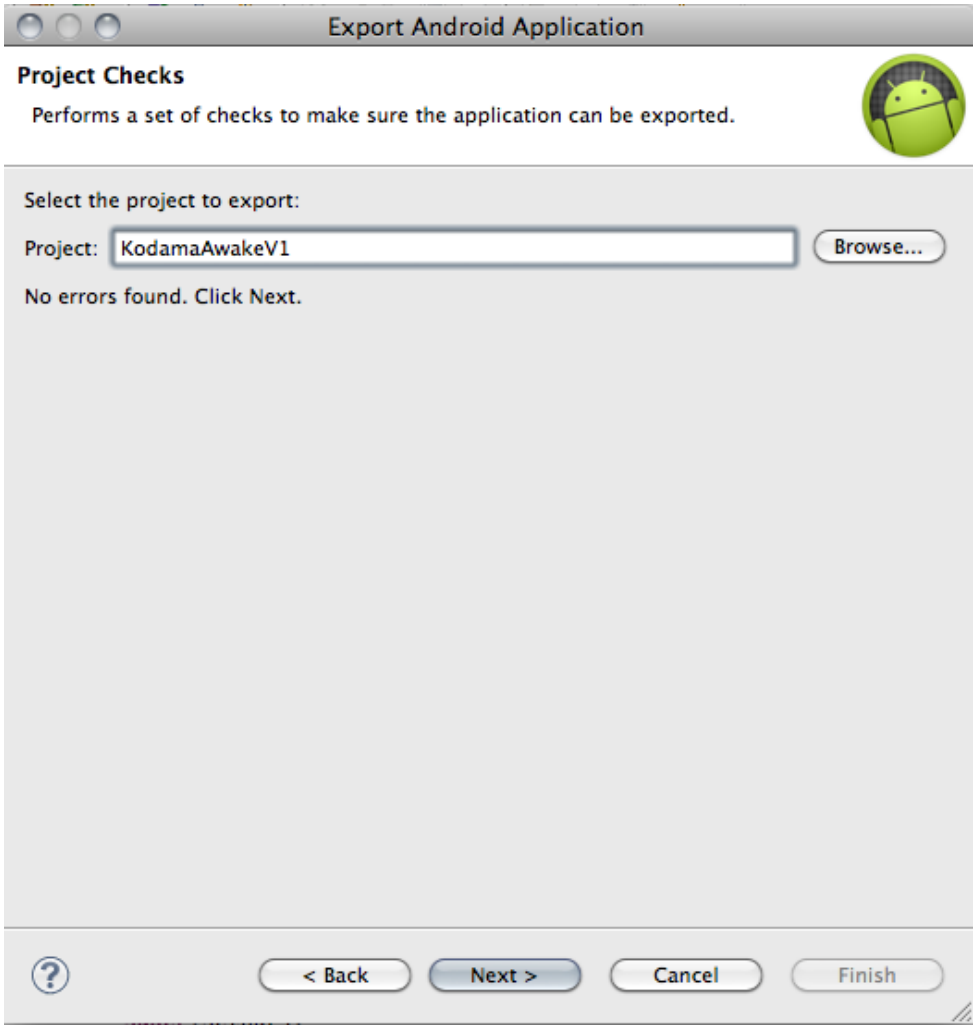

Figura 12.3: Paso 1 exportación

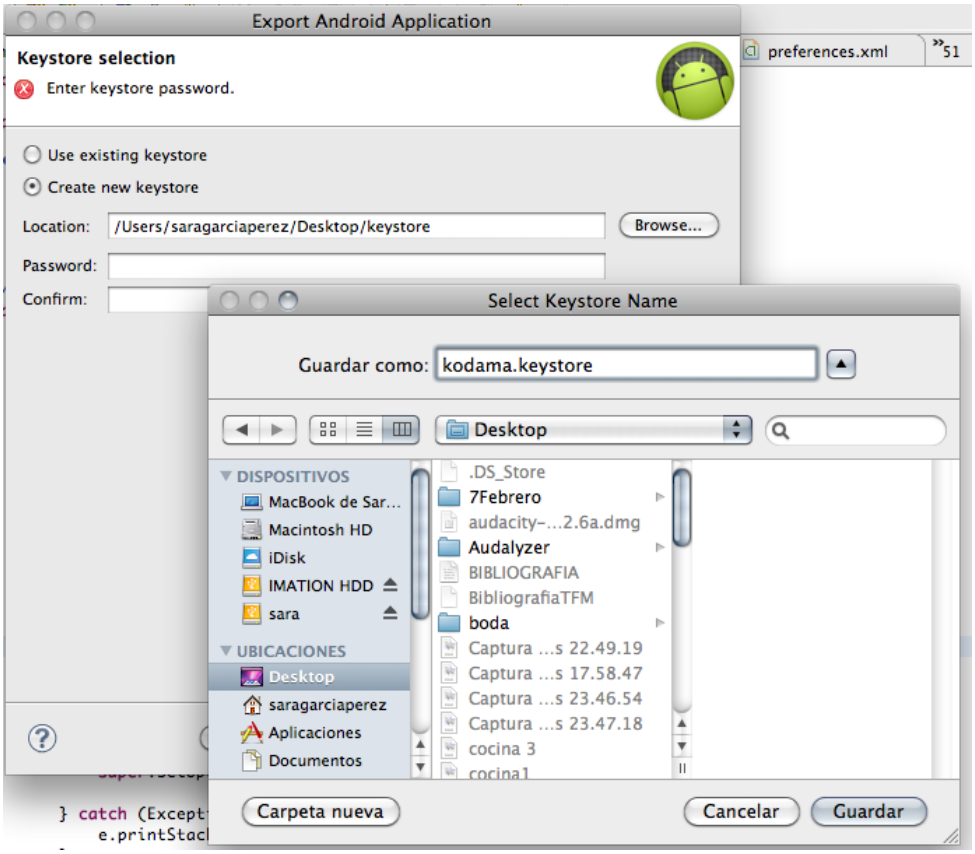

Figura 12.4: Paso 2 exportación

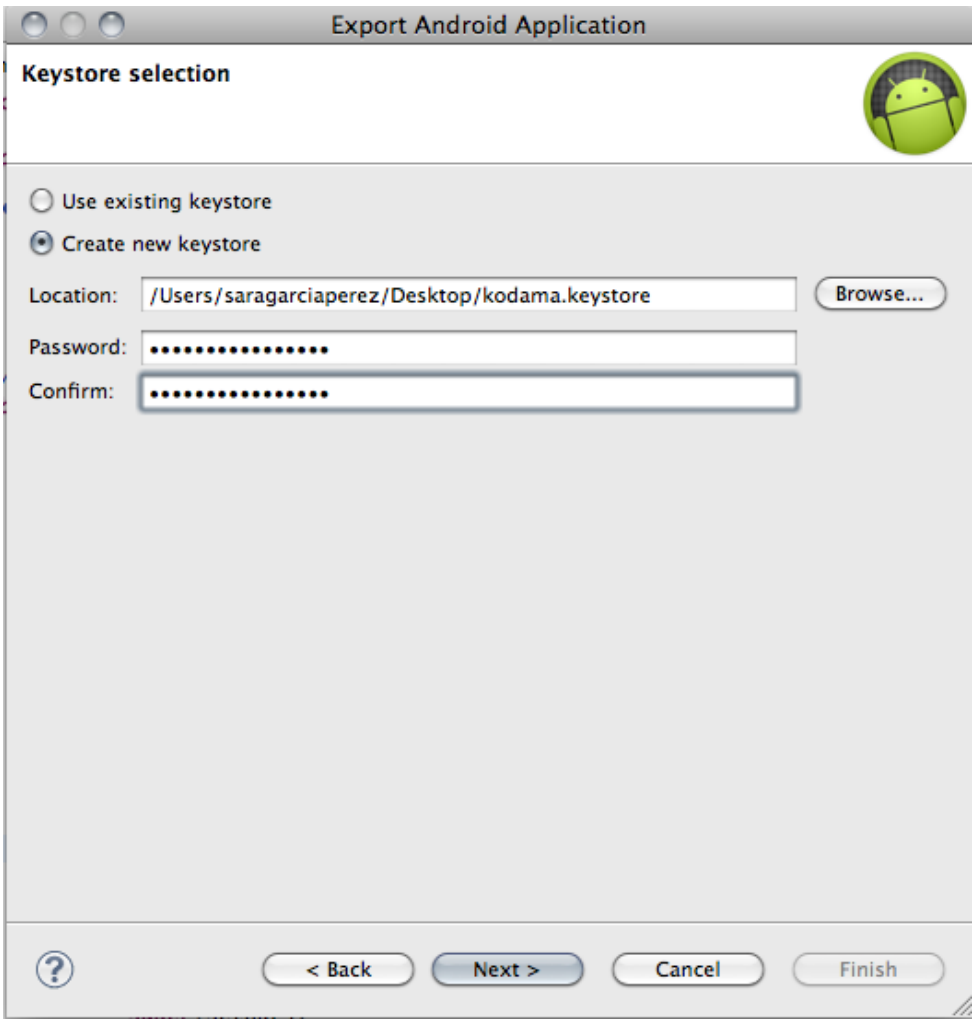

Figura 12.5: Paso 3 exportación

Figura 12.6: Paso 4 exportación

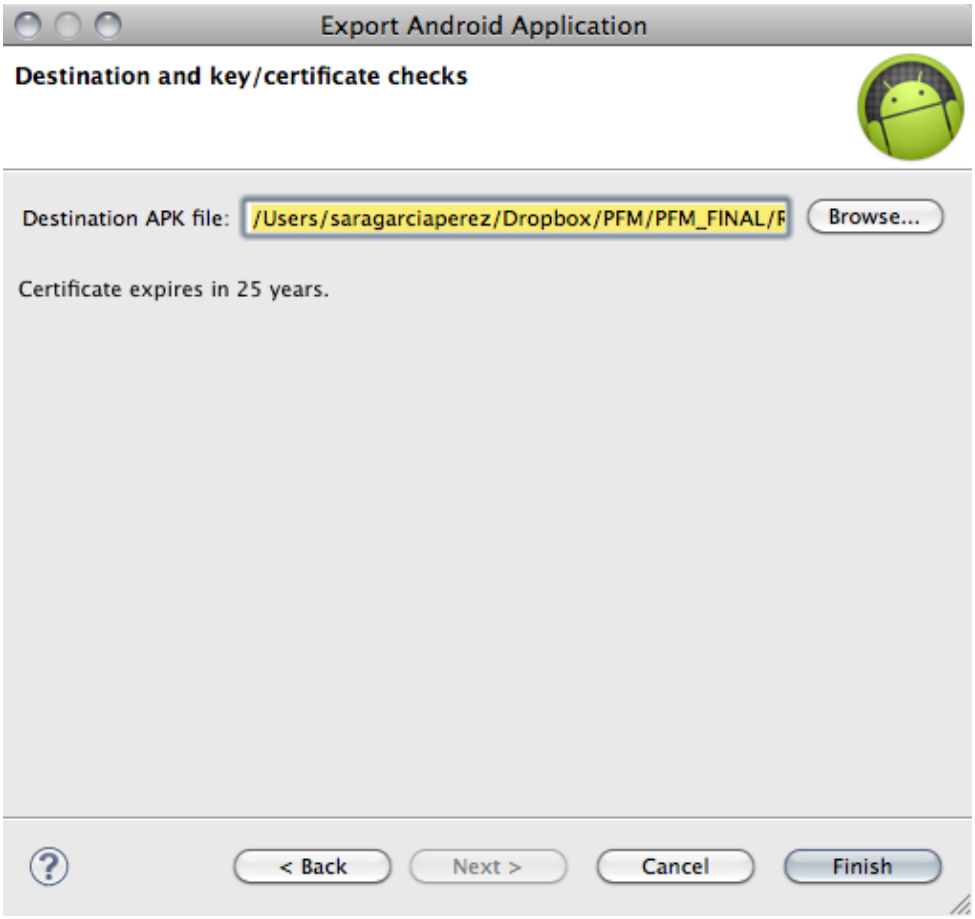

Figura 12.7: Paso 5 exportación

Capítulo 13

User manual

13.1. Sensor

When you access into the application you can see the main screen with the sensor mode in the front 13.1. You have the notifications mode in the rigth tab.

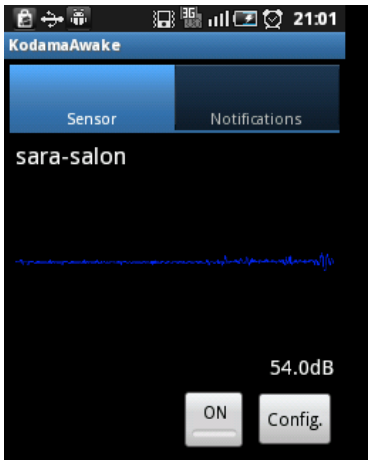

Figura 13.1: User access into the application.

The main function in the sensor is the activation of events capture **(Record button)**. The sensor also can be configured by clicking the **Config. button** in order to add a name, a location and set the sensibility (high, medium or low).

If you have several accounts of Google added 13.2, a list to select one will appear when accesing to the application, in order to create the KodamaAwake Google Calendar in the account selected.

13.2. Notifications

Go to the Notifications tab. To activate the reception of notifications click on the button **ON**. When a new notification arrives the notification data will appear on the

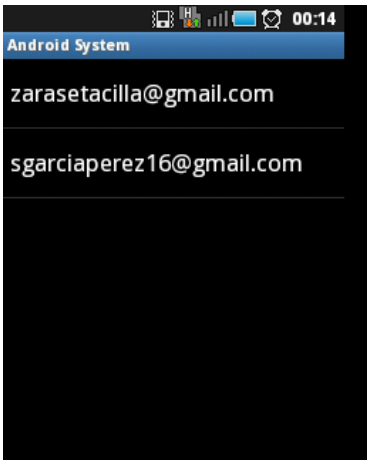

Figura 13.2: Access to the application with several Google accounts.

screen. Also it will be notified with sound and vibration. When clicking this notification, it will be removed from the screen, as checked. The notification remains in Google Calendar, though.

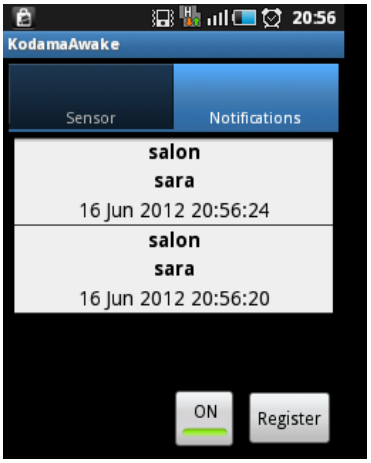

Figura 13.3: Notifications visualization.

Notifications will be displayed ordered, the most recent to the oldest.

13.2.1. Notifications Register

If you want to see the register for today, click *Register*. You will see a new explorer window that leads you to Google Calendar.

Firstly you have to enter your user and password and then you will be able to see your events for the current day or others by moving the arrows or change the month.

Figura 13.4: Register of notifications.

13.3. Application configuration

Take into account than you need to set the same Google Account in several devices in order to synchronize them. Therefore, you must create a new account if necessary, and then select it when you access to the application.

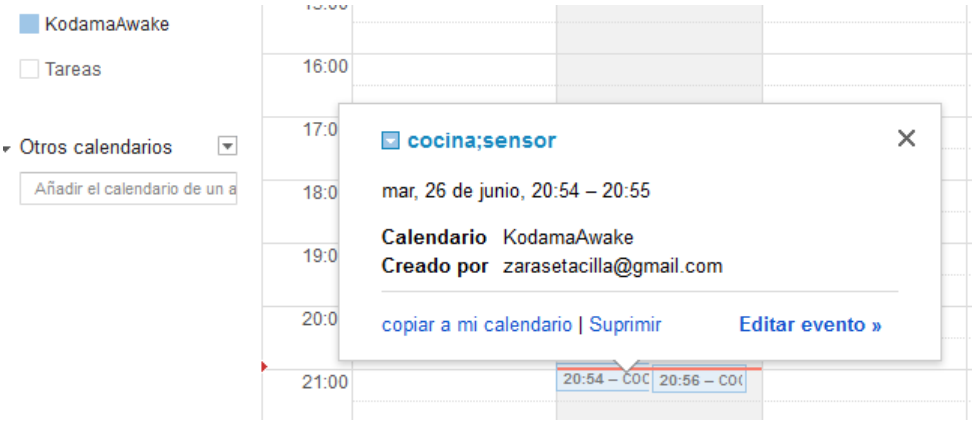

Figura 13.5: Google Calendar screenshot.

Capítulo 14

Conclusiones y ampliaciones

14.1. Conclusiones

La aplicación desarrollada, tal y como se había planteado en los objetivos, permite la interacción con el entorno de personas con deficiencias auditivas. Así mismo también podría servir como sistema de seguridad.

Ahora mismo no dispone de un sistema de identificación de sonidos concretos, pero podría integrarse fácilmente de existir, por ejemplo mediante el uso de un servicio Web que interpretara la naturaleza del bloque de sonido capturado.

La sincronización se lleva a cabo por medio de la API de Google Calendar, que nos ofrece un alto nivel de servicio(alta disponibilidad y seguridad).

14.2. Ampliaciones

Una vez concluído el proyecto quedan abiertas varias vías para seguir trabajando sobre el mismo, con el objetivo de ampliarlo y mejorarlo. A continuación se citan dos posibles futuras ampliaciones.

14.2.1. Captura de distintos eventos

Se podrían añadir a la aplicación la captura de diferentes eventos, no sólo sonido, como son movimiento o cambios de luz. Cuando se detectara un evento nuevo, se utilizaría la misma clase CalendarManager para la sincronización de la información.

14.2.2. Identificación de sonidos concretos

Como ya se ha comentado en las conclusiones, sería muy interesante poder disponer de un servicio Web de identificación de sonidos para integrarlo con nuestra aplicación.

Bibliografía

[1] Android. Android developers, 2012. http://developer.android.com/index.html.

[2] Everyday Developer. Android, multithreading in a ui environment, 2010. http://www.aviyehuda.com/2010/12/android-multithreading-in-a-ui-environment/.

[3] Caroline Golanski Camille Roux Brigitte Meillon FranÃ§ois Porlet, Michel Vacher. Design and evaluation of a smart home voice interface for the elderly: acceptability and objection aspects, 2011.

[4] Google. Google apis, 2012. http://code.google.com/p/google-api-java-client/wiki/APIs#Calendar_API/.

[5] Yusuke Hanba Susumu Kunifuji Hideaki Kanai, Toyohisa Nakada. A support system for context awareness in a group home using sound cues. 2008.

[6] Kaloer. Android preferences, 2010. http://www.kaloer.com/android-preferences/.

[7] mvinformatics@gmail.com. Sound detector, 2011. https://play.google.com/store/apps/details?id=dk.mvainformatics.android.sounddetector&hl=es.

[8] Ian Cameron Smith. Analyze the frequency and strength of sound in android. http://netscale.cse.nd.edu/twiki/pub/Main/Projects/Analyze_the_frequency_and_strength_of_sound_in_Android.pdf/.

[9] www.isisem.eu. Intelligent system for independent living and self-care of seniors with cognitive problems or mild dementia, 2009. [http://www.isisemd.eu].

Apéndice A

Código Fuente

1. Clases

1.1. Paquete kodama.receiver

ReceiverActivity.java

kodama.receiver A.1: ReceiverActivity.java

```
package kodama.receiver;

import java.io.IOException;
import java.util.List;
import java.util.Stack;

import kodama.main.Main;
import kodama.receiver.adapters.NotificationAdapter;
import kodama.receiver.model.Notification;
import kodama.sensor.audio.AudioReader;
import kodama.synchronization.CalendarManager;
import kodamaAwake.audio.R;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.os.Vibrator;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.View;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ListView;

import com.google.protobuf.ServiceException;

/**
 *
 * @author saragarciaperez
 *         <p>
 *          Receiver class, manages user interaction with the receiver
 *         </p>
 */
public class ReceiverActivity extends Activity {
        // ********************************************************************* //
        // Class Data.
        // ********************************************************************* //

        // Debugging tag.
        private static final String TAG = "EventReader";
        // ********************************************************************* //
        // Private Data.
        // ********************************************************************* //
        // Flag whether the thread should be running.
        private boolean running = false;

        // The thread, if any, which is currently reading. Null if not running.
        private Thread readerThread = null;
```

```
private CalendarManager gCal;
private Button btNotif;
private ListView listView;
private boolean mStartReading = true;

public Stack<Notification> notificationsStack = new Stack<Notification>();
private Vibrator vibrator;
private MediaPlayer mediaPlayer;
private Button btRegister;
private static String calendarUrl = "https://www.google.com/calendar/";

public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.notifications);

        btNotif = (Button) findViewById(R.id.toggleBtNotif);
        btRegister = (Button) findViewById(R.id.btRegister);
        listView = (ListView) findViewById(R.id.lvEvents);

        listView.setAdapter(new NotificationAdapter<Notification>(this,
                        notificationsStack));

        // Get instance of Vibrator from current Context
        vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
        mediaPlayer = MediaPlayer.create(ReceiverActivity.this, R.raw.alarm);

        btNotif.setOnClickListener(new View.OnClickListener() {

                public void onClick(View w) {
                        if (mStartReading) {

                                startReader();
                        } else {
                                stopReader();
                        }

                        mStartReading = !mStartReading;
                }
        });

        btRegister.setOnClickListener(new View.OnClickListener() {

                public void onClick(View w) {
                        Intent intent = null;
                        intent = new Intent(Intent.ACTION_VIEW, Uri.parse(calendarUrl));
                        startActivity(intent);
                }
        });

        /*
         * When the notification is touched is marked as read, and removed.
         */
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> adapter, View view, int pos,
                                long id) {

                        notificationsStack.remove(pos);
                        ((BaseAdapter) listView.getAdapter()).notifyDataSetChanged();

                }
        });
        this.gCal = Main.getGestorCalendar();
}

/**
 * Starts the event reader
 */
public void startReader() {
        Log.i(TAG, "Event Reader: Start Thread");

        synchronized (this) {
                running = true;
                readerThread = new Thread(new Runnable() {
                        @Override
                        public void run() {
                                readerRun();
                        }
                }, "Event Reader");
                readerThread.start();
        }
}

/**
 * Stops the event reader
 */
public void stopReader() {
        AudioReader.setIncomingNotification(false);
        running = false;

}

/**
 * Main loop of the event reader. This runs in our own thread.
```

```
         *
         * @throws ServiceException
         * @throws IOException
         */
        private void readerRun() {
                synchronized (this) {
                        Log.i(TAG, "Reader: Start Checking");

                        while (running) {
                                List<Notification> notif = gCal.checkEvents();
                                for (int i = 0; i < notif.size(); i++) {
                                        try {
                                                alert(notif.get(i));
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
                                }

                                if (!running) {
                                        break;
                                }

                                try {
                                        Thread.sleep(2000);
                                } catch (InterruptedException e) {
                                        e.printStackTrace();
                                }
                        }
                }

        }

        /**
         * Manage the notification alert
         *
         * @param notif
         * @throws InterruptedException
         */
        public void alert(Notification notif) throws InterruptedException {

                AudioReader.setIncomingNotification(true);

                notificationsStack.add(0, notif);

                this.runOnUiThread(new Runnable() {

                        public void run() {
                                Main.setCurrentTab(1);

                                ((BaseAdapter) listView.getAdapter()).notifyDataSetChanged();
                                vibrator.vibrate(500);
                                mediaPlayer.seekTo(1500);
                                mediaPlayer.start();
                        }
                });

                Thread.sleep(2000);

                AudioReader.setIncomingNotification(false);
        }

        public boolean onCreateOptionsMenu(Menu menu) {
                MenuInflater inflater = getMenuInflater();
                inflater.inflate(R.menu.menu, menu);
                return true;
        }

}
```

1.2. Paquete kodama.receiver.model

Notification.java

Paquete kodama.receiver.model A.2: Notification.java

```
package kodama.receiver.model;

import java.util.Calendar;
import java.util.Locale;

/**
 *
 * @author saragarciaperez
 *         <p>
 *          Object Notification with attributes
 *          </p>
```

```
 */
public class Notification {

        private String sensor_name;
        private String sensor_place;
        private String time;

        public String getSensor_name() {
                return sensor_name;
        }

        public void setSensor_name(String sensor_name) {
                this.sensor_name = sensor_name;
        }

        public String getSensor_place() {
                return sensor_place;
        }

        public void setSensor_place(String sensor_place) {
                this.sensor_place = sensor_place;
        }

        public String getTime() {
                return time;
        }

        public void setTime(String time) {
                this.time = getFormatedTime(time);
        }

        @Override
        public String toString() {
                return "Sensor name: " + sensor_name + "\nSensor location: "
                                + sensor_place + "\nTime: " + getFormatedTime(time);
        }

        /**
         *
         * @param time2
         *              2013-03-03T15:55:26.000Z
         * @return
         */
        private String getFormatedTime(String time2) {
                String date[] = time2.split("T");

                String day = date[0];

                String timeSplit [] = date[1].split("\\.");
                String timeFormated = timeSplit[0];

                return day + " " + timeFormated;
        }

        public static String getDay() {
                String date = Calendar.getInstance(Locale.getDefault()).getTime()
                                .toGMTString();
                date = date.substring(0, 11);
                date = date.replace(" ", "");
                return date;

        }

}
```

1.3. Paquete kodama.receiver.adapters

NotificationAdapter.java

Paquete kodama.receiver.adapters A.3: NotificationAdapter.java

```
package kodama.receiver.adapters;

import java.util.List;

import kodama.receiver.model.Notification;
import kodama.receiver.wrappers.NotificationWrapper;
import kodamaAwake.audio.R;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;

/**
 *
```

```
 * @author saragarciaperez
 *         <p>
 *          Notification Adapter to show the notification on the screen
 *         </p>
 * @param <T>
 */
public class NotificationAdapter<T extends Notification> extends
                ArrayAdapter<T> {

        private Context context;

        public NotificationAdapter(Context context, List<T> notifications) {
                super(context, R.layout.notification_row, notifications);
                this.context = context;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
                NotificationWrapper wrapper;
                Notification notif = getItem(position);

                if (convertView == null) {
                        LayoutInflater inflater = (LayoutInflater) context
                                        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                        convertView = inflater.inflate(R.layout.notification_row, null);
                        wrapper = new NotificationWrapper(convertView);
                        convertView.setTag(wrapper);
                } else {
                        wrapper = (NotificationWrapper) convertView.getTag();
                }

                wrapper.getSensorName().setText(notif.getSensor_name());
                wrapper.getLocation().setText(notif.getSensor_place());
                wrapper.getTime().setText(notif.getTime());

                return convertView;
        }
}
```

1.4. Paquete kodama.receiver.wrappers

NotificationWrapper.java

Paquete kodama.receiver.wrappers A.4: NotificationWrapper.java

```
package kodama.receiver.wrappers;

import kodamaAwake.audio.R;
import android.view.View;
import android.widget.TextView;

/**
 *
 * @author saragarciaperez
 *         <p>
 *         Wrapper for Notification
 *         </p>
 */
public class NotificationWrapper {

        private View base;
        private TextView sensorName;
        private TextView location;
        private TextView time;

        public NotificationWrapper(View base) {
                this.base = base;
        }

        public void setColors(int background, int foreground) {
                base.setBackgroundColor(background);
                getSensorName().setTextColor(foreground);
                getLocation().setTextColor(foreground);
                getTime().setTextColor(foreground);
        }

        public TextView getSensorName() {
                if (sensorName == null)
                        sensorName = (TextView) base.findViewById(R.id.sensor_name);
                return sensorName;
        }

        public TextView getLocation() {
                if (location == null)
                        location = (TextView) base.findViewById(R.id.sensor_location);
                return location;
        }

        public TextView getTime() {
                if (time == null)
                        time = (TextView) base.findViewById(R.id.time);
                return time;
        }

}
```

1.5. Paquete kodama.sensor

SensorActivity.java

Paquete kodama.sensor A.5: SensorActivity.java

```
package kodama.sensor;

import java.util.Calendar;
import java.util.GregorianCalendar;

import kodama.main.Main;
import kodama.sensor.audio.AudioReader;
import kodama.sensor.audio.SoundManager;
import kodama.sensor.preferences.Configuration;
import kodama.synchronization.CalendarManager;
import kodama.util.Constants;
import kodamaAwake.audio.R;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

/**
 *
 * @author saragarciaperez
 *         <p>
 *         Sensor class, manages user interaction with the sensor
 *         </p>
 */
public class SensorActivity extends Activity implements
                OnSharedPreferenceChangeListener {

        // Buffered audio data, and sequence number of the latest block.
        private byte[] audioData;

        private SoundManager soundManager = null;
        private CalendarManager gCal = null;

        private Button toggleBtPlay;
        private static TextView sensorDesc;
        private Button btConfiguration;
        private ImageView imageView;
        private boolean mStartRecording = true;
        private double dBTransformation = 0;
        private boolean isFirstBlock = true;

        private SharedPreferences preferences;
        private String sensor_name;
        private String sensor_location;

        public void onCreate(Bundle icicle) {
                super.onCreate(icicle);

                setContentView(R.layout.recording);
                soundManager = new SoundManager();

                this.gCal = Main.getGestorCalendar();

                sensorDesc = (TextView) findViewById(R.id.sensor_name_pref);

                preferences = PreferenceManager.getDefaultSharedPreferences(this);
                preferences.registerOnSharedPreferenceChangeListener(this);

                sensor_name = preferences.getString(Constants.SENSOR_NAME, "");
                sensor_location = preferences.getString(Constants.SENSOR_LOCATION, "");
                sensorDesc.setText(sensor_name + "-" + sensor_location);

                toggleBtPlay = (Button) findViewById(R.id.toggleBtPlay);

                toggleBtPlay.setOnClickListener(new View.OnClickListener() {

                        public void onClick(View w) {
                                if (mStartRecording) {
                                        // we try to avoid catching sound when sensor is being
                                        // activated
                                        isFirstBlock = true;
                                        measureStart();
                                } else {
                                        soundManager.stopReader();
                                }

                                mStartRecording = !mStartRecording;
                        }
                });

                btConfiguration = (Button) findViewById(R.id.btConfiguration);
                btConfiguration.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                Intent intent = new Intent(SensorActivity.this,
                                                Configuration.class);
                                startActivity(intent);
                        }
                });

        }

        /**
         * Detecting preferences changed to refresh the screen with new
         * configuration
         */
        @Override
        public void onSharedPreferenceChanged(SharedPreferences arg0, String arg1) {
                sensor_name = Configuration.getSensorName();
                sensor_location = Configuration.getSensorLocation();
                sensorDesc.setText(sensor_name + "-" + sensor_location);
        }

        /**
         * Starts the measurement of audio input
         */
```

```
public void measureStart() {

        SoundManager.getAudioReader().startReader(SensorConstants.SAMPLE_RATE,
                        SensorConstants.BLOCK_SIZE * SensorConstants.SAMPLE_DECIMATE,
                        new AudioReader.Listener() {

                                public final void onReadComplete(byte[] buffer) {
                                        receiveAudio(buffer);
                                }
                        });
}

/**
 * Receives an audio block to process later
 *
 * @param buffer
 */
private final void receiveAudio(byte[] buffer) {
        // Lock to protect updates to these local variables. See run().
        synchronized (this) {
                audioData = buffer;
                processAudio(audioData);
        }
}

// ********************************************************************** //
// Main Loop.
// ********************************************************************** //
/**
 * Processes audio, calculates dB and calls the CalendarManager to create a
 * new event if it takes place
 *
 * @param buffer
 */
private void processAudio(byte[] buffer) {

        double currentPower = soundManager.calculatePowerDb(buffer);

        Log.i("decibelios!!", "CurrentPower: " + currentPower + " dB");

        this.dBTransformation = Math.rint(currentPower * 100) / 100;

        this.runOnUiThread(new Runnable() {
                public void run() {
                        TextView dB = (TextView) findViewById(R.id.text_dB);
                        dB.setText(String.valueOf(dBTransformation + "dB"));
                }
        });

        if (currentPower > Configuration.getSensibility()) {
                try {
                        startAnimation();
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                if (!isFirstBlock) {
                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        gCal.createEvent(sensor_location, sensor_name);
                }
        }
        isFirstBlock = false;
}

/**
 * Generates the wave animation using four different images of a wave, that
 * is easier than try to use a gift
 *
 * @throws InterruptedException
 */
private void startAnimation() throws InterruptedException {

        this.runOnUiThread(new Runnable() {
                public void run() {
                        imageView = (ImageView) findViewById(R.id.wave);
                        imageView.setImageResource(R.drawable.onda1);
                }
        });
        Thread.sleep(500);

        this.runOnUiThread(new Runnable() {
                public void run() {
                        imageView = (ImageView) findViewById(R.id.wave);
                        imageView.setImageResource(R.drawable.onda2);
                }
        });

        Thread.sleep(500);

        this.runOnUiThread(new Runnable() {
                public void run() {
                        imageView = (ImageView) findViewById(R.id.wave);
                        imageView.setImageResource(R.drawable.onda3);
                }
```

```
                });
                Thread.sleep(500);

                this.runOnUiThread(new Runnable() {
                        public void run() {
                                imageView = (ImageView) findViewById(R.id.wave);
                                imageView.setImageResource(R.drawable.onda4);
                        }
                });

        }

}
```

SensorConstants.java

Paquete kodama.sensor A.6: SensorConstants.java

```
package kodama.sensor;

/**
 * <p>
 * Constants that needs the sensor
 * </p>
 *
 * @author saragarciaperez
 *
 */
public class SensorConstants {

        // The desired sampling rate for this analyser, in samples/sec.
        public static final int SAMPLE_RATE = 8000;

    // Audio input block size, in samples.
        public static final int BLOCK_SIZE = 4096;

        public static final int SAMPLE_DECIMATE = 1;

}
```

1.6. Paquete kodama.sensor.audio

AudioReader.java

Paquete kodama.sensor.audio A.7: AudioReader.java

```
package kodama.sensor.audio;

/**
 * org.hermit.android.io: Android utilities for accessing peripherals.
 *
 * These classes provide some basic utilities for accessing the audio
 * interface, at present.
 *
 * <br>Copyright 2009 Ian Cameron Smith
 *
 * <p>This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2
 * as published by the Free Software Foundation (see COPYING).
 *
 * <p>This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 */

import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.util.Log;

/**
 * A class which reads audio input from the mic in a background thread and
 * passes it to the caller when ready.
 *
 * <p>
 * To use this class, your application must have permission RECORD_AUDIO.
 */
public class AudioReader {
```

```
// ******************************************************************** //
// Public Classes.
// ******************************************************************** //

/**
 * Listener for audio reads.
 */
public static abstract class Listener {
        /**
         * Audio read error code: no error.
         */
        public static final int ERR_OK = 0;

        /**
         * Audio read error code: the audio reader failed to initialise.
         */
        public static final int ERR_INIT_FAILED = 1;

        /**
         * Audio read error code: an audio read failed.
         */
        public static final int ERR_READ_FAILED = 2;

        /**
         * An audio read has completed.
         *
         * @param buffer
         *            Buffer containing the data.
         */
        public abstract void onReadComplete(byte[] buffer);

}

// ******************************************************************** //
// Run Control.
// ******************************************************************** //

/**
 * Start this reader.
 *
 * @param rate
 *            The audio sampling rate, in samples / sec.
 * @param block
 *            Number of samples of input to read at a time. This is
 *            different from the system audio buffer size.
 * @param listener
 *            Listener to be notified on each completed read.
 */
public void startReader(int rate, int block, Listener listener) {
        Log.i(TAG, "Reader: Start Thread");
        synchronized (this) {
                // Calculate the required I/O buffer size.
                int audioBuf = AudioRecord.getMinBufferSize(rate,
                                AudioFormat.CHANNEL_IN_MONO,
                                AudioFormat.ENCODING_PCM_16BIT) * 2;

                // Set up the audio input.
                audioInput = new AudioRecord(MediaRecorder.AudioSource.MIC, rate,
                                AudioFormat.CHANNEL_IN_MONO,
                                AudioFormat.ENCODING_PCM_16BIT, audioBuf);
                inputBlockSize = block;
                // sleepTime = (long) (1000f / ((float) rate / (float) block));
                inputBuffer = new byte[2][inputBlockSize];
                inputBufferWhich = 0;
                inputBufferIndex = 0;
                inputListener = listener;
                running = true;
                readerThread = new Thread(new Runnable() {
                        @Override
                        public void run() {
                                readerRun();
                        }
                }, "Audio Reader");
                readerThread.start();
        }
}

/**
 * Stop this reader.
 */
public void stopReader() {
        Log.i(TAG, "Reader: Signal Stop");
        synchronized (this) {
                running = false;
        }
        try {
                if (readerThread != null)
                        readerThread.join();
        } catch (InterruptedException e) {
                e.printStackTrace();
        }
        readerThread = null;

        // Kill the audio input.
```

```
        synchronized (this) {
                if (audioInput != null) {
                        audioInput.release();
                        audioInput = null;
                }
        }

        Log.i(TAG, "Reader: Thread Stopped");
}

// ********************************************************************** //
// Main Loop.
// ********************************************************************** //

/**
 * Main loop of the audio reader. This runs in our own thread.
 */
private void readerRun() {
        byte[] buffer;
        int index, readSize;

        int timeout = 200;
        try {
                while (timeout > 0
                                && audioInput.getState() != AudioRecord.STATE_INITIALIZED) {
                        Thread.sleep(50);
                        timeout -= 50;
                }
        } catch (InterruptedException e) {
        }

        if (audioInput.getState() != AudioRecord.STATE_INITIALIZED) {
                Log.e(TAG, "Audio reader failed to initialize");
                running = false;
                return;
        }

        try {
                Log.i(TAG, "Reader: Start Recording");
                audioInput.startRecording();
                while (running) {
                        long stime = System.currentTimeMillis();

                        if (!running)
                                break;

                        while (inConmingNotification) {
                                try {
                                        Thread.sleep(1000);
                                } catch (InterruptedException e1) {
                                        e1.printStackTrace();
                                }
                        }

                        readSize = inputBlockSize;
                        int space = inputBlockSize - inputBufferIndex;
                        if (readSize > space)
                                readSize = space;
                        buffer = inputBuffer[inputBufferWhich];
                        index = inputBufferIndex;

                        synchronized (buffer) {
                                int nread = audioInput.read(buffer, index, readSize);

                                boolean done = false;
                                if (!running)
                                        break;

                                if (nread < 0) {
                                        Log.e(TAG, "Audio read failed: error " + nread);
                                        running = false;
                                        break;
                                }
                                int end = inputBufferIndex + nread;
                                if (end >= inputBlockSize) {
                                        inputBufferWhich = (inputBufferWhich + 1) % 2;
                                        inputBufferIndex = 0;
                                        done = true;
                                } else
                                        inputBufferIndex = end;

                                if (done) {
                                        readDone(buffer);

                                        // Because our block size is way smaller than the audio
                                        // buffer, we get blocks in bursts, which messes up
                                        // the audio analyzer. We don't want to be forced to
                                        // wait until the analysis is done, because if
                                        // the analysis is slow, lag will build up. Instead
                                        // wait, but with a timeout which lets us keep the
                                        // input serviced.
                                        long etime = System.currentTimeMillis();
                                        long sleep = sleepTime - (etime - stime);
                                        if (sleep < 5)
```

```
                                                sleep = 5;
                                        try {
                                                buffer.wait(sleep);
                                        } catch (InterruptedException e) {
                                        }
                                }
                        }
                }
        } finally {
                Log.i(TAG, "Reader: Stop Recording");
                if (audioInput.getState() == AudioRecord.RECORDSTATE_RECORDING)
                        audioInput.stop();
        }
}

/**
 * Notify the client that a read has completed.
 *
 * @param buffer
 *             Buffer containing the data.
 */
private void readDone(byte[] buffer) {
        inputListener.onReadComplete(buffer);
}

/**
 * To let the sensor stops for a moment if notification is going to be
 * showed
 *
 * @param isIncomingNotif
 */
public static void setIncomingNotification(boolean isIncomingNotif) {
        inConmingNotification = isIncomingNotif;
}

// ********************************************************************** //
// Class Data.
// ********************************************************************** //

// Debugging tag.
private static final String TAG = "Audio Input";

// Our audio input device.
private AudioRecord audioInput;

// Our audio input buffer, and the index of the next item to go in.
private byte[][] inputBuffer = null;
private int inputBufferWhich = 0;
private int inputBufferIndex = 0;

// Size of the block to read each time.
private int inputBlockSize = 0;

// Time in ms to sleep between blocks, to meter the supply rate.
private long sleepTime = 1000;

// Listener for input.
private Listener inputListener = null;

// Flag whether the thread should be running.
private static boolean running = false;

// The thread, if any, which is currently reading. Null if not running.
protected Thread readerThread = null;
private static boolean inConmingNotification = false;

}
```

SoundManager.java

Paquete kodama.sensor.audio A.8: SoundManager.java

```
package kodama.sensor.audio;

import kodama.sensor.operations.SignalPower;

/**
 *
 * @author saragarciaperez
 *         <p>
 *         Class in charge of manage the audio reader and calculate dB
 *         </p>
 */
public class SoundManager {

        // Our audio input device.
        private static AudioReader audioReader = null;

        public SoundManager() {
                audioReader = new AudioReader();
        }

        //Calculate DeciBels dBFS (we are using "Decibels Full Scale") not dB SPL ÒSound Pressure
            LevelÓ
        public double calculatePowerDb(byte[] buffer) {
                final int len = buffer.length;

                double currentPower = SignalPower.calculatePowerDb(buffer, 0, len);


                return currentPower;
        }

        public void stopReader() {
                audioReader.stopReader();
        }

        public static AudioReader getAudioReader() {
                return audioReader;
        }

        public void sleep() throws InterruptedException {
                Thread.sleep(3000);
        }

}
```

1.7. Paquete kodama.sensor.operations

SignalPower.java

Paquete kodama.sensor.operations A.9: SignalPower.java

```
package kodama.sensor.operations;

/**
 * A power metering algorithm.
 */
public final class SignalPower {

        /**
         * Calculate the power of the given input signal.
         *
         * @param sdata
         *            Buffer containing the input samples to process.
         * @param off
         *            Offset in sdata of the data of interest.
         * @param samples
         *            Number of data samples to process.
         * @return The calculated power in dB relative to the maximum input level;
         *         hence 0dB represents maximum power, and minimum power is about
         *         -95dB. Particular cases of interest:
         *         <ul>
         *         <li>A non-clipping full-range sine wave input is about -2.41dB.
         *         <li>Saturated input (heavily clipped) approaches 0dB.
         *         <li>A low-frequency fully saturated input can get above 0dB, but
         *         this would be pretty artificial.
         *         <li>A really tiny signal, which only occasionally deviates from
         *         zero, can get below -100dB.
         *         <li>A completely zero input will produce an output of -Infinity.
         *         </ul>
         *         <b>You must be prepared to handle this infinite result and
```

```
         *          results greater than zero,</b> although clipping them off would
         *          be quite acceptable in most cases.
         */
        public final static double calculatePowerDb(byte[] sdata, int off,
                        int samples) {
                // Calculate the sum of the values, and the sum of the squared values.
                // We need longs to avoid running out of bits.
                double sum = 0;
                double sqsum = 0;
                for (int i = 0; i < samples; i++) {
                        final long v = sdata[off + i];
                        sum += v;
                        sqsum += v * v;
                }

                double power = (sqsum - sum * sum / samples) / samples;

                // Scale to the range 0 - 1.
                power /= MAX_16_BIT * MAX_16_BIT;

                // Convert to dB, with 0 being max power. Add a fudge factor to make
                // a "real" fully saturated input come to 0 dB.
                return Math.log10(power) * 10f + FUDGE;

        }

        // ********************************************************************** //
        // Constants.
        // ********************************************************************** //

        // Maximum signal amplitude for 16-bit data.
        private static final float MAX_16_BIT = 32768;

        // This fudge factor is added to the output to make a realistically
        // fully-saturated signal come to 0dB. Without it, the signal would
        // have to be solid samples of -32768 to read zero, which is not
        // realistic. This really is a fudge, because the best value depends
        // on the input frequency and sampling rate. We optimise here for
        // a 1kHz signal at 16,000 samples/sec.
        private static final float FUDGE = 0.6f;

}
```

1.8. Paquete kodama.sensor.preferences

Configuration.java

Paquete kodama.sensor.preferences A.10: Configuration.java

```
package kodama.sensor.preferences;

import kodama.util.Constants;
import kodamaAwake.audio.R;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

/**
 *
 * @author saragarciaperez
 *         <p>
 *         Sensor configuration class, extends PreferenceActivity, uses layout
 *         R.xml.preferences
 *         </p>
 */
public class Configuration extends PreferenceActivity {
        private static SharedPreferences preferences;
        private static double defaultSensibility = -55;

        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                addPreferencesFromResource(R.xml.preferences);
                preferences = PreferenceManager.getDefaultSharedPreferences(this);
        }

        public static double getSensibility() {

                double value = 0;
                if (preferences != null) {
                        if (preferences.getString(Constants.SENSIBILITY, "").equals(""))
                                return defaultSensibility;
                        value = Double.parseDouble(preferences.getString(
                                        Constants.SENSIBILITY, ""));
                }
```

```
                if (value == 0)
                        value = defaultSensibility;
                return value;
        }

        public static String getSensorName() {
                String sensorName = preferences.getString(Constants.SENSOR_NAME, "");

                if (sensorName == null)
                        return "";

                return sensorName;
        }

        public static String getSensorLocation() {
                String sensorLocation = preferences.getString(
                                Constants.SENSOR_LOCATION, "");
                if (sensorLocation == null)
                        return "";
                return sensorLocation;
        }

}
```

1.9. Paquete kodama.synchronization

CalendarManager.java

Paquete kodama.synchronization A.11: CalendarManager.java

```
package kodama.synchronization;

import java.io.IOException;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.Locale;
import java.util.TimeZone;

import kodama.receiver.model.Notification;
import kodama.synchronization.calendar.CalendarClient;
import kodama.synchronization.calendar.CalendarUrl;
import kodama.synchronization.calendar.model.CalendarEntry;
import kodama.synchronization.calendar.model.CalendarFeed;
import kodama.synchronization.calendar.model.EventEntry;
import kodama.synchronization.calendar.model.When;

import com.google.api.client.util.DateTime;

/**
 *
 * @author saragarciaperez
 *         <p>
 *
 *         Singleton class that uses Google API calendar to store the
 *         notifications and to allow the synchronization
 *         </p>
 */
public class CalendarManager {

        private static CalendarManager instance = null;
        private CalendarClient client = null;

        private CalendarEntry calendarEntry;
        private static DateTime now = null;

        private CalendarManager() {
                Locale.setDefault(Locale.getDefault());
                setDateTimeNow();
        }

        // lazy singleton
        public static CalendarManager getInstance() {
                if (instance == null) {
                        synchronized (CalendarManager.class) {
                                if (instance == null) {
                                        instance = new CalendarManager();
                                }
                        }
                }
                return instance;
        }

        public CalendarClient getClient() {
                return client;
        }
```

```
public void setClient(CalendarClient client) {
        this.client = client;
}

/**
 * Sets the calendar to use, if not created yet, create a new one
 */
public void setCalendar() {
        try {
                CalendarEntry cal = null;

                cal = new CalendarEntry();
                cal.title = "KodamaAwake";
                calendarEntry = getCalendar("KodamaAwake");
                if (calendarEntry != null) {
                        client.eventFeed()
                                        .list()
                                        .execute(
                                                        new CalendarUrl(calendarEntry
                                                                        .getEventFeedLink()));
                }
                if (calendarEntry == null)
                        calendarEntry = client.calendarFeed().insert()
                                        .execute(forOwnCalendarsFeed(), cal);

        } catch (IOException e) {
                e.printStackTrace();
        }
}

/**
 * Looks for the current calendar in use
 *
 * @param string
 * @return
 * @throws IOException
 */
public CalendarEntry getCalendar(String cal) throws IOException {
        CalendarFeed calFeed = client.calendarFeed().list()
                        .execute(forOwnCalendarsFeed());

        for (int i = 0; i < calFeed.calendars.size(); i++) {
                if (calFeed.calendars.get(i).title.equals(cal)) {
                        return calFeed.calendars.get(i);
                }
        }
        return null;
}

/**
 * Creates a new event
 *
 * @param location
 * @param name
 */
public void createEvent(String location, String name) {
        EventEntry eventEntry = new EventEntry();
        eventEntry.title = location + ";" + name;

        When when = new When();

        GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

        when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

        cal.add(GregorianCalendar.MINUTE, 1);

        when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

        eventEntry.when = when;
        try {
                client.eventFeed()
                                .insert()
                                .execute(new CalendarUrl(calendarEntry.getEventFeedLink()),
                                                eventEntry);
        } catch (IOException e) {
                e.printStackTrace();
        }

}

/**
 * Checks if there are new events
 *
 * @return
 */
public List<Notification> checkEvents() {

        EventEntry event = null;

        List<EventEntry> events = getEvents();

        List<Notification> notifications = new ArrayList<Notification>();
```

```
                for (int i = 0; i < events.size(); i++) {
                        event = events.get(i);

                        if (event.when.startTime.getValue() > now.getValue()) {
                                notifications.add(createNotification(event.title,
                                                event.when.startTime.toString()));
                        }

                }

                // we register last time of reading
                setDateTimeNow();
                return notifications;
        }

        /**
         * Creates a new notification
         *
         * @param title
         * @param time
         * @return
         */
        public Notification createNotification(String title, String time) {
                Notification notif = new Notification();
                try {
                        if (title.equals(";")) {
                                notif.setSensor_name(" ");
                                notif.setSensor_place(" ");
                        } else {
                                String[] tokens = title.split(";");
                                notif.setSensor_place(tokens[0]);
                                notif.setSensor_name(tokens[1]);
                        }

                        notif.setTime(time);
                } catch (Exception e) {
                        e.getMessage();
                }
                return notif;
        }

        public void setDateTimeNow() {
                GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());
                now = new DateTime(cal.getTime(), TimeZone.getDefault());
        }

        /**
         * Calendar root URL
         *
         * @return
         */
        private static CalendarUrl forRoot() {
                return new CalendarUrl(CalendarUrl.ROOT_URL);
        }

        /**
         * Calendar Metafeed URL
         *
         * @return
         */
        private static CalendarUrl forCalendarMetafeed() {
                CalendarUrl result = forRoot();
                result.getPathParts().add("default");
                return result;
        }

        /**
         * Calendar Feed
         *
         * @return
         */
        private static CalendarUrl forOwnCalendarsFeed() {
                CalendarUrl result = forCalendarMetafeed();
                result.getPathParts().add("owncalendars");
                result.getPathParts().add("full");
                return result;
        }

        public List<EventEntry> getEvents() {

                List<EventEntry> events = null;
                try {
                        events = client.eventFeed().list()
                                        .execute(new CalendarUrl(calendarEntry.getEventFeedLink())).
                                            events;
                } catch (IOException e) {
                        e.printStackTrace();
                }
                return events;
        }

}
```

1.10. Paquete kodama.synchronization.calendar

CalendarClient.java

Paquete kodama.synchronization A.12: CalendarClient.java

```
package kodama.synchronization.calendar;

/*
 * Copyright (c) 2011 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
 * in compliance with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software distributed under the License
 * is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
 * or implied. See the License for the specific language governing permissions and limitations under
 * the License.
 */

import java.io.IOException;

import kodama.synchronization.calendar.model.CalendarEntry;
import kodama.synchronization.calendar.model.CalendarFeed;
import kodama.synchronization.calendar.model.Entry;
import kodama.synchronization.calendar.model.EventEntry;
import kodama.synchronization.calendar.model.EventFeed;
import kodama.synchronization.calendar.model.Feed;
import kodama.synchronization.gdata.xml.GDataXmlClient;

import com.google.api.client.http.GenericUrl;
import com.google.api.client.http.HttpRequest;
import com.google.api.client.http.HttpRequestFactory;
import com.google.api.client.xml.XmlNamespaceDictionary;


/**
 * Client for Google Calendar Data API.
 *
 * @author Yaniv Inbar
 */
public class CalendarClient extends GDataXmlClient {

        /**
         * Modified by saragarciaperez to allow deletion
         * @param entry
         * @throws IOException
         */
        public void executeEventDelete(EventEntry entry) throws IOException {
                HttpRequest request = this.getRequestFactory().buildDeleteRequest(
                                new GenericUrl(entry.getEditLink()));
                //Sara: I had to change this line in order to get the delete function working
                request.getHeaders().put("If-Match", "*");
                //request.getHeaders().setIfMatch(((EventEntry) entry).etag);
                request.execute().ignore();
        }

        static final XmlNamespaceDictionary DICTIONARY = new XmlNamespaceDictionary()
                        .set("", "http://www.w3.org/2005/Atom")
                        .set("batch", "http://schemas.google.com/gdata/batch")
                        .set("gd", "http://schemas.google.com/g/2005");

        public CalendarClient(HttpRequestFactory requestFactory) {
                super("2", requestFactory, DICTIONARY);
        }

        public void executeDelete(Entry entry) throws IOException {
                CalendarUrl url = new CalendarUrl(entry.getEditLink());
                super.executeDelete(url, null);
        }

        <T> T executeGet(CalendarUrl url, Class<T> parseAsType) throws IOException {
                return super.executeGet(url, parseAsType);
        }

        public <T extends Entry> T executePatchRelativeToOriginal(T original,
                        T updated) throws IOException {
                CalendarUrl url = new CalendarUrl(updated.getEditLink());
                return super.executePatchRelativeToOriginal(url, original, updated,
                                null);
        }

        <T> T executePost(CalendarUrl url, T content) throws IOException {
                return super.executePost(url, content instanceof Feed, content);
        }

        public EventCollection eventFeed() {
                return new EventCollection();
```

```
        }

        /** Event collection. */
        public class EventCollection {

                public ListRequest list() {
                        return new ListRequest();
                }

                /** List request. */
                public class ListRequest {

                        public EventFeed execute(CalendarUrl url) throws IOException {
                                return executeGet(url, EventFeed.class);
                        }
                }

                public BatchRequest batch() {
                        return new BatchRequest();
                }

                /** Batch request. */
                public class BatchRequest {
                        public EventFeed execute(EventFeed eventFeed, CalendarUrl batchUrl)
                                        throws IOException {
                                return executePost(batchUrl, eventFeed);
                        }
                }

                public InsertRequest insert() {
                        return new InsertRequest();
                }

                /** Insert request. */
                public class InsertRequest {
                        public EventEntry execute(CalendarUrl url, EventEntry entry)
                                        throws IOException {
                                return executePost(url, entry);
                        }
                }

                public DeleteRequest delete() {
                        return new DeleteRequest();
                }

                /** Delete request. */
                public class DeleteRequest {
                        public void execute(EventEntry entry) throws IOException {
                                executeDelete(entry);
                        }
                }
        }

        public CalendarCollection calendarFeed() {
                return new CalendarCollection();
        }

        /** Calendar collection. */
        public class CalendarCollection {

                public ListRequest list() {
                        return new ListRequest();
                }

                /** List request. */
                public class ListRequest {

                        public CalendarFeed execute(CalendarUrl url) throws IOException {
                                return executeGet(url, CalendarFeed.class);
                        }
                }

                public InsertRequest insert() {
                        return new InsertRequest();
                }

                /** Insert request. */
                public class InsertRequest {
                        public CalendarEntry execute(CalendarUrl url, CalendarEntry entry)
                                        throws IOException {
                                return executePost(url, entry);
                        }
                }

                public PatchRequest patch() {
                        return new PatchRequest();
                }

                /** Patch request. */
                public class PatchRequest {
                        public CalendarEntry execute(CalendarEntry original,
                                        CalendarEntry updated) throws IOException {
                                return executePatchRelativeToOriginal(original, updated);
                        }
```

```
                }

                public DeleteRequest delete() {
                        return new DeleteRequest();
                }

                /** Delete request. */
                public class DeleteRequest {
                        public void execute(CalendarEntry entry) throws IOException {
                                executeDelete(entry);
                        }
                }
        }
}
```

1.11. Paquete kodama.testing

SoundsTest.java

Paquete kodama.synchronization A.13: SoundsTest.java

```
package kodama.testing;

import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.UnsupportedAudioFileException;

import kodama.sensor.audio.SoundManager;

/**
 *
 * @author saragarciaperez
 *
 *          Testing sound input
 */
public class SoundsTest {
        private SoundManager soundManager;
        private ByteArrayOutputStream out = new ByteArrayOutputStream();
        private AudioInputStream in;

        public void setTestEnvironment() {
                soundManager = new SoundManager();
        }

        // Testing different dB input
        public void testSounds(String file) throws FileNotFoundException,
                        UnsupportedAudioFileException, IOException {

                in = AudioSystem.getAudioInputStream(new FileInputStream(file));

                int read;
                byte[] buff = new byte[1024];
                while ((read = in.read(buff)) > 0) {
                        out.write(buff, 0, read);
                }
                out.flush();

                byte[] audioBytes = out.toByteArray();

                double decibelios = soundManager.calculatePowerDb(audioBytes);

                System.out.println("Decibelios: " + decibelios);
                out.reset();
        }

        // The sound files were extracted from www.audiotest.net
        public static void main(String[] args) {

                SoundsTest soundsTest = new SoundsTest();
                soundsTest.setTestEnvironment();
                try {
                        soundsTest.testSounds("src/test/36db.wav");
                        soundsTest.testSounds("src/test/42db.wav");
                        soundsTest.testSounds("src/test/48db.wav");
                        soundsTest.testSounds("src/test/54db.wav");
                        soundsTest.testSounds("src/test/60db.wav");
                        soundsTest.testSounds("src/test/66db.wav");
                        soundsTest.testSounds("src/test/72db.wav");
                        soundsTest.testSounds("src/test/78db.wav");

                        // (-54.993771537882466,-57.627165366241364)

                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (UnsupportedAudioFileException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

CalendarAPITest.java

Paquete kodama.synchronization A.14: CalendarAPITest.java

```
package kodama.testing;

import java.io.IOException;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.TimeZone;

import kodama.main.Main;
import kodama.synchronization.CalendarManager;
import kodama.synchronization.calendar.CalendarUrl;
import kodama.synchronization.calendar.model.CalendarEntry;
import kodama.synchronization.calendar.model.EventEntry;
import kodama.synchronization.calendar.model.When;

import org.junit.Test;

import android.test.ActivityInstrumentationTestCase2;

import com.android.ddmlib.Log;
import com.google.api.client.util.DateTime;

/**
 *
 * @author saragarciaperez
 *         <p>
 *          Unit Test for Calendar API
 *          </p>
 */
public class CalendarAPITest extends ActivityInstrumentationTestCase2<Main> {

        public CalendarAPITest() throws ClassNotFoundException {
                super("kodama.main.Main", Main.class);
        }

        DateTime date = null;
        private When when = null;
        private CalendarEntry calendarEntry = null;
        private static CalendarManager gCal = null;

        protected void setUp() {
                getActivity();

                // It's neccesary wait here when debug, because we need time to set the
                // calendar and to start to work with it.
                // Then we have to select the account of gmail and one the app is
                // loaded, continue (Step over)
                gCal = CalendarManager.getInstance();

                try {

                        super.setUp();

                } catch (Exception e) {
                        e.printStackTrace();
                }

                // Calendar is set in setAccount (Main)
                assertEquals("KodamaAwake", calendarEntry.title);

                //the text used is extracted from Don Quijote (Cervantes) and Campos de Castilla (
                     Antonio Machado)
                testLenthName();
                testLenthDescription();
                testLenthDescriptionLonger();
        }

        // Testing the lenth of the attribute title
        @Test
        public void testLenthName() {

                StringBuilder cad = new StringBuilder();

                cad.append("Dichosa edad y siglos dichosos aquéllos a quien los antiguos pusieron");
                cad.append(" nombre de dorados, y no porque en ellos el oro, que en esta nuestra edad")
                     ;
                cad.append(" de hierro tanto se estima, se alcanzase en aquella venturosa sin fatiga");
                cad.append(" alguna, sino porque entonces los que en ella vivían ignoraban estas");
                cad.append(" dos palabras de tuyo y mío. Eran en aquella santa edad todas las cosas");
                cad.append(" comunes. A nadie le era necesario para alcanzar su ordinario sustento");
                cad.append(" tomar otro trabajo que alzar la mano y alcanzarle de las robustas encinas,
                     ");
                cad.append(" que liberalmente les estaban convidando con su dulce y sazonado fruto.");
                cad.append(" Las claras fuentes y corrientes ríos, en magnífica abundancia,");
                cad.append(" sabrosas y transparentes aguas les ofrecían. En las quiebras");
                cad.append(" de las peñas y en lo hueco de los árboles formaban su república las");
                cad.append(" solícitas y discretas abejas, ofreciendo a cualquiera mano,");
                cad.append(" sin interés alguno, la fértil cosecha de su dulcísimo trabajo.");
                cad.append(" Los valientes alcornoques despedían de sí, sin otro artificio que");
                cad.append(" el de su cortesía, sus anchas y livianas cortezas, con que se");
                cad.append(" comenzaron a cubrir las casas, sobre rústicas estacas sustentadas,");
                cad.append(" no más que para la defensa de las inclemencias del cielo.");
                cad.append(" Todo era paz entonces, todo amistad, todo concordia: aún no se había");
```

```
        cad.append(" atrevido la pesada reja del corvo arado a abrir ni visitar las entrañas");
        cad.append(" piadosas de nuestra primera madre; que ella, sin ser forzada, ofrecía,");
        cad.append(" por todas las partes de su fértil y espacioso seno, lo que pudiese");
        cad.append(" hartar, sustentar y deleitar a los hijos que entonces la poseían.");
        cad.append(" Entonces sí que andaban las simples y hermosas zagalejas de valle");
        cad.append(" en valle y de otero en otero, en trenza y en cabello, sin más vestidos");
        cad.append(" de aquéllos que eran menester para cubrir honestamente lo que la ");
        cad.append(" honestidad quiere y ha querido siempre que se cubra, y no eran sus");
        cad.append(" adornos de los que ahora se usan, a quien la púrpura de Tiro y la");
        cad.append(" por tantos modos martirizada seda encarecen, sino de algunas hojas");
        cad.append(" verdes de lampazos y yedra, entretejidas, con lo que quizá iban tan");
        cad.append(" pomposas y compuestas como van agora nuestras cortesanas con las raras");
        cad.append(" y peregrinas invenciones que la curiosidad ociosa les ha mostrado.");

        Log.i("Lenth string prueba", String.valueOf(cad.length()));

        crearEvento(cad);

        EventEntry event = chequearEventos().get(0);

        String nameRead = event.title;

        Log.i("Max. Length", String.valueOf(nameRead.length()));
    }

    // Testing the lenth of the attribute summary or description
    @Test
    public void testLenthDescription() {

        StringBuilder cad = new StringBuilder();

        cad.append("He vuelto a ver los álamos dorados,\n");
        cad.append("álamos del camino en la ribera\n");
        cad.append("del Duero, entre San Polo y San Saturio,\n");
        cad.append("tras las murallas viejas\n");
        cad.append("de Soria barbacana\n");
        cad.append("hacia Aragón, en castellana tierra.\n");
        cad.append("Estos chopos del río, que acompañan\n");
        cad.append("con el sonido de sus hojas secas\n");
        cad.append("el son del agua, cuando el viento sopla,\n");
        cad.append("tienen en sus cortezas\n");
        cad.append("grabadas iniciales que son nombres\n");
        cad.append("de enamorados, cifras que son fechas.\n");
        cad.append("¡Álamos del amor que ayer tuvisteis\n");
        cad.append("de ruiseñores vuestras ramas llenas;\n");
        cad.append("álamos que seréis mañana liras\n");
        cad.append("del viento perfumado en primavera;\n");
        cad.append("álamos del amor cerca del agua\n");
        cad.append("que corre y pasa y sueña,\n");
        cad.append("álamos de las márgenes del Duero,\n");
        cad.append("conmigo vais, mi corazón os lleva!\n");
        cad.append("¡Oh!, sí, conmigo vais, campos de Soria,\n");
        cad.append("tardes tranquilas, montes de violeta,\n");
        cad.append("alamedas del río, verde sueño\n");
        cad.append("del suelo gris y de la parda tierra,\n");
        cad.append("agria melancolía\n");
        cad.append("de la ciudad decrépita,\n");
        cad.append("me habéis llegado al alma,\n");
        cad.append("¿o acaso estabais en el fondo de ella?\n");
        cad.append("¡Gentes del alto llano numantino\n");
        cad.append("que a Dios guardáis como cristianas viejas,\n");
        cad.append("que el sol de España os llene\n");
        cad.append("de alegría, del luz y de riqueza!\n");

        try {
                calendarEntry = gCal.getCalendar("KodamaAwake");
        } catch (IOException e1) {
                e1.printStackTrace();
        }

        EventEntry eventEntry = new EventEntry();
        eventEntry.summary = cad.toString();

        when = new When();

        GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

        when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

        cal.add(GregorianCalendar.MINUTE, 1);

        when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

        eventEntry.when = when;
        try {
                gCal.getClient()
                                .eventFeed()
                                .insert()
                                .execute(new CalendarUrl(calendarEntry.getEventFeedLink()),
                                                eventEntry);
        } catch (IOException e) {
                e.printStackTrace();
        }
```

```
        EventEntry event = chequearEventos().get(0);

        String descRead = event.summary;

        Log.i("Max. Length", String.valueOf(descRead.length()));

    }

    // Testing the lenth of the attribute summary or description with a longer text input
    @Test
    public void testLenthDescriptionLonger() {

        StringBuilder cad = new StringBuilder();
        cad.append("Antonio Machado\n");
        cad.append("¡Soria fría, Soria pura,\n");
        cad.append("cabeza de Extremadura,\n");
        cad.append("con su castillo guerrero\n");
        cad.append("arruinado, sobre el Duero;\n");
        cad.append("con sus murallas roídas\n");
        cad.append("y sus casas denegridas!\n");
        cad.append("¡Muerta ciudad de señores\n");
        cad.append("soldados o cazadores;\n");
        cad.append("de portales con escudos\n");
        cad.append("de cien linajes hidalgos,\n");
        cad.append("y de famélicos galgos,\n");
        cad.append("de galgos flacos y agudos,\n");
        cad.append("que pululan\n");
        cad.append("por las sórdidas callejas,\n");
        cad.append("y a la medianoche ululan,\n");
        cad.append("cuando graznan las cornejas!\n");
        cad.append("¡Soria fría! La campana\n");
        cad.append("Soria, ciudad castellana\n");
        cad.append("¡tan bella! bajo la luna.\n");
        cad.append("VII\n");
        cad.append("¡Colinas plateadas,\n");
        cad.append("grises alcores, cárdenas roquedas\n");
        cad.append("por donde traza el Duero\n");
        cad.append("su curva de ballesta\n");
        cad.append("en torno a Soria, obscuros encinares,\n");
        cad.append("ariscos pedregales, calvas sierras,\n");
        cad.append("caminos blancos y álamos del río,\n");
        cad.append("tardes de Soria, mística y guerrera,\n");
        cad.append("hoy siento por vosotros, en el fondo\n");
        cad.append("del corazón, tristeza,\n");
        cad.append("tristeza que es amor! ¡Campos de Soria\n");
        cad.append("donde parece que las rocas sueñan,\n");
        cad.append("conmigo vais! ¡Colinas plateadas,\n");
        cad.append("grises alcores, cárdenas roquedas!..\n");
        cad.append("He vuelto a ver los álamos dorados,\n");
        cad.append("álamos del camino en la ribera\n");
        cad.append("del Duero, entre San Polo y San Saturio,\n");
        cad.append("tras las murallas viejas\n");
        cad.append("de Soria barbacana\n");
        cad.append("hacia Aragón, en castellana tierra.\n");
        cad.append("Estos chopos del río, que acompañan\n");
        cad.append("con el sonido de sus hojas secas\n");
        cad.append("el son del agua, cuando el viento sopla,\n");
        cad.append("tienen en sus cortezas\n");
        cad.append("grabadas iniciales que son nombres\n");
        cad.append("de enamorados, cifras que son fechas.\n");
        cad.append("¡Álamos del amor que ayer tuvisteis\n");
        cad.append("de ruiseñores vuestras ramas llenas;\n");
        cad.append("álamos que seréis mañana liras\n");
        cad.append("del viento perfumado en primavera;\n");
        cad.append("álamos del amor cerca del agua\n");
        cad.append("que corre y pasa y sueña,\n");
        cad.append("álamos de las márgenes del Duero,\n");
        cad.append("conmigo vais, mi corazón os lleva!\n");
        cad.append("¡Oh!, sí, conmigo vais, campos de Soria,\n");
        cad.append("tardes tranquilas, montes de violeta,\n");
        cad.append("alamedas del río, verde sueño\n");
        cad.append("del suelo gris y de la parda tierra,\n");
        cad.append("agria melancolía\n");
        cad.append("de la ciudad decrépita,\n");
        cad.append("me habéis llegado al alma,\n");
        cad.append("¿o acaso estabais en el fondo de ella?\n");
        cad.append("¡Gentes del alto llano numantino\n");
        cad.append("que a Dios guardáis como cristianas viejas,\n");
        cad.append("que el sol de España os llene\n");
        cad.append("de alegría, del luz y de riqueza!\n");
        cad.append("\nCervantes\n");
        cad.append("Dichosa edad y siglos dichosos aquéllos a quien los antiguos pusieron");
        cad.append(" nombre de dorados, y no porque en ellos el oro, que en esta nuestra edad")
            ;
        cad.append(" de hierro tanto se estima, se alcanzase en aquella venturosa sin fatiga");
        cad.append(" alguna, sino porque entonces los que en ella vivían ignoraban estas");
        cad.append(" dos palabras de tuyo y mío. Eran en aquella santa edad todas las cosas");
        cad.append(" comunes. A nadie le era necesario para alcanzar su ordinario sustento");
        cad.append(" tomar otro trabajo que alzar la mano y alcanzarle de las robustas encinas,
            ");
        cad.append(" que liberalmente les estaban convidando con su dulce y sazonado fruto.");
        cad.append(" Las claras fuentes y corrientes ríos, en magnífica abundancia,");
        cad.append(" sabrosas y transparentes aguas les ofrecían. En las quiebras");
        cad.append(" de las peñas y en lo hueco de los árboles formaban su república las");
```

```
            cad.append(" solícitas y discretas abejas, ofreciendo a cualquiera mano,");
            cad.append(" sin interés alguno, la fértil cosecha de su dulcísimo trabajo.");
            cad.append(" Los valientes alcornoques despedían de sí, sin otro artificio que");
            cad.append(" el de su cortesía, sus anchas y livianas cortezas, con que se");
            cad.append(" comenzaron a cubrir las casas, sobre rústicas estacas sustentadas,");
            cad.append(" no más que para la defensa de las inclemencias del cielo.");
            cad.append(" Todo era paz entonces, todo amistad, todo concordia: aún no se había");
            cad.append(" atrevido la pesada reja del corvo arado a abrir ni visitar las entrañas");
            cad.append(" piadosas de nuestra primera madre; que ella, sin ser forzada, ofrecía,");
            cad.append(" por todas las partes de su fértil y espacioso seno, lo que pudiese");
            cad.append(" hartar, sustentar y deleitar a los hijos que entonces la poseían.");
            cad.append(" Entonces sí que andaban las simples y hermosas zagalejas de valle");
            cad.append(" en valle y de otero en otero, en trenza y en cabello, sin más vestidos");
            cad.append(" de aquéllos que eran menester para cubrir honestamente lo que la ");
            cad.append(" honestidad quiere y ha querido siempre que se cubra, y no eran sus");
            cad.append(" adornos de los que ahora se usan, a quien la púrpura de Tiro y la");
            cad.append(" por tantos modos martirizada seda encarecen, sino de algunas hojas");
            cad.append(" verdes de lampazos y yedra, entretejidas, con lo que quizá iban tan");
            cad.append(" pomposas y compuestas como van agora nuestras cortesanas con las raras");
            cad.append(" y peregrinas invenciones que la curiosidad ociosa les ha mostrado.");

            try {
                    calendarEntry = gCal.getCalendar("KodamaAwake");
            } catch (IOException e1) {
                    e1.printStackTrace();
            }

            EventEntry eventEntry = new EventEntry();
            eventEntry.summary = cad.toString();

            when = new When();

            GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

            when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

            cal.add(GregorianCalendar.MINUTE, 1);

            when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

            eventEntry.when = when;
            try {
                    gCal.getClient()
                                    .eventFeed()
                                    .insert()
                                    .execute(new CalendarUrl(calendarEntry.getEventFeedLink()),
                                                    eventEntry);
            } catch (IOException e) {
                    e.printStackTrace();
            }

            EventEntry event = chequearEventos().get(0);

            String descRead = event.summary;

            Log.i("Max. Length", String.valueOf(descRead.length()));

    }

    private void crearEvento(StringBuilder cad) {

            try {
                    calendarEntry = gCal.getCalendar("KodamaAwake");
            } catch (IOException e1) {
                    e1.printStackTrace();
            }
            EventEntry eventEntry = new EventEntry();
            eventEntry.title = "testLenthName" + ";" + cad.toString();

            When when = new When();

            GregorianCalendar cal = new GregorianCalendar(TimeZone.getDefault());

            when.startTime = new DateTime(cal.getTime(), TimeZone.getDefault());

            cal.add(GregorianCalendar.MINUTE, 1);

            when.endTime = new DateTime(cal.getTime(), TimeZone.getDefault());

            eventEntry.when = when;

            String eventFeedLink = calendarEntry.getEventFeedLink();

            CalendarUrl calUrl = new CalendarUrl(eventFeedLink);

            try {
                    gCal.getClient().eventFeed().insert().execute(calUrl, eventEntry);
            } catch (IOException e) {
                    e.printStackTrace();
            }

    }

    private List<EventEntry> chequearEventos() {
```

```
                List<EventEntry> events = null;
                try {
                        events = gCal.getClient().eventFeed().list()
                                        .execute(new CalendarUrl(calendarEntry.getEventFeedLink())).
                                            events;
                } catch (IOException e) {
                        e.printStackTrace();
                }

                return events;
        }

}
```

SensorSensibilityTest.java

Paquete kodama.synchronization A.15: SensorSensibilityTest.java

```
package kodama.testing;

import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Calendar;
import java.util.GregorianCalendar;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.UnsupportedAudioFileException;
import org.junit.Test;


import kodama.sensor.audio.SoundManager;

/**
 *
 * @author saragarciaperez
 *
 *
 *          Testing the sensibility of the sensor
 */
public class SensorSensibilityTest {

        public SensorSensibilityTest() {
                setTestEnvironment();
        }

        private SoundManager soundManager;
        private ByteArrayOutputStream out = new ByteArrayOutputStream();
        private AudioInputStream in;

        protected void setUp() {
                System.out
                                .println("TEST-SENSIBILITY --> dB Full Scale [Min: -100, Max: 0]\n");

                try {

                        // desde micro -55.0 -diferencia => 0.73
                        testSounds("src/test/grifocerca.wav");
                        testSounds("src/test/grifolejos.wav");
                        // With the sensor near the principal door, portero in kitchen
                        testSounds("src/test/portero.wav");
                        testSounds("src/test/toctoc.wav");
                        testSounds("src/test/comedor.wav");
                        testSounds("src/test/tele.wav");
                        testSounds("src/test/oficinahoracomida.wav");
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (UnsupportedAudioFileException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        public void setTestEnvironment() {
                soundManager = new SoundManager();
        }

        @Test
        public void testSounds(String file) throws FileNotFoundException,
                        UnsupportedAudioFileException, IOException {

                in = AudioSystem.getAudioInputStream(new FileInputStream(file));

                int read;
```

```
                byte[] buff = new byte[1024];
                while ((read = in.read(buff)) > 0) {
                        out.write(buff, 0, read);
                }
                out.flush();

                byte[] audioBytes = out.toByteArray();

                // double decibelios = soundManager.calculatePowerDb(audioBytes);

                // System.out.println("Decibelios: "+ String.valueOf(decibelios));

                processAudio(audioBytes, file);
                out.reset();
        }

        // ********************************************************************* //
        // Main Loop of sensor activity (Testing)
        // ********************************************************************* //
        /**
         * Processes audio, calculates dB and calls the CalendarManager to create a
         * new event if it takes place
         *
         * @param buffer
         */
        private void processAudio(byte[] buffer, String file) {

                double dBTransformation = 0;
                double currentPower = soundManager.calculatePowerDb(buffer);

                dBTransformation = currentPower + 0.73;

                /*
                 * <string-array name="sensibility_array"> <item>@string/high</item>
                 * <item>@string/medium</item> <item>@string/low</item> </string-array>
                 *
                 *
                 * <string-array name="sensibility_array_values"> <item>-80</item> <!--
                 * fijar limite inferior --> <item>-58</item> <!-- mayor que -58 ,
                 * limite sensibiliad media--> <item>-55</item> <!-- es mayor
                 * que -55.8 => sensibilidad baja --> </string-array>
                 */

                System.out.println(file + " Current power: "
                                + String.valueOf(dBTransformation));

                if (currentPower > -80) {

                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        System.out.println("HIGH > -80");
                }

                if (currentPower > -58) {

                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        System.out.println("MEDIUM > -58");
                }

                if (currentPower > -55) {

                        Calendar cal = new GregorianCalendar();
                        cal.add(Calendar.SECOND, 1);
                        System.out.println("LOW > -55");
                }
                System.out.println("\n");
        }

        public static void main(String[] args) {
                SensorSensibilityTest sst = new SensorSensibilityTest();
                sst.setUp();
        }
}
```

NotificationTest.java

Paquete kodama.synchronization A.16: NotificationTest.java

```
package kodama.testing;

import java.util.List;

import kodama.main.Main;
import kodama.receiver.model.Notification;
import kodama.synchronization.CalendarManager;

import org.junit.Test;
```

```
import android.test.ActivityInstrumentationTestCase2;

import com.android.ddmlib.Log;
import com.google.api.client.util.DateTime;

/**
 *
 * @author saragarciaperez
 *         <p>
 *          Unit Test for Notifications
 *         </p>
 */
public class NotificationTest extends ActivityInstrumentationTestCase2<Main> {

        public NotificationTest() throws ClassNotFoundException {
                super("kodama.main.Main", Main.class);
        }

        CalendarManager calManager = null;
        DateTime date = null;

        protected void setUp() {

                try {

                        super.setUp();

                } catch (Exception e) {
                        e.printStackTrace();
                }

                getActivity();

                calManager = Main.getGestorCalendar();

                // It's neccesary wait here when debug, because we need time to set the
                // calendar and to start to work with it.
                // Then we have to select the account of gmail and one the app is
                // loaded, continue (Step over)

                testCheckEvents(0);

                testCheckEvents(0);
                testCreateEvent();
                testCheckEvents(1);
                testCreateEvent();
                testCreateEvent();
                testCheckEvents(2);

                long ms1 = System.currentTimeMillis();

                // testing also the performance where we have a lot of events

                int cont = 10;
                do {
                        testCreateEvent();
                        cont--;
                } while (cont > 0);

                long ms2 = System.currentTimeMillis();

                Log.i("Create 10 events", String.valueOf(ms2 - ms1));

                long ms3 = System.currentTimeMillis();

                testCheckEvents(10);

                long ms4 = System.currentTimeMillis();

                Log.i("Check new 10 events", String.valueOf(ms4 - ms3));

                testCheckFormatedDate();
        }

        @Test
        public void testCreateEvent() {

                calManager.setCalendar();

                calManager.createEvent("here", "test");
        }

        @Test
        public void testCheckEvents(int num) {

                if (calManager.checkEvents() != null)
                        assertEquals(num, calManager.checkEvents().size());
                else
                        assertEquals(num, 0);
        }

        //We use a simple reg Exp to check the format of the date
        public void testCheckFormatedDate() {
```

```
                calManager.createEvent("NotificationTest", "testCheckFormatTime");

                try {
                        Thread.sleep(2000);
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }

                List<Notification> notifs = calManager.checkEvents();

                String time = notifs.get(0).getTime();

                assertEquals(true,
                                time.matches("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}"));

        }
}
```

1.12. Paquete kodama.util

Constants.java

Paquete kodama.util A.17: Constants.java

```
package kodama.util;

/**
 * <p>
 * Constants used by Kodama Awake application.
 * </p>
 * @author saragarciaperez
 *
 */
public class Constants {
        /**
         * Should be used by all log statements
         */
        public static final String VERSION = "2.0";

        /**
         * onActivityResult request codes:
         */
        public static final int GET_LOGIN = 0;
        public static final int AUTHENTICATED = 1;
        public static final int CREATE_EVENT = 2;
        public static final String AUTH_TOKEN_TYPE = "cl";
        public static final int REQUEST_AUTHENTICATE = 0;
        public static final String TAG = "Kodama Awake";
        public static final String PREF = TAG;
        public static final String PREF_ACCOUNT_NAME = "accountName";
        public static final String PREF_AUTH_TOKEN = "authToken";
        public static final String PREF_GSESSIONID = "gsessionid";

        /**
         * The type of account that we can use for API operations.
         */
        public static final String ACCOUNT_TYPE = "com.google";

        /**
         * The name of the service to authorize for.
         */
        public static final String OAUTH_SCOPE = "oauth2:https://www.googleapis.com/auth/calendar";

        public static final String SENSOR_NAME = "sensor_name";
        public static final String SENSOR_LOCATION = "sensor_location";
        public static final String SENSIBILITY = "sensibility";

}
```

1.13. Fichero AndroidManifest.xml

Paquete kodama.util A.18: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kodamaAwake.audio"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="8" />

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.USE_CREDENTIALS" >
    </uses-permission>
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />

    <instrumentation
        android:name="android.test.InstrumentationTestRunner"
        android:targetPackage="kodamaAwake.audio" >
    </instrumentation>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <uses-library android:name="android.test.runner" />

        <activity
            android:name="kodama.main.Main"
            android:label="@string/app_name" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="android.intent.category.DEFAULT" />

                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.receiver.ReceiverActivity" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.sensor.SensorActivity" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.sensor.preferences.Configuration" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.PREFERENCE" />
            </intent-filter>
        </activity>
        <activity android:name="kodama.receiver.RegisterActivity" >
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

1.14. Fichero main.xml

Paquete kodama.util A.19: layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical"
        android:padding="5dp" >

        <TabWidget
```

```
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />

        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp" />
    </LinearLayout>

</TabHost>
```

1.15. Fichero notifications.xml

Paquete kodama.util A.20: layout/notifications.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:keepScreenOn="true">

    <ListView
        android:id="@+id/lvEvents"
        android:layout_width="fill_parent"
        android:layout_height="364dp"
        android:layout_weight="2.43" >
    </ListView>

    <LinearLayout
        android:id="@+id/buttons"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right" >

        <ToggleButton
            android:id="@+id/toggleBtNotif"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:textOff="ON"
            android:textOn="ON" />

        <Button
            android:id="@+id/btRegister"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="right"
            android:layout_marginLeft="10dp"
            android:text="@string/register" />
    </LinearLayout>

</LinearLayout>
```

1.16. Fichero notificationRow.xml

Paquete kodama.util A.21: layout/notificationRow.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#EEEEEE"
    android:stretchColumns="*" >

    <TextView
        android:id="@+id/sensor_location"
        android:layout_width="0dip"
        android:layout_weight="2"
```

```
        android:gravity="center"
        android:textColor="#000000"
        android:textSize="18dip"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/sensor_name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textColor="#000000"
        android:textSize="18dip"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/time"
        android:layout_width="0dip"
        android:layout_margin="1dip"
        android:layout_weight="1"
        android:gravity="center"
        android:textColor="#000000"
        android:textSize="18dip" />

</TableLayout>
```

1.17. Fichero recording.xml

Paquete kodama.util A.22: layout/recording.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="right"
    android:orientation="vertical" >

    <LinearLayout
        android:id="@+id/config_values"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/sensor_name_pref"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/not_defined_not_defined"
            android:textAppearance="?android:attr/textAppearanceLarge" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="0.76"
        android:gravity="right"
        android:orientation="vertical" >

        <ImageView
            android:id="@+id/wave"
            android:layout_width="wrap_content"
            android:layout_height="0dip"
            android:layout_weight="0.46"
            android:src="@drawable/onda4"
            android:contentDescription="@string/wave"/>

        <TextView
            android:id="@+id/text_dB"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dip"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <LinearLayout
            android:id="@+id/buttons"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="right" >

            <ToggleButton
                android:id="@+id/toggleBtPlay"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="center_vertical"
                android:gravity="center"
                android:textOff="ON"
```

```
                android:textOn="ON" />

            <Button
                android:id="@+id/btConfiguration"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="right"
                android:layout_marginLeft="10dp"
                android:text="@string/configuration" />
        </LinearLayout>

    </LinearLayout>

</LinearLayout>
```

1.18. Fichero preferences.xml

Paquete kodama.util A.23: xml/preferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory android:title="@string/sensor_config" >
        <EditTextPreference
            android:name="@string/name"
            android:defaultValue="@string/not_defined"
            android:key="sensor_name"
            android:summary="@string/sensor_name_desc"
            android:title="@string/sensor_name" />
        <EditTextPreference
            android:name="@string/place"
            android:defaultValue="@string/not_defined"
            android:key="sensor_location"
            android:summary="@string/sensor_location_desc"
            android:title="@string/sensor_location" />
            <ListPreference android:summary="@string/sensibility_desc" android:title="@string/
                sensibility" android:dialogTitle="@string/sensibility" android:key="sensibility"
                android:entryValues="@array/sensibility_array_values" android:entries="@array/
                sensibility_array"/>
    </PreferenceCategory>

</PreferenceScreen>
```

1.19. Fichero arrays.xml

Paquete kodama.util A.24: values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="sensibility_array">
        <item>@string/high</item>
        <item>@string/medium</item>
        <item>@string/low</item>
    </string-array>


    <string-array name="sensibility_array_values">
        <item>-80</item> <!-- lower limit -->
        <item>-58</item> <!-- greater than -58, medium sensibility limit-->
        <item>-55</item> <!-- is greater than -55 => low sensitivity -->
    </string-array>
</resources>
```

1.20. Fichero strings.xml

Paquete kodama.util A.25: values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <!-- app general -->
    <string name="app_name">KodamaAwake</string>

    <!-- sensor -->
    <string name="start_recording">Sensor</string>
    <string name="select">Select</string>
    <string name="wave">Wave of sound</string>
    <string name="configuration">Config.</string>

    <!-- sensor preferences-->
    <string name="sensor_config">Sensor Configuration</string>
    <string name="name">Name</string>
    <string name="place">Location</string>
    <string name="sensibility">Sensibility</string>
    <string name="sensibility_prompt">Sensibility</string>
    <string name="sensibility_desc"></string>
    <string name="sensor_name">Sensor Name</string>
    <string name="sensor_name_desc">Set the sensor name</string>
    <string name="sensor_location">Sensor Location</string>
    <string name="sensor_location_desc">Set the sensor location</string>
    <string name="not_defined_not_defined">Localization - Sensor</string>
    <string name="not_defined">Not defined</string>

    <!-- sensor preferences for sensibility -->
    <string name="low">Low</string>
    <string name="medium">Medium</string>
    <string name="high">High</string>

    <!-- notifications -->
    <string name="check_events">Notifications</string>
    <string name="time">Time</string>
    <string name="register">Register</string>


</resources>
```