Títol: Aplicación de dispositivos móviles para secretario de reuniones.

Volum: 1

Alumne: Antonio Leandro Páez

Director/Ponent: Javier Béjar Alonso

Departament: Intel·ligència Artificial

DADES DEL PROJECTE

Títol del Projecte: Aplicación de dispositivos móviles para secretario de reuniones.

Nom de l'estudiant: Antonio Leandro Páez

Titulació: Ingeniería Superior en Informática

Crèdits: 37,5

Director/Ponent: Javier Béjar Alonso

Departament: Intel·ligència Artificial

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Javier Vazquez Salceda

Vocal: Mercè Mora Giné

Secretari: Javier Béjar Alonso

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

Contenido

Agradecimientos

Quiero agradecer a mis amigos y compañeros por haberme apoyado durante estos años.

A todos los miembros de mi familia, especialmente a mi abuela, que sé que le hace mucha ilusión verme licenciado.

Pero, sobre todo, a mis padres y a mi hermana. Por haber estado a mi lado en los malos momentos, por haberme ayudado cuando lo he necesitado y por conseguir que llegara este momento. Una parte del título es de ellos.

Gracias a todos por haber estado conmigo.

Y a mi abuelo, que lo estará siempre.

1. Introducción

La aplicación "LeAndroid Meeting" es una utilidad diseñada para facilitar el trabajo de secretario de reuniones. El programa permite llevar el control de cada uno de los puntos de la reunión, crear todas sus estructuras de datos necesarias y, además, facilitar la realización del acta.

La aplicación permite crear y editar reuniones detallando todos sus datos, además de su orden del día y de sus asistentes. También se pueden crear participantes, con sus respectivos datos. Todas estas estructuras serán guardadas en la base de datos de la aplicación con el sistema de gestión de bases de datos SQLite.

El programa ayuda al usuario a desarrollar una reunión. El moderador podrá, fácilmente, controlar manualmente los turnos de los participantes, teniendo como ayuda varios indicadores de tiempo. Para guardar el acta, la aplicación permite grabar audio de los asistentes y realizar transcripciones, todo ello etiquetado con el nombre del asistente que lo protagonice. También permite tomar apuntes durante la reunión.

Los datos del acta se guardarán como archivos de texto o de audio y podrán ser consultados cuando el usuario así lo desee, ordenados por punto del día, participante y tiempo.

La aplicación ofrece una interfaz gráfica intuitiva, profesional y agradable, haciendo la navegación muy fácil. El usuario no necesitará tiempo de aprendizaje para moverse por todas las opciones y sacarle todo el provecho al programa.

LeAndroid Meeting está destinado a plataformas móviles y tablets que utilicen el sistema operativo Android. El objetivo ha sido que el máximo número de personas puedan utilizar esta aplicación. Así pues, se ha optado por hacerla compatible con las versiones de Android 2.2 y posteriores, lo que hace un 99% de los dispositivos Android actuales.

2. Objetivos

En este capítulo se explicarán los objetivos que se querían conseguir, los problemas o caminos a elegir y las decisiones finales

2.1 Objetivos iniciales

Afortunadamente, las pretensiones iniciales no han distado demasiado de la aplicación final, pero sí ha habido cambios respecto al concepto original.

La aplicación pretendía ser un programa que ayudara al secretario de reuniones, que pudiera levantar acta de forma sencilla y que permitiera realizar transcripciones y reconociera las voces.

Las estructuras de datos (reuniones, participantes y orden del día), que en un principio iban a ser más pequeñas, no tenían decidido su tipo de almacenamiento, siendo XML y SQLite sus dos opciones.

La APP pretendía ser una herramienta que guiara al usuario al realizar la reunión.

Se estudió con detenimiento el reconocimiento de voces y la transcripción de audio, sabiendo que era complicada su implementación en la aplicación. Se propuso, como alternativa al reconocimiento de voz, la opción de selección manual de participantes.

Inicialmente, la única forma planteada de guardar actas era tomar notas y la posible transcripción de voces.

Finalmente, como la mayoría de pantallas navegables eran listas o campos de texto colocados verticalmente, se pensó en diseñar la aplicación sólo en modo vertical.

2.2 Problemas

Durante las primeras fases de planificación se comenzaron a ver los problemas y se disiparon muchas de las dudas iniciales del proyecto.

Las estructuras de datos se hicieron más grandes y su almacenamiento en XML era del todo ineficiente. Para cada dato nuevo, requeriría borrar todo el archivo y crearlo de nuevo. Además, algunas estructuras están relacionadas. Por todo esto, se decidió que el sistema en el que se guardaría los datos de la aplicación sería SQLite.

Después de investigar sobre reconocimiento de voz y transcripción, llegaron los peores presagios. Es necesario que el usuario grabe muestras de las voces de todos los participantes. Posteriormente, en un ordenador, se deberían general los modelos de cada uno de los asistentes. Los modelos pueden tardar horas o días en generarse ya que deben realizar una fase de "entrenamiento". En esta fase, se comparan con las demás muestras y se establecen los límites de cada modelo. Todo eso se debería realizar antes de las reuniones. Una vez en una reunión, cuando el asistente hable, la aplicación debería enviar al ordenador la muestra para que la compare con los modelos. Si obtiene una respuesta correcta, el ordenador debe enviar a la aplicación los resultados. Para desarrollar este tipo de tecnología que funcionase adecuadamente requeriría varios años de testeos y, una vez llegase al mercado, requeriría al usuario ralentizar sus reuniones y un equipo potente para crear sus modelos y compararlos.

Además, todo esto no va acorde a la filosofía de la aplicación: una aplicación fácil e intuitiva para ayudar al secretario de reuniones.

Por fortuna, no ocurre lo mismo con las transcripciones. Google ofrece un servicio por el cual podemos enviarle un audio y ellos generan la respuesta. Aunque el principio es el mismo que el comentado anteriormente, la ventaja aquí es que Google ya tiene los modelos con todas las palabras en sus servidores y sólo hay que enviarle la muestra para que éste nos dé la respuesta. Obviamente, esto nos exige conexión a internet.

Obviamente, todos sabemos que, en ocasiones, las respuestas no son del todo correctas. Así que se decidió implementar una nueva opción para guardar acta: la grabación de audio. Si no queremos la transcripción, siempre podemos guardar el audio y escucharlo posteriormente. También se modificó el concepto de notas. En lugar de ser notas de reunión, serían notas asociadas a la reunión, al punto del día y al participante.

2.3 Decisión final

Pese a conservar la mayor parte de la idea inicial, la aplicación sufrió algunos cambios durante su evolución.

Para almacenar los datos se empleó SQLite. Sin embargo, con la inclusión de los archivos de audio, se desechó la idea de guardar las actas en una tabla de SQLite y se optó por guardarlas en archivos de texto y audio.

Se implementaron las transcripciones utilizando el servicio de Google, la nueva funcionalidad de grabar audio y el reproductor de estos archivos en la sección de actas. Considerando la problemática del reconocimiento del hablante, se implementó la selección manual, haciéndola rápida y fácil para el usuario.

Finalmente, una de las partes que más variaciones sufrió fue la interfaz gráfica. Inicialmente, cada una de las funcionalidades iba a estar disponible a través de un botón. Esto sería ideal en un programa de PC, sin embargo, el exceso de botones en una aplicación móvil resultaba visualmente recargada. Por eso, se decidió implementar adaptadores personalizados para las listas de Android, de tal forma que guardaran información sobre los elementos seleccionados y utilizar botones únicos en la pantalla (o Activity, en lenguaje Android), en lugar de botones por cada elemento de la lista. También se implementaron menús contextuales para limpiar la interfaz gráfica y adaptarla a los tiempos que corren.

Una vez acabado el diseño, se quiso realizar el esfuerzo de adaptar la interfaz a la vista apaisada. Se tuvieron que rediseñar todos los elementos de todas las pantallas, pero el resultado mereció la pena. A pesar de que, al contener listas la mayoría de las pantallas, es más eficiente la utilización del modo retrato, la opción apaisada resulta interesante para los usuarios acostumbrados a utilizar sus móviles o tablets de esta manera.

3. Tecnologías

En este capítulo se hablará de las tecnologías principales que intervienen en el proyecto. Se explicará en qué consisten y qué aportan, que dificultan o qué implicación tienen en la aplicación.

Hablaremos del sistema operativo sobre el cual corre la aplicación, el sistema de gestión de base de datos usado para sus estructuras, el lenguaje de programación principal del software, del lenguaje de marcas usado por el propio sistema operativo para crear la interfaz gráfica y, por último, de la plataforma utilizada para programar y probar la aplicación.

3.1 Android

¿Cómo definir Android?

Android es un sistema operativo de Google creado principalmente para dispositivos táctiles, ya sean teléfonos móviles, tablets u otros dispositivos inteligentes. Es un sistema operativo basado en el kernel de Linux y de código abierto, desarrollado por Android Inc. (empresa que en 2005 fue comprada por Google)

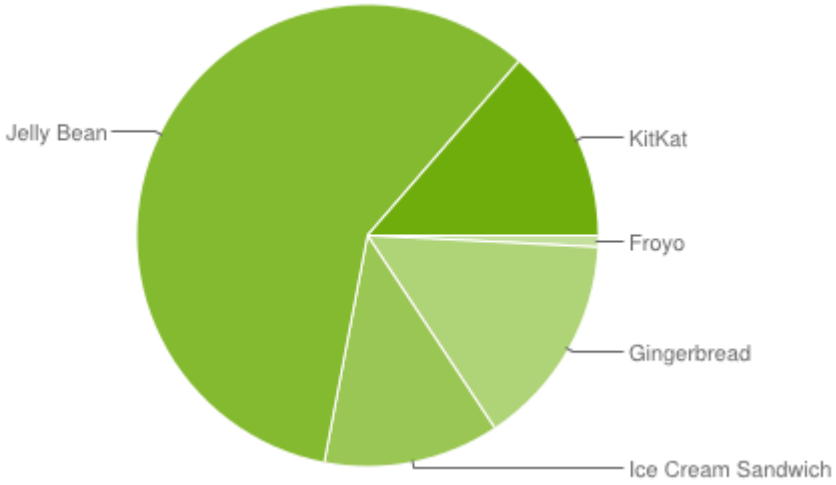
La primera versión de Android llegó en 2008 y, desde entonces, no ha parado de crecer. Actualmente, ya posee más del 50% de la cuota de mercado, por encima del iOS de Apple.

Android es el sistema operativo sobre el que corre la aplicación y uno de los principales campos de batalla de todo el proyecto.

Primero se realizó un estudio con los porcentajes de personas que usaban las versiones de Android y se decidió que se utilizaría como versión mínima la 2.2, para llegar a, práctimente el 100%.

A continuación, vemos unos gráficos con la distribución de las versiones:

Versión	Nombre de versión	API	Distribución
2.2	Froyo	8	0,8%
2.3.3 - 2.3.7	Gingerbread	10	14.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	12.3%
4.1.x	Jelly Bean	16	29.0%
4.2.x		17	19.1%
4.3		18	10.3%
4.4	KitKat	19	13.6%

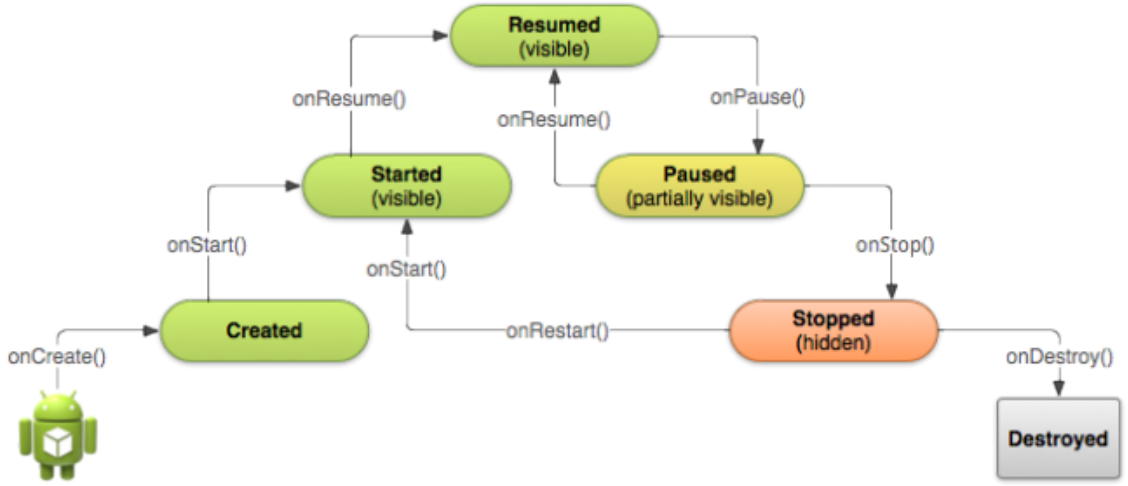


Estos datos fueron publicados por Android el 4 de junio de 2014 teniendo en cuenta las visitas a Google Play, la tienda de Google. Sin embargo, la tienda sólo tiene soporte para las versiones 2.2 o superiores. Sin embargo, se calcula que un porcentaje inferior al 1% utiliza versiones inferiores a la 2.2.

Android tiene sus particularidades para programar y entender este funcionamiento es vital para tener éxito al construir una aplicación.

Android trabaja con *activities*. Una *activity* es cada una de las pantallas de la aplicación. Tiene dos partes: una lógica y una gráfica. La parte gráfica es un archivo *XML* donde se describen los widgets que se van a utilizar, así como los *layouts*. La parte lógica es un archivo java en el que se programan las operaciones, los eventos de los *widgets* del archivo *XML* y se realizan acciones según el estado actual en el ciclo de vida de la actividad.

El ciclo de vida de una actividad es uno de los conceptos más importantes de Android. Una actividad se crea cuando aparece en pantalla. Si pasamos a otra actividad, salimos de la aplicación o cambiamos de orientación el móvil, esta actividad se detiene y se destruye. En el caso del cambio de orientación se vuelve a construir. Sólo si vemos una parte de la actividad mientras otra aplicación obtiene el foco del sistema operativo, entonces la actividad estará pausada. El gráfico que se muestra a continuación muestra las operaciones que el sistema operativo llama al pasar de un estado a otro. Se pueden sobreescribir estas operaciones para evitar perder datos, cerrar canales o cerrar dispositivos (cámara, micrófono…) y para describir el comportamiento de la actividad cuando se crea y recibe el foco.



Otras de las complicaciones es para pasar datos entre actividades. Para esto existen los *bundles*, que son una especie de tabla de *hash*, que sirven para enviar datos a otra aplicación.

Android tiene sus peculiaridades y complicaciones y por eso es necesario entenderlas para poder sacar toda la potencia de este sistema operativo, que es mucha.

3.2 Java

Java es un lenguaje de alto nivel orientado a objetos, desarrollado por la empresa *Sun Microsystem* (actualmente propiedad de *Oracle* Corporation) y cuya primera versión se publicó en 1995. Es un lenguaje parecido en sintaxis a *C* y *C++*, sin embargo tiene algunas diferencias. *Java* tiene menos llamadas a sistema y es algo más lento en su ejecución. Entonces, ¿qué hace que este lenguaje sea tan popular? Los programas de *Java* se ejecutan sobre una máquina virtual, esto hace que no se tenga que tener en cuenta la arquitectura de la máquina a la hora de programar. Además, *Java* tiene una gran comunidad apoyándolo, haciendo que una gran cantidad de librerías estén disponibles para los programadores.

A pesar de que, internamente, *Android* está programado en *C*, en sus aplicaciones utiliza *Java*, teniendo así toda la potencia de este lenguaje de programación, pudiendo utilizar sus librerías y algunas propias de *Android*. Además, con este lenguaje se evitan las preocupaciones por las arquitecturas de los dispositivos móviles o tablets.

Gracias a *Java*, programar en *Android* hace que la curva de dificultad inicial no sea tan elevada ya que, si dominas este lenguaje de programación, rápidamente podrás utilizar toda su potencia para crear aplicaciones, siempre teniendo en cuenta las particulares de *Android*.

3.3 SQLite

Las bases de datos relacionales fueron creadas por IBM en la década de los 70 (al menos, las postulaciones iniciales). Las tablas donde se guardan los datos están relacionadas entre ellas.

Android utiliza SQLite como sistema de gestión de base de datos. Esto es debido a lo liviana que es, algo fundamental para dispositivos móviles de poca memoria.

SQLite está limitada en algunos aspectos. Tiene pocos tipos de datos y no posee claves foráneas. Sin embargo, la cantidad de ventajas la hace ideal para una aplicación móvil: es portable, es estable, compatible con ACID, es un sistema libre y es muy eficiente.

3.4 Eclipse

Eclipse es un *IDE* (entorno desarrollo) de código abierto utilizado inicialmente para programar *Java*. Sin embargo, se puede utilizar para programar en una gran cantidad de lenguajes.

Hasta hace poco, era la única opción real de programar en *Android*. Sin embargo, está en desarrollo un nuevo IDE para este sistema operativo: *Android Studio*.

Este *IDE* es más eficiente ya que es exclusivamente para *Android*, sin embargo, aún está en desarrollo y es conveniente esperar a que sea una versión estable para comenzar un proyecto.

Con *Eclipse* se puede programar en *Android* al instalar el *plugin ADT*, que son las herramientas de desarrollo de *Android*. Con este *IDE* podemos

programar cómodamente, crear actividades con facilidades, ayudando a la creación de la parte gráfica y, sobre todo, ofrece un emulador de *Android*. Con este emulador podemos configurar un dispositivo de varios tamaños de pantalla y de diferentes versiones de *Android*, algo ideal para comprobar cómo se verá la aplicación en diferentes dispositivos móviles.

4. Planificación

En este capítulo, se expondrá la planificación que se hizo para desarrollar el proyecto.

4.1 Alcance

A continuación, se expondrá el alcance del proyecto. Es decir, el conjunto de funcionalidades que debían incorporarse al proyecto.

La aplicación debe tener las estructuras de reunión, participante, turno y orden del día como datos consultables, modificables y que se puedan añadir o eliminar.

También se debe ofrecer al usuario las opciones de consultar las actas de las reuniones finalizadas.

Al desarrollar las reuniones, se deben mostrar los cronómetros con los tiempos acumulados de todos los participantes de la reunión en ese punto del orden del día. Además, el usuario podrá seleccionar manualmente a los participante activos y podrá parar y reanudar la reunión.

El usuario podrá ir guardando el acta a través de notas, grabaciones de audio y transcripciones.

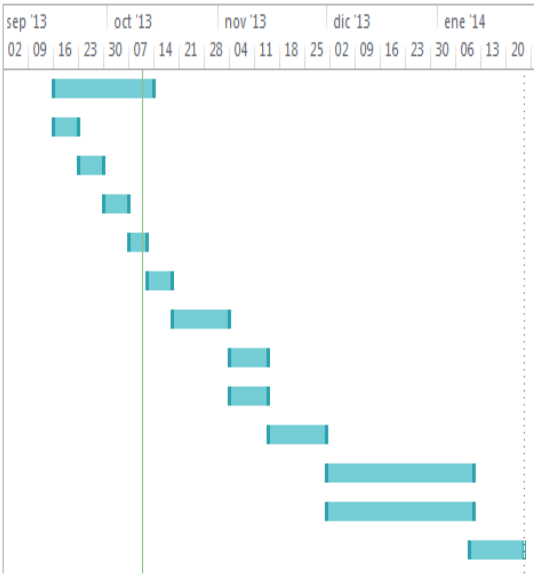
Finalmente, el usuario podrá reproducir todos los archivos guardados al finalizar la reunión.

A la hora de la planificación, cumplir con el alcance era lo más importante. Esto hizo que, para implementar todas las funcionalidades, se tuviera que ampliar el tiempo de desarrollo.

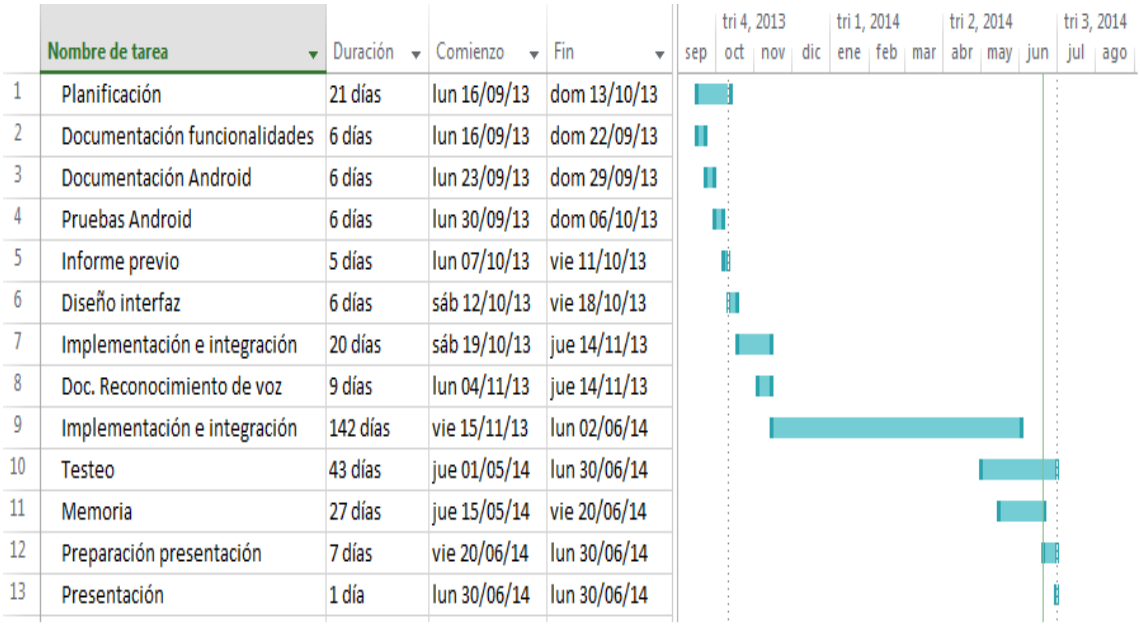
4.2 Tiempo estimado

Como se ha comentado en el apartado anterior, debido al largo desarrollo y con el objetivo de tener todas las funcionalidades desarrolladas y cumplir con el alcance planificado, el tiempo de desarrollo del proyecto se incrementó con respecto al planificado inicialmente. Esta es la planificación temporal inicial:

	Nombre de tarea	Duración	Comienzo	Fin
1	Planificación	21 días	lun 16/09/13	dom 13/10/13
2	Documentación funcionalidades	6 días	lun 16/09/13	dom 22/09/13
3	Documentación Android	6 días	lun 23/09/13	dom 29/09/13
4	Pruebas Android	6 días	lun 30/09/13	dom 06/10/13
5	Informe previo	5 días	lun 07/10/13	vie 11/10/13
6	Diseño interfaz	6 días	sáb 12/10/13	vie 18/10/13
7	Implementación e integración	12 días	sáb 19/10/13	dom 03/11/13
8	Testeo	9 días	lun 04/11/13	jue 14/11/13
9	Doc. Reconocimiento de voz	9 días	lun 04/11/13	jue 14/11/13
10	Implementación e integración	12 días	vie 15/11/13	sáb 30/11/13
11	Testeo	31 días	dom 01/12/13	vie 10/01/14
12	Memoria	31 días	dom 01/12/13	vie 10/01/14
13	Presentación	11 días	vie 10/01/14	vie 24/01/14



Además, durante unos meses, de detuvo el desarrollo debido a cuestiones externas. Este gráfico muestra la evolución temporal real del proyecto:



	Nombre de tarea	Duración	Comienzo	Fin
1	Planificación	21 días	lun 16/09/13	dom 13/10/13
2	Documentación funcionalidades	6 días	lun 16/09/13	dom 22/09/13
3	Documentación Android	6 días	lun 23/09/13	dom 29/09/13
4	Pruebas Android	6 días	lun 30/09/13	dom 06/10/13
5	Informe previo	5 días	lun 07/10/13	vie 11/10/13
6	Diseño interfaz	6 días	sáb 12/10/13	vie 18/10/13
7	Implementación e integración	20 días	sáb 19/10/13	jue 14/11/13
8	Doc. Reconocimiento de voz	9 días	lun 04/11/13	jue 14/11/13
9	Implementación e integración	142 días	vie 15/11/13	lun 02/06/14
10	Testeo	43 días	jue 01/05/14	lun 30/06/14
11	Memoria	27 días	jue 15/05/14	vie 20/06/14
12	Preparación presentación	7 días	vie 20/06/14	lun 30/06/14
13	Presentación	1 día	lun 30/06/14	lun 30/06/14

4.3 Costes

Desarrollar un proyecto conlleva unos costes.

En este caso, era necesario poseer un ordenador para programar la aplicación. Como las tecnologías eran libres, no conllevan un coste adicional utilizarlas. Además, es necesario un par de móviles para probar en diferentes dispositivos.

Finalmente, hay que añadir el coste del programador que desarrolla la aplicación, un jefe de proyecto que planifica y gestiona y un analista para el análisis de requisitos y el diseño.

Teniendo en cuenta que un programador puede ganar unos 25€ la hora, calculando una programación aproximada de 6 meses (26 semanas) haciendo 40 horas semanales.

También que un analista puede ganar 40€ la hora y que el trabajo del analista pudo durar 2 semanas haciendo 40 horas semanales.

Finalmente, el jefe de proyecto, que podría cobrar unos 60€ la hora, haciendo 40 horas semanales durante 4 semanas.

Esta es la tabla con los costes aproximados:

Adquisición	Precio
Ordenador	1000€
Móvil (pantalla pequeña, Android versión 2.x)	200€
Móvil (pantalla grande, Android version 4.x)	400€
Jefe de proyecto	9.600€
Analista	3.200€
Programador	26.000€
Total	40.400€

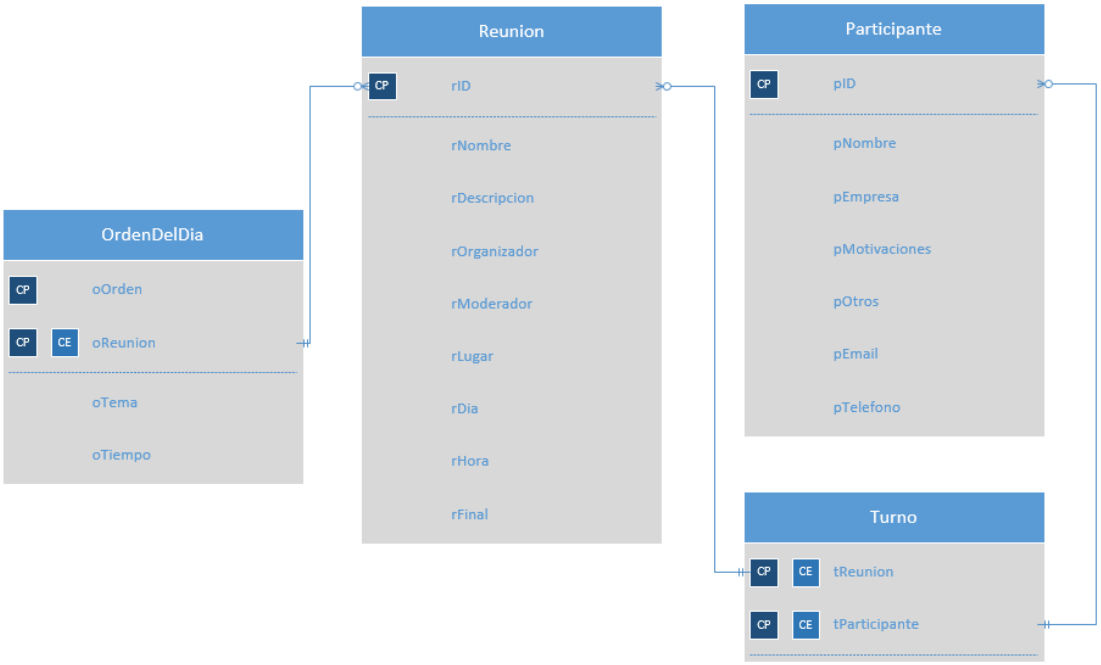
Teniendo en cuenta que esta aplicación se encuentra en el marco de proyecto de final de carrera, no existe coste por el programador ni el jefe de proyecto ni el analista. Además, como ya se posee de un ordenador para programar y de dos móviles que cumplan con las especificaciones para testear, el desarrollo no exige ninguna inversión.

5. Módulos

5.1 Datos de la aplicación

Esta aplicación gestiona numerosos datos que necesitan ser guardados de forma permanente y de forma ordenada. Así pues, resulta inevitable el uso de bases de datos. La tecnología óptima para crear y gestionar bases de datos en Android es SQLite, que es un sistema de gestión de bases de datos liviano y perfectamente integrado en el sistema operativo móvil. Para más información, es conveniente leer el apartado en el cual se explica con detenimiento la tecnología SQLite.

LeAndroid Meeting trabaja con cuatro tablas: Reunion, Participante, OrdenDelDia y Turno.



5.1.1 Reunion

En esta tabla se almacena todos los datos relativos a la reunión, así como un identificador único para cada una. A continuación, se detallan todas sus columnas:

-rID: Es el identificador de cada reunión. Es un valor entero, es único y es incremental, teniendo como primer valor el 1.

-rNombre: Es la cadena de texto en la cual se encuentra el nombre de la reunión. Pueden haber dos reuniones con el mismo nombre y, además, el valor puede ser modificado. El valor no puede ser vacío.

-rDescripcion: Es la cadena de texto que guarda la descripción de la reunión. Puede ser un valor vacío.

-rModerador: Es la columna en la cual se guarda el nombre de la persona que se encargará de moderar la reunión. No guarda relación con los valores de la tabla Participante. Puede ser un valor vacío.

-rOrganizador: En esta cadena de texto se almacena el nombre de la persona que ha organizado la reunión. En esta caso, tampoco existe relación con ninguno de los valores de la tabla Participante y también puede ser información vacía.

-rDia: Este dato contiene la fecha en la cual está programada la reunión. Esta valor sí debe contener información.

-rHora: Esta casilla contiene la hora programada para el inicio de la reunión. Como en el caso de la fecha, este valor tampoco puede ser vacío.

-rFin: Valor booleano en el cual se expresa si una reunión está finalizada o pendiente. Es necesaria para saber si esta reunión debe estar en la lista de actas o en la lista de reuniones pendientes.

5.1.2 Participantes

En esta tabla se almacenan todos los datos relativos a los participantes de las reuniones. Aquí se detallan sus columnas:

-pID: es el identificador de cada participante. El identificador es numérico, es único y es incremental. El primer pID es el valor 1, correspondiente al Moderador. Este usuario, considerado el usuario de aplicación, no se puede borrar pero sí de permite editar todos sus valores, incluido el nombre. A pesar de ser un usuario que no se puede eliminar, no es obligatorio incluirlo en las reuniones.

-pNombre: es la columna en la cual se guarda el nombre del participante. Este valor no puede ser vacío. La aplicación permite varios participantes con el mismo nombre.

-pOrganizacion: cadena de texto en la cual se describe la organización para la que trabaja el participante. No es obligatorio rellenar este dato.

-pEmail: en este campo se indica la dirección de correo electrónico del participante. El valor puede ser vacío.

-pTelefono: en esta columna se indica el número telefónico de un participante de reuniones. No es obligatorio rellenarlo.

-pMotivaciones: cadena de texto en la que se describe qué pretensiones tiene el participante en las reuniones o que objetivos persigue. Tampoco es un campo obligatorio.

5.1.3 OrdenDelDia

En esta tabla se especifican los puntos del día de los que constará una reunión. Esta tabla utiliza referencias de las tablas anteriores (Participante y Reunion):

-oOrden: Este identificador indica el orden en el cual se discutirá el punto del día. Este valor junto con oReunion es la "primary key" de la tabla.

-oReunion: Este valor es el identificador de la reunión. Hace referencia al identificador rID de la tabla Reunion. Este valor junto con oOrden es la "primary key" de la tabla.

-oNombre: Esta cadena de texto contiene el nombre del punto del día. Hace referencia al identificador pID de la tabla Participante. No puede ser vacío y puede repetirse en varias filas.

-oDuración: Este campo detalla los minutos estimados para este punto del día. La información no puede ser vacía.

5.1.4 Turno

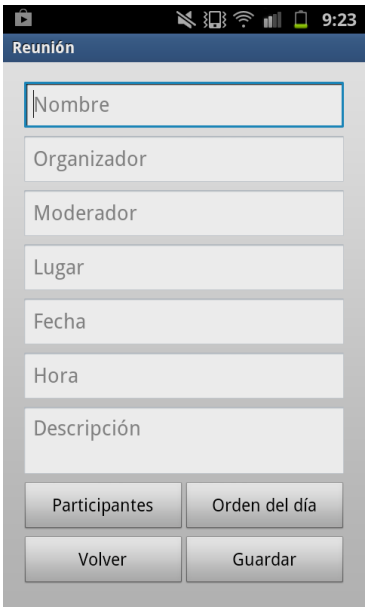
En esta última tabla se encuentra la información de los participantes que estarán en una reunión. Es una tabla "join" de Participante y Reunion. Estos son sus dos campos:

-tReunion: Este campo contiene el identificador de la reunión. Hace referencia al identificador rID de la tabla Reunion. Este valor junto con tParticipante es la "primary key" de la tabla.

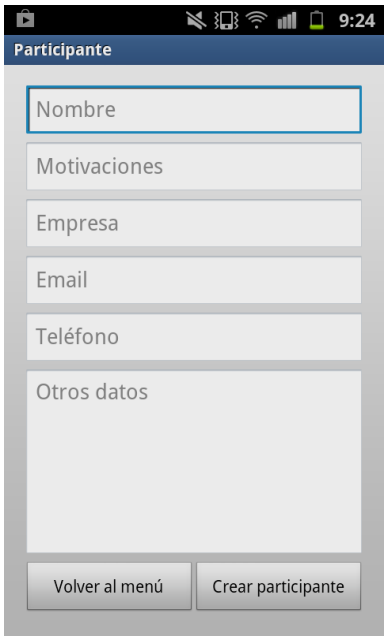
-tParticipante: Esta columna almacena el identificador del participante de la reunión. Hace referencia al identificador pID de la tabla Participante. Este valor junto con tReunion es la "primary key" de la tabla.

Todos estos datos se introducen cómodamente en las siguiente pantallas de la aplicación:

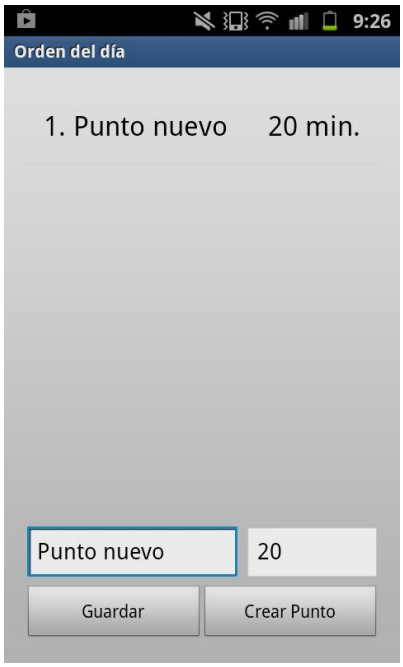
Pantalla para crear reunión:



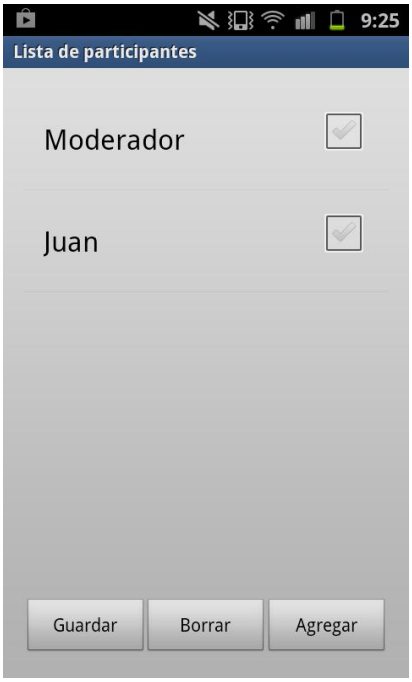
Pantalla para crear participante:



Pantalla para crear orden del día:



Pantalla para crear turno:



5.2 Desarrollo de la reunión

A continuación, se explicará el modulo que gestiona el desarrollo de la reunión y todas sus posibilidades.

Esta es la pantalla principal del módulo:

Cuando se decide comenzar una reunión, entrará en escena la pantalla anteriormente mostrada.

La aplicación muestra el punto del día actual que se procederá a tratar, así como además su duración estimada.

5.2.1 Selección manual

A continuación, la pantalla muestra la lista de participantes de la reunión. Con un sólo click, podemos seleccionar y cambiar el participante que está exponiendo. La lista, además, muestra un cronómetro con el tiempo que lleva utilizado cada participante en este punto en concreto.

En el momento en que quede seleccionado un participante, se habilitará el botón para iniciar el cronómetro. Este botón sirve para activar o desactivar el tiempo del punto del día. Activándolo, se pondrán en marcha dos cronómetros: el propio del participante seleccionado y el general. Ambos cronómetros se utilizan para dar al moderador la orientación de los tiempos

utilizados, de esta forma se le facilitará la gestión de los próximos turnos o del orden del día.

Al seleccionar el participante se habilitarán las tres opciones para recoger actas: tomar nota, grabar audio y transcribir audio.

5.2.2 Tomar nota

En ocasiones, es necesario tomar algunas notas de lo ocurrido o discutido en una reunión. Con el participante seleccionado, al clickar sobre el icono de tomar nota, la aplicación mostrará la pantalla en la cual podremos introducirla.

Para que no rompa el ritmo de la reunión, se implementó como actividad secundaria. Esto se traduce en que, aunque estemos escribiendo una nota para un participante, la actividad en la cual se gestiona la reunión sigue funcionando. Así pues, los cronómetros de la reunión seguirán activos, de modo que no hace falta interrumpir la reunión para tomar notas.

5.2.3 Grabar audio

La aplicación nos da otra opción de recoger un acta. Tomar notas en medio de una reunión puede retrasarla o romper el ritmo, además de ser una tarea tediosa para el moderador.

Por esta razón, es muy interesante este servicio. Al activar el botón de grabación de audio se comenzará a grabar la intervención del participante hasta que se vuelva a pulsar el botón. Cada vez que hagamos una grabación, se generará un archivo de audio que después podremos reproducir en la sección de actas.

5.2.4 Transcripción de audio

La tercera y última opción para recoger un acta es la transcripción de audio.

Al activar el botón, se iniciará la transcripción. Una vez que el participante deje de hablar, la aplicación mostrará una pantalla similar a la de toma de notas, con el resultado de la transcripción. Al usuario le corresponderá la decisión de modificar el resultado, guardarlo o descartarlo.

De nuevo, como en la toma de notas, los cronómetros seguirán activos si estos ya lo estaban.

La aplicación utiliza un servicio de Google, así que es necesario la conexión a internet. Esto es debido a que la aplicación envía a los servidores de Google el audio para que pueda comparar con los modelos de los que disponen. Una vez encuentra las coincidencias, el servidor retorna la respuesta a la aplicación.

Una vez se haya finalizado el punto del día, se clickará la opción "siguiente punto" para pasar al siguiente punto, o "finalizar reunión" para dar por acabada la reunión actual.

Al finalizar la reunión, ésta pasará a la lista de actas. El módulo de actas se describe a continuación.

5.3 Actas

Una vez ha finalizado una reunión, los datos que hemos recogido durante su desarrollo podrán ser consultados.

Para ello, hay que ir a la sección de actas, donde se muestran todas las reuniones que han acabado con éxito. Al seleccionar una reunión se muestra un desplegable con todos los puntos del día. Al seleccionar uno de ellos, se habilitarán las tres opciones para consultar las actas

Podremos consultar las notas. Al seleccionar su correspondiente botón, se mostrará una lista con un archivo por cada uno de los participantes. El archivo de texto puede estar vacío si no se editó durante la reunión. Podremos modificar las notas para corregir errores o mejorar su redacción si no quisimos perder mucho tiempo durante el desarrollo de la reunión

También podremos consultar las grabaciones de audio. Se mostrará una lista con cada uno de los archivos ordenados por fecha de grabación. Se mostrará el nombre del participante. Una vez seleccionado un archivo, podemos reproducir el audio.

Finalmente, se puede consultar las transcripciones de audio. Como en el caso de las grabaciones de audio, se mostrará una lista con cada uno de los archivos ordenados por fecha de transcripción. Se mostrará el nombre del participante. Una vez seleccionado un archivo, podemos acceder a la transcripción. Se podrá modificar la transcripción para corregir algunos errores o modificar algunos datos.

5.4 Interfaz gráfica

Por último, se ha querido conseguir una interfaz gráfica amigable, sencilla e intuitiva. Que sea fácilmente navegable y sin necesidad de aprendizaje, haciendo que se pueda acceder rápidamente a todas sus posibilidades, evitando los diálogos entre la aplicación y el usuario.

La aplicación muestra una interfaz profesional, seria y perfectamente adecuada a la temática de las reuniones de empresa.

Comenzando por la pantalla principal, la cual muestra todas las secciones a través de botones, y muestra la próxima reunión a realizar, con el objetivo de facilitar y agilizar las acciones más probables del usuario.

La aplicación utiliza listas con adaptadores personalizados para mostrar sus datos. Cada lista se adapta para lograr el objetivo de agilidad y facilidad para el usuario. Se ha optado por crear menús contextuales para editar, borrar o realizar otras acciones a los datos de las listas, con el objetivo de evitar sobrecargar las pantallas con botones, adaptándose así perfectamente a los dispositivos móviles, dejando una aplicación muy limpia e intuitiva.

En una primera planificación, se pensó en no permitir navegar por la aplicación con el móvil en horizontal, ya que requería rediseñar todas las pantallas para el nuevo formato, y tampoco resultaba óptimo debido a que muchas pantallas muestran listas verticales. Finalmente, se decidió realizar el esfuerzo para adaptar la aplicación al formato horizontal, teniendo en cuenta que mucha gente suele visualizar sus aplicaciones de esta manera. Para ello, se tuvieron que rediseñar todas las pantallas, haciendo las listas "scrollables" y adaptando los botones al nuevo diseño. Si las listas en formato vertical se llenan demasiado, también aparecerá la barra de "scroll", aunque de esta manera aprovechará mejor el tamaño de la pantalla.

La pantalla más importante es en la que se desarrolla la reunión. Es la única pantalla en la que no se permite el formato horizontal, debido a que se requiere velocidad a la hora de tomar decisiones durante el desarrollo de la reunión y tener la lista de participantes con barra de "scroll" ralentizaría el cambio de participante.

6. Especificación

A continuación, se realizará el análisis de los requisitos funcionales y no funcionales del proyecto.

6.1 Requisitos funcionales

Se mostrarán los casos de uso de las funcionalidades del sistema.

6.1.1 Caso de uso: Añadir participante

· **Descripción:** El usuario añade un nuevo participante al sistema.

· **Precondición:** El participante no está en el sistema.

· **Postcondición:** El participante se crea en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de participantes.
- El sistema muestra la lista de participantes.
- El usuario hace click en la opción de Crear participante.
- El sistema muestra un listado de campos.
- El usuario rellena los datos
- El usuario pulsa la opción Guardar.
- El sistema da de alta al nuevo participante.

6.1.2 Caso de uso: Añadir reunión

· **Descripción:** El usuario añade una nueva reunión al sistema.

· **Precondición:** La reunión no está en el sistema.

· **Postcondición:** La reunión se crea en el sistema.

· **Flujo:**

- El usuario hace click en la opción de Crear reunión.
- El sistema muestra un listado de campos.
- El usuario rellena los datos
- El usuario pulsa la opción Guardar.
- El sistema da de alta a la nueva reunión.

6.1.3 Caso de uso: Añadir punto

· **Descripción:** El usuario añade un nuevo punto al sistema.

· **Precondición:** El punto no está en el sistema.

· **Postcondición:** El punto se crea en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario hace click en la opción de Crear reunión.
- El sistema muestra un listado de campos.
- El usuario pulsa la opción Orden del día.
- El sistema muestra una lista con el orden del día.
- El usuario rellena los datos.
- El usuario pulsa la opción Crear punto.
- El usuario pulsa la opción Guardar.
- El sistema regresa a la actividad anterior.
- El usuario pulsa la opción Guardar.
- El sistema actualiza la reunión.

6.1.4 Caso de uso: Editar participante

· **Descripción:** El usuario edita un participante del sistema.

· **Precondición:** El participante existe en el sistema.

· **Postcondición:** El participante de actualiza en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de participantes.
- El sistema muestra la lista de participantes.
- El usuario abre el menú contextual.
- El usuario pulsa la opción Editar.
- El sistema muestra un listado de campos.
- El usuario modifica los datos
- El usuario pulsa la opción Guardar.
- El sistema da de alta al nuevo participante.

6.1.5 Caso de uso: Editar reunión

· **Descripción:** El usuario edita una reunión del sistema.

· **Precondición:** La reunión existe en el sistema.

· **Postcondición:** La reunión se actualiza en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario abre el menú contextual.
- El usuario pulsa la opción Editar.
- El sistema muestra un listado de campos.
- El usuario actualiza los datos
- El usuario pulsa la opción Guardar.

- El sistema actualiza la reunión.

6.1.6 Caso de uso: Borrar participante

· **Descripción:** El usuario borra un participante del sistema.

· **Precondición:** El participante existe en el sistema.

· **Postcondición:** El participante no existe en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de participantes.
- El sistema muestra la lista de participantes.
- El usuario abre el menú contextual.
- El usuario pulsa la opción Borrar.
- El sistema borra al participante.

6.1.7 Caso de uso: Borrar reunión

· **Descripción:** El usuario borra reunión del sistema.

· **Precondición:** La reunión existe en el sistema.

· **Postcondición:** La reunión no existe en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario abre el menú contextual.
- El usuario pulsa la opción Borrar.
- El sistema borra la reunión

6.1.8 Caso de uso: Borrar punto

· **Descripción:** El usuario borra un punto del sistema.

· **Precondición:** El punto está en el sistema.

· **Postcondición:** El punto no está en el sistema.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario hace click en la opción de Crear reunión.
- El sistema muestra un listado de campos.
- El usuario pulsa la opción Orden del día.
- El sistema muestra una lista con el orden del día.
- El usuario abre el menú contextual.
- El usuario pulsa la opción Borrar.
- El usuario pulsa la opción Guardar.
- El sistema regresa a la actividad anterior.
- El usuario pulsa la opción Guardar.
- El sistema actualiza la reunión.

6.1.9 Caso de uso: Añadir participante a la reunión

· **Descripción:** El usuario añade un participante a una reunión.

· **Precondición:** El participante no asistirá a la reunión.

· **Postcondición:** El participante asistirá a la reunión.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario hace click en la opción de Crear reunión.

- El sistema muestra un listado de campos.
- El usuario pulsa la opción Lista de participantes.
- El sistema muestra una lista con los participantes de la reunión
- El usuario pulsa la opción Añadir participante.
- El sistema muestra una lista con los participantes descartados
- El usuario marca los checkbox de los participantes a añadir
- El usuario pulsa la opción Guardar.
- El sistema muestra una lista con los participantes de la reunión
- El usuario pulsa la opción Guardar.
- El sistema regresa a la actividad anterior.
- El usuario pulsa la opción Guardar.
- El sistema actualiza la reunión.

6.1.10 Caso de uso: Borrar participante de la reunión

· **Descripción:** El usuario borra un participante de una reunión.

· **Precondición:** El participante asistirá a la reunión.

· **Postcondición:** El participante no asistirá a la reunión.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario hace click en la opción de Crear reunión.
- El sistema muestra un listado de campos.
- El usuario pulsa la opción Lista de participantes.
- El sistema muestra una lista con los participantes de la reunión
- El usuario marca los checkbox de los participantes a eliminar
- El usuario pulsa la opción Borrar.

- El usuario pulsa la opción Guardar.

- El sistema regresa a la actividad anterior.

- El usuario pulsa la opción Guardar.

- El sistema actualiza la reunión.

6.1.11 Caso de uso: Copiar al portapapeles los emails

· **Descripción:** El usuario copia al portapapeles las direcciones de los emails de los asistentes a una reunión.

· **Precondición:** La reunión existe en el sistema.

· **Postcondición:** Los emails están copiados en el portapapeles.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.

- El sistema muestra la lista de reuniones.

- El usuario abre el menú contextual.

- El usuario pulsa la opción Copiar al portapapeles los emails.

- El sistema copia al portapapeles los emails.

6.1.12 Caso de uso: Iniciar reunión

· **Descripción:** El usuario inicia el desarrollo de una reunión.

· **Precondición:** La reunión existe en el sistema y no ha finalizado.

· **Postcondición:** -

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.

- El sistema muestra la lista de reuniones.

- El usuario selecciona una reunión

- El usuario selecciona la opción Comenzar reunión.

- El sistema muestra la pantalla de desarrollo de reunión.

6.1.13 Caso de uso: Selección de participante

· **Descripción:** El usuario selecciona a un participante.

· **Precondición:** La reunión existe en el sistema y no ha finalizado.

· **Postcondición:** -

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario selecciona una reunión
- El usuario selecciona la opción Comenzar reunión.
- El sistema muestra la pantalla de desarrollo de reunión.
- El usuario selecciona un participante
- El sistema guarda al participante como seleccionado

6.1.14 Caso de uso: Iniciar grabación

· **Descripción:** El usuario inicia la grabación de un archivo de audio.

· **Precondición:** La reunión existe en el sistema y no ha finalizado y no se está grabando un archivo.

· **Postcondición:** Se está grabando un archivo.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario selecciona una reunión
- El usuario selecciona la opción Comenzar reunión.

- El sistema muestra la pantalla de desarrollo de reunión.

- El usuario selecciona un participante

- El sistema guarda al participante como seleccionado

- El usuario selecciona la opción de grabar.

- El sistema comienza la grabación de un archivo.

6.1.15 Caso de uso: Detener grabación

· **Descripción:** El usuario detiene la grabación de un archivo de audio.

· **Precondición:** La reunión existe en el sistema, no ha finalizado y se está grabando un archivo.

· **Postcondición:** Se ha guardado el archivo grabado.

· **Flujo:**

- El sistema está grabando un archivo

- El usuario selecciona la opción de grabar.

- El sistema detiene la grabación de un archivo y lo guarda en el sistema.

6.1.16 Caso de uso: Transcribir

· **Descripción:** El usuario inicia la transcripción de un archivo de audio.

· **Precondición:** La reunión existe en el sistema y no ha finalizado.

· **Postcondición:** Se ha guardado la transcripción

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.

- El sistema muestra la lista de reuniones.

- El usuario selecciona una reunión

- El usuario selecciona la opción Comenzar reunión.

- El sistema muestra la pantalla de desarrollo de reunión.

- El usuario selecciona un participante
- El sistema guarda al participante como seleccionado
- El usuario selecciona la opción de transcripción.
- El sistema comienza la transcripción de un archivo.
- El usuario habla por el micrófono.
- El sistema muestra el resultado de la transcripción.
- El usuario selecciona la opción Guardar.
- El sistema guarda el archivo.

6.1.17 Caso de uso: Nota

· **Descripción:** El usuario escribe una nota.

· **Precondición:** La reunión existe en el sistema y no ha finalizado.

· **Postcondición:** Se ha guardado la nota

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones.
- El usuario selecciona una reunión
- El usuario selecciona la opción Comenzar reunión.
- El sistema muestra la pantalla de desarrollo de reunión.
- El usuario selecciona un participante
- El sistema guarda al participante como seleccionado
- El usuario selecciona la opción de nota.
- El sistema muestra el campo de texto de la nota
- El usuario escribe la nota.
- El usuario selecciona la opción Guardar.
- El sistema guarda el archivo.

6.1.18 Caso de uso: Visualizar nota

· **Descripción:** El usuario visualiza una nota.

· **Precondición:** La nota existe.

· **Postcondición:** -

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario selecciona la opción Nota
- El sistema muestra el listado de notas
- El usuario selecciona una nota
- El usuario selecciona la opción Leer
- El sistema muestra la nota

6.1.19 Caso de uso: Reproducir grabación

· **Descripción:** El usuario reproduce un archivo de audio

· **Precondición:** La grabación existe y no se está reproduciendo

· **Postcondición:** La grabación se está reproduciendo.

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario selecciona la opción Grabación
- El sistema muestra el listado de grabaciones
- El usuario selecciona una grabación

- El usuario selecciona la opción Reproducir
- El sistema reproduce la grabación

6.1.20 Caso de uso: Detener reproducción grabación

· **Descripción:** El usuario detiene la reproducción de un archivo de audio

· **Precondición:** La grabación existe y se está reproduciendo

· **Postcondición:** La grabación no se está reproduciendo.

· **Flujo:**

- El sistema muestra el listado de grabaciones
- El usuario selecciona la opción Detener
- El sistema detiene la reproducción de la grabación

6.1.21 Caso de uso: Visualizar transcripción

· **Descripción:** El usuario visualiza una transcripción.

· **Precondición:** La transcripción existe.

· **Postcondición:** -

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario selecciona la opción Transcripción
- El sistema muestra el listado de transcripciones
- El usuario selecciona una transcripción
- El usuario selecciona la opción Leer
- El sistema muestra la transcripción

6.1.22 Caso de uso: Borrar nota

· **Descripción:** El usuario borra una nota.

· **Precondición:** La nota existe.

· **Postcondición:** La nota no existe.

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario selecciona la opción Nota
- El sistema muestra el listado de notas
- El usuario abre el menú contextual.
- El sistema muestra las opciones del menú contextual.
- El usuario selecciona la opción Borrar archivo.
- El sistema borra el archivo.

6.1.23 Caso de uso: Borrar grabación

· **Descripción:** El usuario borra un archivo de grabación de audio.

· **Precondición:** La grabación existe.

· **Postcondición:** La grabación no existe.

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario selecciona la opción Grabación.
- El sistema muestra el listado de grabaciones.

- El usuario abre el menú contextual.
- El sistema muestra las opciones del menú contextual.
- El usuario selecciona la opción Borrar archivo.
- El sistema borra el archivo.

6.1.24 Caso de uso: Borrar transcripción

· **Descripción:** El usuario borra una transcripción.

· **Precondición:** La transcripción existe.

· **Postcondición:** La transcripción no existe.

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario selecciona la opción Transcripción
- El sistema muestra el listado de transcripciones
- El usuario abre el menú contextual.
- El sistema muestra las opciones del menú contextual.
- El usuario selecciona la opción Borrar archivo.
- El sistema borra el archivo.

6.1.25 Caso de uso: Borrar acta

· **Descripción:** El usuario borra una acta completa de una reunión.

· **Precondición:** La reunión ha finalizado.

· **Postcondición:** La reunión no tiene acta.

· **Flujo:**

- El usuario accede a la actividad de Actas.

- El sistema muestra la lista de reuniones acabadas.
- El usuario selecciona una reunión y un punto.
- El usuario abre el menú contextual.
- El sistema muestra las opciones del menú contextual.
- El usuario selecciona la opción Borrar acta.
- El sistema borra el acta.

6.1.26 Caso de uso: Reanudar reunión

· **Descripción:** El usuario reanuda una reunión ya finalizada.

· **Precondición:** La reunión está finalizada.

· **Postcondición:** La reunión no está finalizada.

· **Flujo:**

- El usuario accede a la actividad de Actas.
- El sistema muestra la lista de reuniones acabadas.
- El usuario abre el menú contextual.
- El sistema muestra las opciones del menú contextual.
- El usuario selecciona la opción Reanudar reunión.
- El sistema pasa la reunión a la lista de no finalizadas.

6.1.27 Caso de uso: Finalizar reunión

· **Descripción:** El usuario finaliza una reunión.

· **Precondición:** La reunión no está finalizada.

· **Postcondición:** La reunión está finalizada.

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.
- El sistema muestra la lista de reuniones no acabadas.

- El usuario abre el menú contextual.

- El sistema muestra las opciones del menú contextual.

- El usuario selecciona la opción Finalizar reunión.

- El sistema pasa la reunión a la lista de finalizadas.

6.1.28 Caso de uso: Siguiente punto

· **Descripción:** El usuario avanza al siguiente punto de la reunión

· **Precondición:** La reunión existe en el sistema y no ha finalizado.

· **Postcondición:** -

· **Flujo:**

- El usuario accede a la actividad de Lista de reuniones.

- El sistema muestra la lista de reuniones.

- El usuario selecciona una reunión.

- El usuario selecciona la opción Comenzar reunión.

- El sistema muestra la pantalla de desarrollo de reunión.

- El usuario selecciona la opción Siguiente punto

- El sistema avanza al siguiente punto de la reunión.

6.2 Requisitos no funcionales

Los requisitos no funcionales de esta aplicación son los siguientes:

- Mantenibilidad: La arquitectura en tres capas garantiza que el mantenimiento será sencillo de realizar.

- Escalabilidad: Por la misma razón anterior, mejorar e implementar nuevas funcionalidades será sencillo de realizar.

- Usabilidad: La aplicación es realmente fácil de utilizar gracias a la intuitiva interfaz gráfica.

- Rendimiento: Aunque las tres capas no van a favor de la eficiencia, se ha trabajado para maximizar el rendimiento de Android.

- Estabilidad: Se ha conseguido una aplicación estable cuyas funcionalidades están muy testeadas.

- Interfaz: El programa muestra una interfaz intuitiva, amigable y profesional.

Hay muchos más requisitos no funcionales en la aplicación, pero estos son los principales que se han tenido en cuenta.

7. Diseño

7.1 Arquitectura en tres capas

Para esta aplicación se ha decidido utilizar la arquitectura en tres capas. Esto significa que se diferenciarán las clases en tres tipos: gestión de datos, la capa de dominio (también llamada capa lógica o de negocio) y la capa de presentación.

Esta decisión viene dada a que es el diseño ideal para hacer una aplicación modificable y altamente escalable.

La capa de gestión de datos es aquella en que sus clases se encargan de tratar directamente con las bases de datos. Tienen operaciones de añadido, modificado, eliminado o consulta de los datos de las bases de datos de la aplicación.

La capa de dominio es aquella en que sus clases se encargan de atender a peticiones, procesar datos, comunicarse con la capa de gestión de datos y, finalmente, retornar resultados a la capa de presentación.

La capa de presentación es aquella en que sus clases se comunican con el usuario, atendiendo a sus peticiones y capturando eventos para realizar peticiones posteriormente a la capa de dominio.

7.2 Otras consideraciones de diseño

Además de la arquitectura en tres capas, hay otras consideraciones con el diseño de la aplicación.

Se ha utilizado el paradigma de programación orientada a objetos. Además, Java es un lenguaje de programación con este paradigma. Las clases poseen identidad, un estado y un comportamiento.

También hay que hacer una consideración acerca de la arquitectura en tres capas. Hay algunas acciones que se hacen en la capa de presentación en lugar de la de dominio. Esto es debido a que, por eficiencia en *Android*, es más óptimo realizarlas en estas clases.

8. Implementación

En este capítulo se explicarán en profundidad todas las clases de la aplicación. Se comenzará haciendo una explicación general sobe su funcionamiento y sus principales objetivos. Posteriormente, se comentarán sus atributos. Finalmente, se explicarán sus operaciones.

8.1 Clases de gestión de datos

A continuación, se explicarán las clases de gestión de datos. Estas clases son las que trabajan directamente con las bases de datos en las que están guardadas las estructuras de datos de la aplicación.

8.1.1 GestorSQLite

Explicación general

Esta clase es la que se encarga de gestionar las tablas de SQLite. Tiene como objetivo principal la creación de las tablas y la restauración de las bases de datos.

Atributos

- nombreBD: Contiene el nombre de la base de datos.

- Participante: El nombre de la tabla Participante.

- pID: Contiene el nombre de la columna del identificador de participante.

- pEmpresa: Posee el nombre de la columna de la empresa del participante.

- pMotivaciones: Contiene el nombre de la columna de motivaciones del participante.

- pOtros: El nombre de la columna de otros datos del participante.

- pEmail: Tiene el nombre de la columna de dirección de email del participante.

- pTelefono: Aquí está el nombre de la columna de teléfono del participante.

- Turno: Contiene el nombre de la tabla Turno.

- tReunion: Contiene el nombre de la columna del identificador de reunión de la tabla Turno.

- tParticipante: Es el nombre de la columna del identificador de participante de la tabla Turno.

- Reunion: Es el nombre de la tabla Reunion.

- rID: El nombre de la columna del identificador de la reunión.

- rNombre: Contiene el nombre de la columna del nombre de la reunión.

- rDescripcion: Posee el nombre de la columna de la descripción de la reunión.

- rOrganizador: Contiene el nombre de la columna del organizador de la reunión.

- rModerador: El nombre de la columna del moderador de la reunión.

- rLugar: Contiene nombre de la columna del lugar de la reunión.

- rDia: Es el nombre de la columna de la fecha de la reunión.

- rHora: El nombre de la columna de la hora de la reunión.

- rFinal: Contiene el nombre de la columna de reunión finalizada.

- OrdenDelDia: Es el nombre de la tabla OrdenDelDia.

- oTema: Contiene el nombre de la columna tema de un punto del orden del día.

- oReunion: Contiene el nombre de la columna del identificador de la reunión de un punto del orden del día.

- oOrden: Es el nombre de la columna del orden de un punto del orden del día.

- oTiempo: Contiene el nombre de la columna del tiempo estimado de un punto del orden del día.

Métodos:

- GestorSQLite(): Es la función creadora que, recibiendo como parámetro el contexto de la aplicación, llama a la función de la clase padre *SQLiteOpenHelpe*r para trabajar con la base de datos *nombreBD.*

- onCreate(): Es la acción que, recibiendo la base de datos, crear las tablas con los nombres y columnas de los atributos de la operación.

- onUpgrade: Es la acción que, recibiendo la base de datos, borra las tablas y las vuelve a generar llamando a la acción *onCreate()*.

8.1.2 GestorReunion

Explicación general

Esta clase es la que se encarga de gestionar la tabla Reunion, realizando operaciones de añadido, borrado, modificación y consulta.

Atributos

- Reunion: Es el nombre de la tabla Reunion.

- rID: El nombre de la columna del identificador de la reunión.

- rNombre: Contiene el nombre de la columna del nombre de la reunión.

- rDescripcion: Posee el nombre de la columna de la descripción de la reunión.

- rOrganizador: Contiene el nombre de la columna del organizador de la reunión.

- rModerador: El nombre de la columna del moderador de la reunión.

- rLugar: Contiene nombre de la columna del lugar de la reunión.

- rDia: Es el nombre de la columna de la fecha de la reunión.

- rHora: El nombre de la columna de la hora de la reunión.

- rFinal: Contiene el nombre de la columna de reunión finalizada.

Métodos:

- anadirReunion(): Es la función que, recibiendo como parámetros la base de datos y los valores de la reunión, crea una nueva reunión en la tabla Reunion. Si la reunión no es nueva, actualiza la fila. La función retorna el identificador de la reunión añadida o modificada.

- eliminarReunion(): Es la acción que, recibiendo como parámetros la base de datos y el identificador de la reunión, elimina la fila de la tabla Reunion.

- getReunion(): Es la función que, recibiendo como parámetros la base de datos y el identificador de la reunión, devuelve la fila en forma de cursor iterable.

- getAllReunion(): Es la función que, recibiendo como parámetro la base de datos, devuelve todas las reuniones en forma de cursor iterable.

8.1.3 GestorParticipante

Explicación general

Esta clase se encarga de gestionar la tabla Participante de la base de datos, haciendo operaciones de añadido, borrado, modificación y consulta.

Atributos

- Participante: El nombre de la tabla Participante.

- pID: Contiene el nombre de la columna del identificador de participante.

- pEmpresa: Posee el nombre de la columna de la empresa del participante.

- pMotivaciones: Contiene el nombre de la columna de motivaciones del participante.

- pOtros: El nombre de la columna de otros datos del participante.

- pEmail: Tiene el nombre de la columna de dirección de email del participante.

- pTeléfono: Aquí está el nombre de la columna de teléfono del participante.

Métodos

- anadirParticipante(): Esta función se encarga de, recibiendo como parámetros la base de datos y los valores del participante, añadir una nueva fila a la tabla Participante o modificarla si ya existe. La función retorna el identificador del participante nuevo o modificado.

- eliminarParticipante(): Es la acción que, recibiendo como parámetros la base de datos y el identificador del participante, elimina la fila de la tabla Participante.

- getParticipante(): Es la función que, recibiendo como parámetros la base de datos y el identificador del participante, devuelve la fila en forma de cursor iterable.

- getAllParticipante(): Es la función que, recibiendo como parámetro la base de datos, devuelve todos los participantes en forma de cursor iterable.

8.1.4 GestorOrdenDelDia

Explicación general

Esta clase es la que se encarga de gestionar la tabla OrdenDelDia, haciendo operaciones de añadido, borrado, modificación y consulta.

Atributos

- OrdenDelDia: Es el nombre de la tabla OrdenDelDia.

- oTema: Contiene el nombre de la columna tema de un punto del orden del día.

- oReunion: Contiene el nombre de la columna del identificador de la reunión de un punto del orden del día.

- oOrden: Es el nombre de la columna del orden de un punto del orden del día.

- oTiempo: Contiene el nombre de la columna del tiempo estimado de un punto del orden del día.

Métodos:

- anadirOrdenDelDia(): Esta acción se encarga de, recibiendo como parámetros la base de datos y los valores del punto de la orden del día, añadir una nueva fila a la tabla OrdenDelDia.

- eliminarOrdenDelDia(): Es la acción que, recibiendo como parámetros la base de datos, el orden del punto y el identificador de la reunión, elimina la fila de la tabla OrdenDelDia.

- getOrdenDelDia(): Es la función que, recibiendo como parámetros la base de datos, el identificador de la reunión y el orden del punto, devuelve la fila en forma de cursor iterable.

- getOrdenDelDiaReunion(): Es la función que, recibiendo como parámetros la base de datos y el identificador de la reunión, devuelve todo el orden del día de la reunión en forma de cursor iterable.

- getMaxOrden(): Es la función que, recibiendo como parámetros la base de datos y el identificador de la reunión, devuelve el orden máximo de un punto.

8.1.5 GestorTurno

Explicación general

Esta clase es la que se encarga de gestionar las tablas de SQLite. Tiene como objetivo principal la creación de las tablas y la restauración de las bases de datos.

Atributos

- Turno: Contiene el nombre de la tabla Turno.

- tReunion: Contiene el nombre de la columna del identificador de reunión de la tabla Turno.

- tParticipante: Es el nombre de la columna del identificador de participante de la tabla Turno.

Métodos:

- anadirTurno(): Esta acción se encarga de, recibiendo como parámetros la base de datos y los valores del turno, añadir una nueva fila a la tabla Turno.

- eliminarTurno (): Es la acción que, recibiendo como parámetros la base de datos y el identificador de la reunión, elimina las filas de la tabla Turno.

- getTurno (): Es la función que, recibiendo como parámetros la base de datos, el identificador de la reunión y el identificador del participante, devuelve la fila en forma de cursor iterable.

- getTurnoReunion(): Es la función que, recibiendo como parámetros la base de datos y el identificador de la reunión, devuelve todos los turnos de la reunión en forma de cursor iterable.

- eliminarTurnoParticipante (): Es la acción que, recibiendo como parámetros la base de datos y el identificador del participante, elimina las filas de la tabla Turno.

8.2 Clases de dominio

A continuación, se explicarán las clases de dominio. Estas son clases en las que se reciben peticiones, se procesan, se hacen peticiones a las clases de gestión de datos y se generan resultados.

8.2.1 Acta

Explicación general

Esta clase se encarga de generar los nombres de los archivos de las actas que se desean borrar y reordenar los archivos para seleccionarlos por tiempo.

Atributos

Esta clase no tiene atributos.

Métodos:

- borrarActaReunion(): Esta acción se encarga de, recibiendo como parámetros el contexto de la aplicación y el identificador de la reunión, eliminar todos los ficheros de acta de esta reunión.

- borrarActaParticipante(): Esta acción se encarga de, recibiendo como parámetros el contexto de la aplicación y el identificador de participante, eliminar todos los ficheros de acta de este participante.

- borrarOrdenDelDia(): Esta acción se encarga de, recibiendo como parámetros el contexto de la aplicación, el identificador de reunión y el orden de un punto del orden del día , eliminar todos los ficheros de acta de este punto.

- borrarActaNota(): Esta acción se encarga de, recibiendo como parámetros el contexto de la aplicación y el nombre de un archivo de notas , eliminar ese fichero.

- borrarActaFile(): Esta acción se encarga de, recibiendo como parámetros el contexto de la aplicación y el nombre de un archivo de grabación o de transcripción , eliminar ese fichero.

- getParticipante(): Esta función se encarga de, recibiendo como parámetro el nombre del archivo, devolver el identificador del participante.

- ordenar(): Esta función, al recibir la lista de ficheros de grabación o de transcripción, devuelve la lista ordenada por tiempo.

8.2.2 Cronómetro

Explicación general

Esta clase se encarga de gestionar la lógica de los cronómetros de la aplicación.

Atributos

- crono: Es el cronómetro de la clase.

- time: Es el tiempo acumulado por el cronómetro *crono*.

- on: Es el valor booleano que indica si el cronómetro *crono* está activo.

Métodos:

- Cronometro (): Es la función creadora que inicializa el cronómetro *crono*, el tiempo total y el booleano *on*.

- start(): Esta acción inicia el cronómetro *crono* y actualiza el valor booleano *on*.

- stop(): Esta acción detiene el cronómetro *crono* y actualiza los valores *time* y *on*.

- reset(): Esta acción restablece el tiempo acumulado a *0* e inicializa de nuevo el cronómetro *crono*.

- getTime(): Esta función devuelve el tiempo acumulado *time*.

- getTimeActual(): Esta función devuelve el tiempo que marca actualmente el cronómetro *crono*.

- getSeconds(): Esta función retorna el tiempo acumulado en segundos.

- getCronometer(): Esta función retorna el cronómetro *crono*.

8.2.3 Fecha

Explicación general

Esta clase contiene las operaciones necesarias para preparar las fechas de las reuniones para compararlas y ordenarlas.

Atributos

- reunion: Es la reunión de la cual queremos obtener la fecha en formato comparable.

- dateTime: Es la fecha de la reunión en formato *Date*, ideal para compararla y ordenarla.

- formatter: Indica el formato en el cual será introducida la fecha.

Métodos:

- Cronometro (): Es la función creadora que, obteniendo como parámetro la reunión, inicializa los atributos de la clase.

- getDateTime(): Esta función retorna el valor *dateTime*.

- getReunion(): Esta función devuelve la reunión de la clase.

- getTiempoActual(): Esta función retorna el tiempo que marca el sistema operativo.

- compareTo(): Esta función, obteniendo como parámetro una fecha, retorna el resultado de la comparación con la fecha de la clase.

- ordenar(): Esta acción ordena recursivamente una lista obtenida como parámetro.

8.2.4 Grupo

Explicación general

Esta clase tiene como misión crear una estructura de padres e hijos para las listas expandibles.

Atributos

- string: Contiene la cadena de texto que hará la función de padre en las listas.

- hijos: Contiene una lista de cadenas de texto con los valores que harán de hijos en las lista.

Métodos:

- Grupo(): es la función creadora que, obteniendo el valor del padre, inicializa el valor de *string*.

8.2.5 OrdenDelDia

Explicación general

Esta clase contiene la lógica para tratar el orden del día de las reuniones.

Atributos

- oTema: Contiene el tema del punto del orden del día.

- oReunión: Contiene el identificador de la reunión a la cual referencia el punto del orden del día.

- oOrden: Es la posición que ocupa el punto en el orden del día.

- oTiempo: Es el tiempo estimado que durará el punto del orden del día.

Métodos:

- OrdenDelDia(): es la función creadora que, recibiendo como parámetros la base de datos, el tema y el identificador de la reunión, obtiene el punto del orden del día llamando a la función *getOrdenDelDia().*

- OrdenDelDia(): es otra función creadora que, recibiendo como parámetros la base de datos y el resto de valores de un punto, rellena los atributos de la clase.

- getOrdenDelDia(): es una acción que, recibiendo como parámetros la base de datos, el tema y el identificador de la reunión, obtiene el punto a través de la clase *GestorOrdenDelDia* y rellena los atributos.

- isUniqueOrdenDelDia(): Comprueba si es único el punto de la orden del día y retorna la respuesta.

- get_oTema(): Consultora del atributo *oTema*.

- get_oReunion(): Consultora del atributo *oReunion*.

- get_oOrden(): Consultora del atributo *oOrden*.

- get_oTiempo(): Consultora del atributo *oTiempo*.

- set_oOrden(): Modificadora del atributo *oOrden*.

- anadirOrdenDelDia(): Esta acción llama al método de añadir el punto a la tabla de orden del día a través del gestor, recibiendo como parámetros la base de datos y todos los campos necesarios para la creación.

- eliminarOrdenDelDia(): Esta acción llama al método de eliminar el orden del día de una reunión, recibiendo como parámetros la base de datos y el identificador de la reunión.

- eliminarPuntoDelDia(): Esta acción llama al método de eliminar el punto de la tabla de orden del día a través del gestor, recibiendo como parámetros la base de datos y todos los campos necesarios para su eliminacion.

- getOrdenDelDiaReunion(): Esta función que, recibe como parámetros la base de datos y el identificador de la reunión, retorna la lista con los puntos del orden del día de la reunión.

- recalcularOrden(): Esta acción, recibiendo como parámetros la lista con todos los puntos del orden del día y el índice del que queremos eliminar, recalcula la posición de cada uno de los puntos y los modifica.

- getMaxOrden(): Esta función, recibiendo como parámetros la base de datos y el identificador de la reunión, llama al gestor para obtener la posición máxima en el orden del día.

8.2.6 Participante

Explicación general

Esta clase contiene la lógica para tratar a los participantes de las reuniones.

Atributos

- pID: Contiene el identificador del participante.

- pNombre: Contiene el nombre del participante.

- pEmpresa: Contiene el nombre de la empresa en la cual trabaja el participante.

- pMotivaciones: Cadena de texto que contiene las motivaciones del participante.

- pEmail: Es la dirección de correo electrónico del participante.

- pTelefono: Es el número de teléfono del participante.

- pOtros. Es la cadena de texto en la cual se expresan otros datos de interés del participante.

Métodos:

- Participante (): es la función creadora que, recibiendo como parámetros la base de datos y el identificador del participante, obtiene el participante llamando a la función *getParticipante()*.

- getParticipante(): es una acción que, recibiendo como parámetros la base de datos y el identificador del participante, obtiene el participante a través de la clase *GestorParticipante* y rellena los atributos.

- isUniqueOrdenDelDia(): Comprueba si es único el punto de la orden del día y retorna la respuesta.

- get_pID(): Consultora del atributo *pID*.

- get_pNombre(): Consultora del atributo *pNombre*.

- get_pEmpresa(): Consultora del atributo *pEmpresa*.

- get_pMotivaciones(): Consultora del atributo *pMotivaciones*.

- get_pEmail(): Consultora del atributo *pEmail*.

- get_pTelefono(): Consultora del atributo *pTelefono*.

- get_pOtros(): Consultora del atributo *pOtros*.

- anadirParticipante(): Esta acción llama al método de añadir el participante a través del gestor, recibiendo como parámetros la base de datos y todos los campos necesarios para la creación.

- eliminarParticipante(): Esta acción llama al método de eliminar el participante, recibiendo como parámetros la base de datos y el identificador del participante.

- getAll_pNombre(): Esta función, recibiendo como único parámetro la base de datos, retorna una lista con todos los nombres de los participantes a través del gestor.

- getAll_pID(): Esta función, recibiendo como único parámetro la base de datos, retorna una lista con todos los identificadores de los participantes a través del gestor.

- getListaParticipantesID(): Esta función, recibiendo como parámetros la base de datos y el identificador de la reunión, retorna una lista con todos los identificadores de los participantes a la reunión a través del gestor de turnos.

- getListaParticipantesNombre(): Esta función, recibiendo como parámetros la base de datos y el identificador de la reunión, retorna una lista con todos los nombres de los participantes a la reunión a través del gestor de turnos.

- getListaParticipantesReunion(): Esta función, recibiendo como parámetros la base de datos y el identificador de la reunión, retorna una lista con todos los participantes a la reunión a través del gestor de turnos.

8.2.7 Reunion

Explicación general

Esta clase contiene la lógica para añadir, modificar, consultar o eliminar reuniones.

Atributos

- rID: Contiene el identificador de la reunión.

- rNombre: Contiene el nombre de la reunión.

- rDescripcion: Contiene la descripción de la reunión.

- rOrganizador: Es el nombre del organizador de la reunión.

- rModerador: Es el nombre del moderador de la reunión.

- rLugar: Es el lugar en el que se desarrollará la reunión.

- rDia: Contiene la fecha para la cual está prevista la reunión.

- rHora: Contiene la hora prevista de la reunión.

- rFinal: Valor que indica si está finalizada la reunión.

Métodos:

- Reunion(): Es la función creadora que, recibiendo como parámetros la base de datos y el identificador de la reunión, rellena los atributos de la clase llamando al método *getReunion()*.

- Reunion(): Es otra función creadora que, recibiendo como parámetros la base de datos y un cursor iterable con los valores de la reunión, rellena los atributos llamando a la operación *rellenaReunionBD()*.

- Reunion(): Es una nueva función creadora que, recibiendo como parámetros todos los valores de una reunión, rellena los atributos llamando a la operación *rellenaReunionBD()*.

- rellenaReunion(): Es una acción que recibe los valores de una reunión y se los asigna a los atributos de la clase.

- getReunion():Esta acción, que recibe como parámetros la base de datos y el identificador de la reunión, consigue sus valores a través del gestor y rellena los atributos de la clase.

- rellenaReunionBD(): Esta acción, que recibe como parámetros la base de datos y el cursor iterable con los valores de una reunión, rellena los atributos de la clase.

- get_rID(): Consultora del atributo *rID*.

- get_rNombre(): Consultora del atributo *rNombre*.

- get_rDescripcion(): Consultora del atributo *rDescripcion*.

- get_rOrganizador(): Consultora del atributo *rOrganizador*.

- get_rActa(): Consultora del atributo *rActa*.

- get_rModerador(): Consultora del atributo *rModerador*.

- get_rLugar(): Consultora del atributo *rLugar*.

- get_rDia(): Consultora del atributo *rDia*.

- get_rHora(): Consultora del atributo *rHora*.

- get_rFinal(): Consultora del atributo *rFinal.*

- set_rID(): Modificadora del atributo rID.

- setAcabada(): Acción que, recibiendo como parámetros la base de datos y un booleano, modifica la reunión actualizando su valor *rFinal*. Utiliza el gestor de reuniones.

- getAcabada(): Función que retorna si está finalizada la reunión en forma de booleano.

- anadirReunion: Función que, recibiendo como parámetros el contexto de la aplicación, la base de datos y todos los valores de la reunión, crea la nueva reunión o la modifica si ya existía. A través de los gestores, la función crea la reunión, el orden del día y los turnos.

- eliminarReunion(): Acción que recibe como parámetros el contexto de la aplicación, la base de datos y el identificador de la reunión y, a través de los gestores, elimina la reunión y el orden del día y los turnos asociados.

- getAll_rNombre(): Función que, dada una base de datos, retorna una lista con todos los nombres de reuniones de la aplicación.

- getAll_rID(): Función que, dada una base de datos, retorna una lista con todos los identificadores de reuniones de la aplicación.

- getCreated(): Función que, recibiendo una base de datos y una columna de la tabla, retorna el identificador de la reunión.

- get_ReunionById: Función que, recibiendo como parámetros una base de datos y un identificador de reunión, obtiene del gestor una reunión y la devuelve.

8.2.8 ReunionPlay

Explicación general

Esta clase contiene las operaciones para el funcionamiento del desarrollo de la reunión.

Atributos

- timeList: Es una lista con los tiempos acumulados de todos los participantes.

- temaActual: Contiene el identificador del tema actual del orden del día.

- lp: Es la lista de asistentes a la reunión.

- odd: Una lista que contiene el orden del día.

- reunion: Es la reunión actual.

- bd: Es la base de datos.

Métodos:

- ReunionPlay(): Es la función creadora que inicializa todos los atributos de la clase.

- getTemaActual(): Consultora del atributo *temaActual*.

- getTimeByName(): Función que recibe como parámetro el nombre de un participante y ésta devuelve el tiempo acumulado del mismo.

- getTimeTemaActual(): Función que retorna el tiempo estimado del actual tema del orden del día.

- getTiempoMedio(): Función que retorna el tiempo asignado a cada participante.

- timeRestante(): Función que retorna el tiempo restante del actual punto.

- esUltimoTema(): Función que retorna en forma de booleano si el actual tema es el último en el orden del día.

- nuevoTema(): Acción que avanza el tema actual y resetea los cronómetros de los participantes.

- getParticipantesNombres(): Función que retorna una lista con los nombres de los participantes.

- getParticipantesID(): Función que retorna una lista con los identificadores de los participantes.

- finishReunion(): Acción que finaliza una reunión.

- getReunion(): Consultora del atributo *reunion*.

8.2.9 Turno

Explicación general

Esta clase contiene la lógica para el funcionamiento de los turnos de una reunión.

Atributos

- tReunion: Es el identificador de la reunión a la cual hace referencia.

- tParticipante: Es el identificador del participante al cual hace referencia.

Métodos:

- Turno(): Es la función creadora que, recibiendo como parámetros la base de datos y los identificadores de reunión y de participante, obtiene el turno a través de la operación *getTurno()*.

- getTurno(): Es una acción que recibe una base de datos y los identificadores de reunión y de participante y, usando su gestor, obtiene el turno de la base de datos y rellena los atributos de la clase.

- get_tReunion(): Es la consultora del atributo *tReunion*.

- get_tParticipante(): Es la consultora del atributo *tParticipante*.

- anadirTurno(): Es la operación que, recibiendo como parámetros la base de datos y los identificadores de una reunión y un participante y usando su gestor, crea un nuevo turno en su tabla.

- eliminarTurno(): Esta acción, dadas una base de datos y un identificador de reunión, a través de su gestor elimina el turno de la reunión.

8.3 Clases de presentación

A continuación se explicarán las clases de presentación. Estas clases son conocidas en *Android* como *activities*. Las vistas están diseñadas con archivos *XML* mientras que en las *activities* se implementa la lógica que harán que se todos los elementos se comporten como es necesario. Por eficiencia en Android, algunas funcionalidades están programadas directamente en las *activities*, para evitar complicaciones con los flujos de datos y no tener que enviar continuamente los contextos de la aplicación o las *activities* como parámetro en la capa de dominio.

La función principal de todas las clases es *onCreate()*, que es la que se ejecuta nada más iniciar la *activity*. Así pues, todos los eventos estarán programados en esta operación.

8.3.1 ActaActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_acta*. Es la clase en la que se muestran las notas de las actas.

Atributos

- files: Es una lista que contiene los nombres de los archivos de notas de las actas.

- users: Es una lista que contiene los nombres de los participantes de la reunión.

- lista: Es la vista de la lista de los archivos de notas.

- bd: Es la base de datos.

- adaptador: Es el adaptador personalizado de la *ListView* que contiene los archivos de las notas.

- atras: Es el botón para volver a la *activity* anterior.

- leer: Es el botón para leer una nota seleccionada.

- seleccionado: Contiene el valor de la selección actual de la lista.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene todos los nombres de los ficheros de la *activity* anterior. Después, prepara los eventos para los botones y la lista. Al seleccionar una opción en la lista se modifica el marcador y se cambia el color se la opción seleccionada. Al botón leer se le prepara para comenzar la actividad *NotasActivity* cuando se haga click a su opción.

- onCreateContextMenu(): Se crea el menú contextual con la opción de borrar archivo.

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.2 ActaReproducirActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_acta_reproducir*. Es la clase en la que se muestran las listas de todas las reuniones acabadas y sus órdenes del día para visualizar las actas.

Atributos

- grupos: Es una lista de *Grupo* en la cual se encuentran los valores de la lista expandible.

- lista: Es la vista de la lista expandible, es decir, un listado de opciones con otros listados desplegables.

- notas: Es el botón de consultar notas.

- grabaciones: Es el botón de consultar las grabaciones.

- transcripciones: Es el botón de consultar las transcripciones.

- atras: Es el botón para volver a la *activity* anterior.

- padre: Es la opción actual seleccionada de los padres de la lista.

- hijo: Es la opción actual seleccionada de los hijos de la lista.

- lr: Es la lista de reuniones finalizadas.

- bd: Es la base de datos.

- adapter: Es el adaptador personalizado de la *ExpandableListView*.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene todos los nombres de las reuniones finalizadas y las ordena por fecha. También, prepara los datos que irán en la lista. Después, prepara los eventos para los botones y la lista. Al seleccionar una opción en la lista se modifica el marcador y se cambia el color se la opción seleccionada. A cada botón se le asigna la operación adecuada que describiremos a continuación.

- notas(): Acción que prepara los archivos y los envía a la actividad secundaria que lanza (*ActaActivity*).

- grabaciones(): Esta acción prepara los archivos, buscándolos aplicando expresiones regulares, e inicia *GrabacionesActivity* como actividad secundaria.

- transcripciones(): La acción prepara los archivos, buscándolos aplicando expresiones regulares, e inicia *TranscripcionesActivity* como actividad secundaria.

- rellenarDatos(): Esta acción rellena los datos del atributo *grupos* que después se usarán en la lista.

- setChild(): Modificadora del atributo *hijo*.

- setGroup(): Modificadora del atributo *padre*.

- onCreateContextMenu(): Se crea el menú contextual con las opciones de editar reunión, borrar reunión, copiar emails al portapapeles, borrar acta y reanudar reunión.

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

- activar(): habilita los tres botones de acta: notas, grabaciones y transcripciones.

- notificar(): Acción que notifica al adaptador que los datos han sido modificados y que hay que volver a generar la lista.

- getReunion(): Consultora del atributo *padre*.

- getPunto(): Consultora del atributo *hijo*.

8.3.3 GrabacionesActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_grabaciones*. Es la clase en la que se muestran las listas de los archivos de grabaciones de una reunión y un punto concreto del orden del día.

Atributos

- files: Es una lista de nombres de los archivos de grabación de audio pertenecientes al punto seleccionado del orden del día.

- users: Es la lista con los nombres de los participantes de la reunión.

- lista: Es la vista de la lista de los archivos de grabación de audio.

- reproducir: Es el botón de reproducir grabación.

- atras: Es el botón para volver a la *activity* anterior.

- reproduciendo: Es el booleano que indica si algún archivo está en reproducción.

- key: Es la opción actual seleccionada de la lista.

- mediaPlayer: Es la clase con la que se reproducirá el archivo de audio.

- context: Es el contexto de la actividad.

- adaptador: Es el adaptador personalizado de la *ListView*.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene todos los nombres de los ficheros de la *activity* anterior. Después, prepara los eventos para los botones y la lista. Al seleccionar una opción en la lista se modifica el marcador y se cambia el color se la opción seleccionada. Al botón *reproducir* se le prepara para comenzar a reproducir un archivo en cuanto se haga click. Una vez esté un archivo en reproducción, el botón click cambiará su imagen y su funcionalidad por la de detener reproducción.

- onCreateContextMenu(): Se crea el menú contextual con la opción de borrar archivo

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

- volverActividadPrincipal(): Al haberse iniciado la *activity* como actividad secundaria, al regresar a la actividad anterior se utilizará este método.

8.3.4 ListaParticipantesActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_lista_participantes*. Es la clase en la que se muestran la lista de participantes en el sistema. También permite crear, modificar o eliminar participantes de la aplicación.

Atributos

- lNombres: Es la lista con los nombres de los participantes del sistema.

- lID: Es la lista con los identificadores de los participantes del sistema.

- list: Es la vista de la lista de los nombres de los participantes.

- crearParticipante: Es el botón de crear participante.

- atras: Es el botón para volver a la *activity* anterior.

- adaptador: Es el adaptador personalizado de la *ListView*.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, comprueba la acción a realizar y obtiene los participantes de la clase de dominio. Después, prepara los eventos para los botones y la lista. Al botón crearParticipante se le prepara para ir a la actividad *ParticipanteActivity* en cuanto se haga click.

- onCreateContextMenu(): Se crea el menú contextual con las opciones de editar participante y borrar participante.

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.5 ListaParticipantesReunionActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_lista_participantes_reunion*. Es la clase en la que se muestra la lista de participantes en una reunión. Podemos añadir nuevos participante o eliminarlos.

Atributos

- lp: Es la lista con los nombres de los participantes de la reunión.

- lpID: Es la lista con los identificadores de los participantes de la reunión.

- lp_seleccionados: Es la lista con los nombres de los participantes seleccionados para eliminar de la reunión.

- lp_seleccionadosID: Es la lista con los identificadores de los participantes seleccionados para eliminar de la reunión.

- guardarLPR: Es el botón de guardar lista de participantes.

- agregarLPR: Es el botón de agregar participantes.

- borrarAllLPR: Es el botón de eliminar a todos los participantes seleccionados de la reunión.

- list: Es la vista de la lista de los participantes a la reunión.

- odd: Es el orden del día de la reunión.

- orden: Es el orden máximo actual del orden del día.

- reunion: Es la reunión actual.

- inicio: indicador para saber si venimos de la pantalla principal.

- bd: Es la base de datos.

- oddBorrar: Son los puntos del orden del día que se han de eliminar. Se arrastran a esta pantalla ya que sólo se borrarán si se guardan las modificaciones de la reunión.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene los datos procedentes de anteriores actividades. Después, prepara los eventos para los botones y la lista. Al botón agregarLPR se le prepara para ir a la actividad *SeleccionarParticipantesActivity* en cuanto se haga click.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.6 ListaReunionesActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_lista_reuniones*. Es la clase en la que se muestran la lista de las reuniones en el sistema. También permite crear, modificar o eliminar reuniones de la aplicación, así como comenzar el desarrollo de ésta.

Atributos

- lReuniones: Es la lista con los nombres de las reuniones del sistema.

- lRID: Es la lista con los identificadores de las reuniones del sistema.

- list: Es la vista de la lista de los nombres de las reuniones.

- crearReunion: Es el botón de crear reunión.

- comenzarReunion: Es el botón para comenzar el desarrollo de la reunión.

- atras: Es el botón para volver a la *activity* anterior.

- adaptador: Es el adaptador personalizado de la *ListView*.

- bd: Es la base de datos.

- reunionNumber: Indica la opción seleccionada actualmente.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, comprueba la acción a realizar y obtiene las reuniones de la clase de dominio. Después, prepara los eventos para los botones y la lista. Al seleccionar un elemento de la lista se actualiza el marcador y se cambian los colores. Al botón *crearReunion* se le prepara para ir a la actividad Reunion*Activity* en cuanto se haga click y al botón *comenzarReunion para ir a ReunionPlayActivity*.

- onCreateContextMenu(): Se crea el menú contextual con las opciones de editar reunión, borrar reunión, copiar emails al portapapeles y finalizar reunión.

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.7 MainActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_main*. Como su nombre indica, es la clase principal de la aplicación y en la que se ponen a disposición del usuario todas las opciones.

Atributos

- crear: Es el botón de crear reunión.

- listaReunion: Es el botón para abrir la lista de reuniones del sistema.

- listaParticipante: Es el botón para abrir la lista de participantes del sistema.

- comenzar: Es el botón para comenzar la próxima reunión programada.

- actas: Es el botón para abrir la lista de actas.

- nombreProx: Es el *TextView* con el nombre de la próxima reunión programada.

- reunion: Es el nombre de la próxima reunión programada.

- rid: Es el identificador de la próxima reunión programada.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, comprueba si es la primera vez que se ejecuta la aplicación para crear las bases de datos. Después, obtiene la lista de reuniones del sistema, las ordena por fecha y obtiene la reunión más próxima. Luego, prepara todos los botones para sus respectivas acciones.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.8 NotasActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_notas*. Es la clase en la que se muestra el contenido de una nota de un acta. Se puede leer y modificar.

Atributos

- nota: Es el *EditText* que contiene la nota

- guardar: Es el botón para guardar el contenido de la nota.

- descartar: Es el botón para descartar el contenido o los cambios realizados en la nota.

- filename: Es el nombre del archivo de la nota.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, adapta el nombre de los

datos procedentes de la *activity* anterior. Después, carga en la nota el contenido del fichero. Prepara los botones para guardar el contenido si así lo desea el usuario o para descartar los cambios.

- writeToFile(): Es la acción que, recibiendo como parámetros el nombre del archivo y el texto, crea un archivo de texto con el contenido obtenido.

- readFromFile (): Es una función que recibe como parámetro el nombre del archivo y retorna el contenido del mismo.

- volverActividadPrincipal(): Al haberse iniciado la *activity* como actividad secundaria, al regresar a la actividad anterior se utilizará este método.

8.3.9 OrdenDelDiaActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_orden_del_dia*. Es la clase en la que se muestran la lista de puntos del orden del día. Se pueden añadir o eliminar.

Atributos

- lp: Es la lista con los nombres de los participantes de la reunión. Se conserva en esta actividad porque no se guardará hasta que el usuario decida conservar los cambios de la reunión.

- lpID: Es la lista con los identificadores de los participantes de la reunión. Como en el caso anterior, también se conserva en esta actividad.

- lista: Es la vista de la lista de los puntos del orden del día.

- guardarO: Es el botón de guardar el orden del día.

- crearPuntoO: Es el botón para crear un punto nuevo.

- nombreO: Es el *EditText* que contiene el nombre del punto nuevo.

- duracionO: Es el *EditText* que contiene la duración del punto nuevo.

- orden: Es el orden máximo del orden del día.

- odd: Es la lista con todos los puntos del orden del día.

- oddBorrar: Es una lista con los puntos a borrar. No se ejecutará la eliminación hasta que el usuario no guarde la reunión.

- context: Es el contexto de la actividad.

- reunion: Contiene la reunión actual.

- inicio: Indicador para saber si la actividad anterior fue *MainActivity*.

- adaptador: Es el adaptador personalizado de la *ListView*.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene datos de la actividad anterior. Después, prepara los eventos para los botones y la lista. También se prepara para ejecutar los cambios de posición si se efectúan borrados o añadidos.

- onCreateContextMenu(): Se crea el menú contextual con la única opción de borrar punto.

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- bajarOrden(): Desciende en una unidad el valor de *orden*, en caso de que se elimine un punto.

- maxOrden(): calcula el máximo orden del orden del día.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.10 ParticipanteActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_participante.* Es la clase en la que se introducen los datos de un nuevo participante para crearlo en el sistema.

Atributos

- nombre: Es el *EditText* que contiene el nombre del participante

- empresa: Es el *EditText* que contiene el nombre de la empresa en la cual trabaja el participante

- motivaciones: Es el *EditText* que muestra las motivaciones del participante.

- email: Es el *EditText* que contiene la dirección de correo electrónico del participante.

- telefono: Es el *EditText* que contiene el número de teléfono del participante.

- otros: Es el *EditText* que contiene otros datos de interés del participante.

- crear: Es el botón para guardar un nuevo participante

- atras: Es el botón para volver a la *activity* anterior.

- context: Es el contexto de la actividad.

- pID: contiene el identificador del participante.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, comprueba si hay que cargar datos de algún participante. Finalmente, prepara los botones para dar de alta a un nuevo participante o descartarlo.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.11 ReunionActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_reunion.* Es la clase en la que se introducen los datos de una nueva reunión para crearla en el sistema.

Atributos

- nombreR: Es el *EditText* que contiene el nombre de la reunión.

- descripcionR: Es el *EditText* que contiene la descripción de la reunión

- organizadorR: Es el *EditText* que muestra el nombre del organizador de la reunión.

- moderadorR: Es el *EditText* que contiene el nombre del moderador de la reunión.

- lugarR: Es el *EditText* que contiene el lugar en el que se desarrollará la reunión.

- diaR: Es el *EditText* que contiene la fecha para la cual está prevista la reunión.

- horaR: Es el *EditText* que contiene la hora a la que está prevista la reunión.

- crearOrdenR: Es el botón para crear una nueva orden del día.

- guardarR: Es el botón para guardar la nueva reunión.

- participantesR: Es el botón para crear una lista de asistentes a la reunión.

- atras: Es el botón para volver a la *activity* anterior.

- context: Es el contexto de la actividad.

- rID: contiene el identificador de la reunión.

- orden: Es el orden máximo del orden del día.

- odd: Es la lista del orden del día.

- lp: Es la lista que contiene el nombre de los participantes a la reunión.

- lpID: Es la lista que contiene el identificador de los participantes a la reunión.

- reunion: Es la reunión actual.

- inicio: Indicador para saber si la actividad precedente es *MainActivity*.

- oddBorrar: Puntos de la orden del día que hay que borrar si se aceptan los cambios en la reunión.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, comprueba si hay que cargar datos de alguna reunión. Después, hace las comprobaciones pertinentes y si todo está correcto comenzará la actualización de reunión, orden del día y turnos. Finalmente, prepara los botones para dar de alta a una nueva reunión o descartarla.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

- diaExiste(): Función que retorna si la fecha introducida existe.

- horaExiste():Función que retorna si la hora introducida existe.

8.3.12 ReunionPlayActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_reunion_play*. Es la clase en la que se desarrolla la reunión, controlando los tiempos y recogiendo el acta.

Atributos

- cronometro: Es la vista del cronómetro general del punto de la reunión.

- list: Es la vista de la lista que contiene los participantes de la reunión.

- nota: Es el botón de escribir notas.

- grabacion: Es el botón de realizar las grabaciones.

- transcripcion: Es el botón de realizar las transcripciones.

- iniciar: Es el botón para iniciar el cronómetro.

- siguientePunto: Es el botón para pasar al siguiente punto de la reunión.

- atras: Es el botón para volver a la *activity* anterior.

- play: Es la instancia de la clase de dominio *ReunionPlay*.

- punto: Es el *TextView* que indica el nombre del punto actual.

- tiempo: Es el *TextView* que indica el tiempo estimado para el punto actual.

- crono: Es el cronómetro del participante seleccionado.

- cronoTotal: Es el cronómetro general del punto de la reunión.

- on: Es el booleano que indica si el cronómetro está activo.

- participante: Indica al participante de la lista seleccionado.

- grabar: Indica el número de archivos grabados.

- transcribir: Indica el número de archivos transcritos.

- grabando: Booleano que indica si la aplicación está grabando audio.

- mr: instancia de la clase necesaria para realizar las grabaciones de audio.

- bd: Es la base de datos.

- reunion: Instancia de la clase Reunion con los datos de la reunión actual.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, prepara a la aplicación para las tres acciones para guardar el acta. También, prepara los datos que irán en la lista. Después, prepara los eventos para los botones y la lista. Al seleccionar una opción en la lista se modifica el marcador, se cambia el color se la opción seleccionada y se seleccionado el cronómetro del participante activo. A cada botón se le asigna la operación adecuada que describiremos a continuación.

- notas(): Acción que prepara los archivos y los envía a la actividad secundaria que lanza (*ActaActivity*).

- grabaciones(): Esta acción llama a *iniciarGrabacion()* si no está grabando y a *finalizarGrabacion()* si sí lo está.

- transcripciones(): La acción prepara los archivos, buscándolos aplicando expresiones regulares, e inicia la aplicación de Google de transcripción como actividad secundaria.

- iniciarGrabacion(): Esta operación prepara al archivo para grabar audio, configurando el nombre, el formato de audio y el codificador, y comienza la grabación.

- finalizarGrabacion(): Finaliza una grabación de audio.

- getTranscripcionNumber(): obtiene el último número de los archivos de transcripción.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

- onActivityResult(): Aquí recibe los resultados de la transcripción de Google, que la aplicación manda a la actividad de *NotasActivity*.

8.3.13 SeleccionarParticipantesActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_seleccionar_participantes*. Es la clase en la que se muestra la lista de posibles participantes en una reunión que aún no han sido elegidos.

Atributos

- lp: Es la lista con los nombres de los participantes que aún no están en la reunión.

- lpID: Es la lista con los identificadores de los participantes que aún no están en la reunión.

- agregarSP: Es el botón para confirmar a los participante agregados

- listSP: Es la vista de la lista de los participantes que aún no están en la reunión.

- odd: Es el orden del día de la reunión.

- orden: Es el orden máximo actual del orden del día.

- reunion: Es la reunión actual.

- inicio: indicador para saber si venimos de la pantalla principal.

- bd: Es la base de datos.

- oddBorrar: Son los puntos del orden del día que se han de eliminar. Se arrastran a esta pantalla ya que sólo se borrarán si se guardan las modificaciones de la reunión.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene los datos procedentes de anteriores actividades. Después, prepara los eventos para los botones y la lista. Al botón agregarSP se le prepara para ir a la actividad *ListaParticipantesReunionActivity* en cuanto se haga click, agregando los participantes que tengan su *check* marcado.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

8.3.14 TranscripcionesActivity

Explicación general

Esta clase contiene la vista del archivo *XML activity_transcripciones*. Es la clase en la que se muestran las listas de los archivos de transcripciones de una reunión y un punto concreto del orden del día.

Atributos

- files: Es una lista de nombres de los archivos de transcripción de audio pertenecientes al punto seleccionado del orden del día.

- users: Es la lista con los nombres de los participantes de la reunión.

- lista: Es la vista de la lista de los archivos de transcripción de audio.

- leer: Es el botón de leer transcripción.

- atras: Es el botón para volver a la *activity* anterior.

- key: Es la opción actual seleccionada de la lista.

- context: Es el contexto de la actividad.

- adaptador: Es el adaptador personalizado de la *ListView*.

- bd: Es la base de datos.

Métodos:

- onCreate (): Es la función principal de la clase. Primeramente, inicializa todos los atributos de la clase. Posteriormente, obtiene todos los nombres de los ficheros de la *activity* anterior. Después, prepara los eventos para los botones y la lista. Al seleccionar una opción en la lista se modifica el marcador y se cambia el color se la opción seleccionada. Al botón *leer* se le prepara para ir a la actividad *NotasActivity* con el archivo seleccionado al hacer click.

- onCreateContextMenu(): Se crea el menú contextual con la opción de borrar archivo.

- onContextItemSelected(): Se prepara la acción a seguir al seleccionar una opción del menú contextual.

- onStop (): Al salir de la actividad, cierra el canal de la base de datos.

- onRestart(): Al volver a la actividad, se vuelve a abrir el canal de la base de datos.

- volverActividadPrincipal(): Al haberse iniciado la *activity* como actividad secundaria, al regresar a la actividad anterior se utilizará este método.

8.4 Funciones detalladas

En este capítulo, se explicarán con detenimiento tres funciones: grabación, transcripción y reproducción. Se comentarán detalladamente debido a que son las funciones con una lógica más compleja.

8.4.1 Grabación

Empezaremos comentando la función de grabación.

```
100         mr = new MediaRecorder();
```

Primero, si inicia la instancia de la clase encargada de la grabación de audio, *MediaRecorder*.

```
280     public void iniciarGrabacion(){
281         Participante p = new Participante(bd, play.getParticipantesID().get(participante));
282         String nameFile = play.getReunion().get_rID()+"_"+p.get_pID()+"_"+play.getTemaActual().get_oOrden();
283         mr.setAudioSource(MediaRecorder.AudioSource.MIC);
284         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.GINGERBREAD_MR1)
285         mr.setOutputFormat(MediaRecorder.OutputFormat.AMR_NB);
286         else mr.setOutputFormat(MediaRecorder.OutputFormat.RAW_AMR);
287         String path = this.getFilesDir()+ "//Graba_"+nameFile+"_"+Integer.toString(grabar)+".amr";
288         mr.setOutputFile(path);
289         mr.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
290         try {
291             mr.prepare();
292             mr.start();
293             grabando=true;
294             Drawable draw = getResources().getDrawable(R.drawable.ic_action_stop);
295             grabacion.setCompoundDrawablesWithIntrinsicBounds(draw, null, null, null);
296         } catch (IllegalStateException e) {
297             // TODO Auto-generated catch block
298             e.printStackTrace();
299         } catch (IOException e) {
300             // TODO Auto-generated catch block
301             e.printStackTrace();
302         }
303     }
```

Al iniciar la función de grabación, lo primero es obtener el nombre del archivo. Después, configuramos como dispositivo de entrada el micrófono del teléfono. Posteriormente, se configura el formato de audio. Se decidió el formato *AMR NB* porque es el que mejor captura las voces. Hacemos la comparación de versión porque a partir de la versión *Gingerbread MR1*, se sustituyeron los formatos. Esto es necesario para hacer la aplicación compatible con todas las versiones. Después, se actualiza el nombre del archivo según el número de grabaciones, se configura el archivo de salida y el codificador. Finalmente, se prepara para empezar la grabación y la comenzamos. Actualizamos booleano y sustituimos la imagen de grabar del botón por la de detener grabación.

Este es el código para detener la grabación:

```
305     public void finalizarGrabacion(){
306         grabar++;
307         mr.stop();
308         mr.release();
309         Drawable draw = getResources().getDrawable(R.drawable.grabar);
310         grabacion.setCompoundDrawablesWithIntrinsicBounds(draw, null, null, null);
311         mr = new MediaRecorder();
312 
313     }
```

Primero marcamos que hay un nuevo archivo grabado. Paramos la grabación y liberamos el canal. Finalmente, volvemos a poner la imagen de grabar en el botón y se inicia una nueva instancia de *MediaRecorder*.

8.4.2 Transcripción

A continuación, comentaremos la función de transcripción.

```
256    public void transcribir(){
257          Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
258          intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
259          RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
260          intent.putExtra(RecognizerIntent.EXTRA_PROMPT,"Comenzando Transcripción");
261          startActivityForResult(intent, 1);
262      }
```

Primero, indicamos que la próxima actividad será la transcripción de *Google*. Acto seguido, indicamos que el idioma de la transcripción será el estándar, el configurado en nuestro dispositivo. Indicamos al usuario que comienza la transcripción. Finalmente, iniciamos la transcripción como actividad secundaria.

```
316    protected void onActivityResult(int requestCode,int resultCode, Intent pData)
317    {
318
319        if(requestCode == 1 && resultCode == RESULT_OK){
320            ArrayList<String> matches = pData.getStringArrayListExtra
321              (RecognizerIntent.EXTRA_RESULTS);
322
323            Intent k = new Intent(ReunionPlayActivity.this, NotasActivity.class);
324            Participante p = new Participante(bd, play.getParticipantesID().get(participante));
325            String nameFile = play.getReunion().get_rID()+"_"+p.get_pID()+"_"+play.getTemaActual().get_oOrden();
326            k.putExtra("playNameFile", nameFile);
327            k.putExtra("playOption", "Transcripcion");
328            k.putExtra("playTexto", matches.get(0));
329            k.putExtra("playVez", transcribir);
330            transcribir++;
331            startActivityForResult(k, 0);
332
333        }
334    }
```

Cuando *Google* acaba su servicio, esta función comprueba si el resultado ha sido correcto. Después, le indicamos que la próxima actividad será la actividad de notas. Configuramos el nombre del archivo. Preparamos los datos que se mandarán a la próxima actividad. Indicamos que hay un

nuevo archivo creado de transcripción. Finalmente, iniciamos la actividad de notas como secundaria.

8.4.3 Reproducción

A continuación, se detallará la función de reproducción de audio.

```
80              if (key >= 0){
81
82                  if (reproduciendo == false){
83
84                      mediaPlayer = new MediaPlayer();
85                      mediaPlayer.setOnCompletionListener(new OnCompletionListener() {
86
87                          @Override
88                          public void onCompletion(MediaPlayer mp) {
89                              // TODO Auto-generated method stub
90                              if (reproduciendo == true){
91                                  reproduciendo = false;
92                                  reproducir.setText("Reproducir");
93                              }
94                          }
95                      });
96
97                      File file = new File(context.getFilesDir(),files.get(key));
98                      FileInputStream inputStream;
99                      try {
100                         inputStream = new FileInputStream(file);
101                         mediaPlayer.setDataSource(inputStream.getFD());
102                         inputStream.close();
103                     } catch (FileNotFoundException e) {
104                         // TODO Auto-generated catch block
105                         e.printStackTrace();
106                     } catch (IllegalArgumentException e) {
107                         // TODO Auto-generated catch block
108                         e.printStackTrace();
109                     } catch (IllegalStateException e) {
110                         // TODO Auto-generated catch block
111                         e.printStackTrace();
112                     } catch (IOException e) {
113                         // TODO Auto-generated catch block
114                         e.printStackTrace();
```

Primero, se inicializa la instancia de la clase encargada de reproducir archivos: *MediaPlayer*.

Al iniciar la función de reproducción, primero comprobamos que no haya archivo reproduciendo. Además, creamos un evento que indica cuando un archivo ha terminado su reproducción. Posteriormente, obtenemos el fichero que queremos reproducir. Y capturamos excepciones si éstas ocurren.

```
116                     try {
117                         mediaPlayer.prepare();
118                     } catch (IllegalStateException e) {
119                         // TODO Auto-generated catch block
120                         e.printStackTrace();
121                     } catch (IOException e) {
122                         // TODO Auto-generated catch block
123                         e.printStackTrace();
124                     }
125 
126                     reproduciendo = true;
127                     mediaPlayer.start();
128                     reproducir.setText("Detener");
```

Después, preparamos al archivo para grabar. Si no hay excepciones, reproducimos el archivo, actualizamos booleano y cambiamos el texto del botón de reproducción por *Detener*.

Este es el código para detener la reproducción:

```
131                 else{
132                     reproduciendo = false;
133                     mediaPlayer.stop();
134                     reproducir.setText("Reproducir");
135                 }
```

Sólo hay que detener la reproducción, actualizar el booleano y restablecer el texto del botón de reproducción.

9. Conclusiones

Este proyecto tenía una serie de objetivos y, en su mayoría, se han cumplido.

El proyecto representa el último paso para obtener la licenciatura en Ingeniería Superior en Informática. Tras un duro y largo recorrido en la facultad, el PFC es el final de esta carrera.

Este proyecto trabaja casi todos los aprendizajes adquiridos durante la carrera: programación orientada a objetos, arquitectura de tres capas, sistemas operativos, lenguajes de marcas, inteligencia artificial, estructuras de datos, punteros, bases de datos, gestión de ficheros a nivel de sistema operativo, planificación y gestión de proyectos, ingeniería de software y visualización gráfica.

La aplicación consigue los objetivos marcados, se ha creado una APP rápida, intuitiva y completa, que permita al usuario utilizarla como herramienta apoyo en sus reuniones.

Se ha podido crear una herramienta que almacena los datos necesarios para realizar reuniones, permite fácilmente cambiar entre participantes, guardar actas en forma de notas, transcripciones y grabaciones de audio y consultar los datos y las actas de una forma eficaz.

Haciendo una valoración del trabajo realizado, sorprende que la parte de implementación de funcionalidades (grabación, transcripción o desarrollo de la reunión) no haya sido la parte más complicada. Trabajar con Android y todas sus particularidades, como el concepto de ciclo de vida de las actividades, el envío de datos entre ellas, las limitaciones propias de un sistema liviano, el tratamiento de las listas y los problemas de compatibilidad de las diferentes versiones del sistema operativo han sido las tareas más complicadas y que más quebraderos de cabeza me ha dado. Realmente, trabajar con Java es un placer, sin embargo, con las particularidades de Android se complica la tarea de programar.

Por otro lado, el programa es muy extenso. Un total de 30 clases y una gran cantidad de operaciones y atributos han hecho que el tiempo de programación del proyecto se haya incrementado bastante más del previsto en un principio.

A pesar de todo esto, he preferido mantener el alcance planeado en un principio aumentando el tiempo de desarrollo del proyecto.

Finalmente, al comprobar que la aplicación cumple con todo lo planeado y ver el resultado final, me siento muy satisfecho del trabajo realizado. Pues, al final, todo los conocimientos adquiridos durante la carrera se han materializado en esta aplicación, LeAndroid Meeting.

9.1 Posibles mejoras futuras

Con más tiempo de desarrollo, se podrían realizar mejoras en algunos apartados.

Se podría investigar sobre el reconocimiento del hablante, aunque esto llevaría bastante tiempo y necesitaría una aplicación y un servidor aparte que genere y entrene los modelos para compararlos posteriormente.

También podría ser interesante implementar un servicio de envíos de correos electrónicos para informar a los asistentes. Actualmente, hay una

funcionalidad para copiar al portapapeles las direcciones de correos electrónicos de todos los participantes de una reunión, de tal forma que se puedan pegar en nuestro webmail favorito.

Finalmente, se podría implementar un sistema para importar contactos de nuestro móvil aunque exigiría un mapeo de campos.

Todos estos cambios mejorarían la aplicación, pero necesitan un tiempo de desarrollo y unos medios tecnológicos que no los hacían viables para este proyecto.

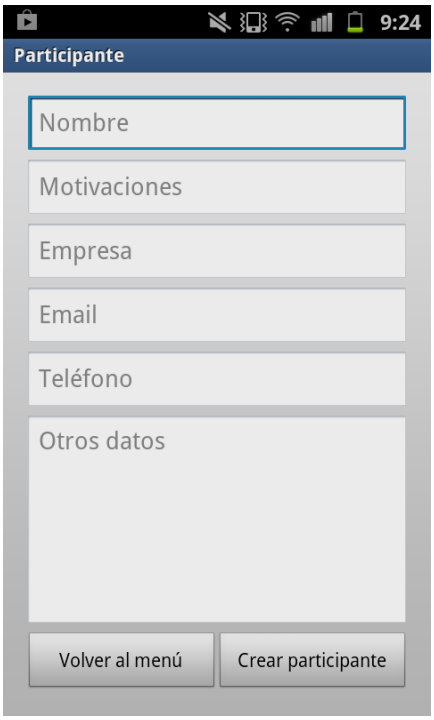
10. Manual de instrucciones

Este capítulo muestra un manual para el funcionamiento de la aplicación. Esta es la pantalla principal del programa y de la cual partimos:

10.1 Añadir participante

Vamos a Lista de Participante.

Y seleccionamos Crear Participante.

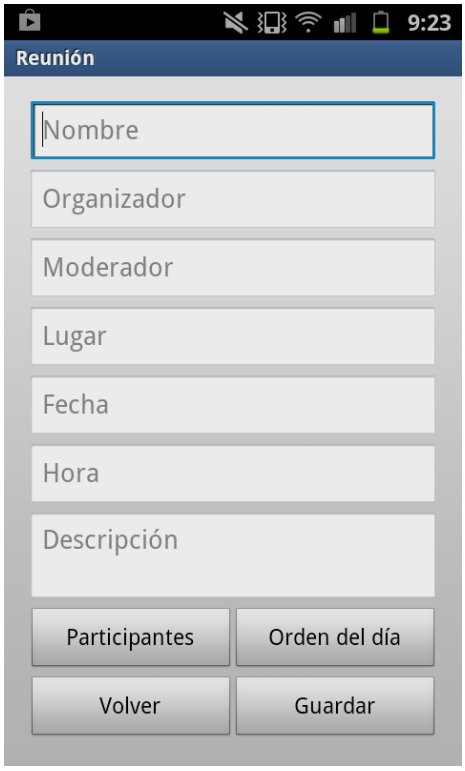


Rellenamos los datos y creamos al participante.

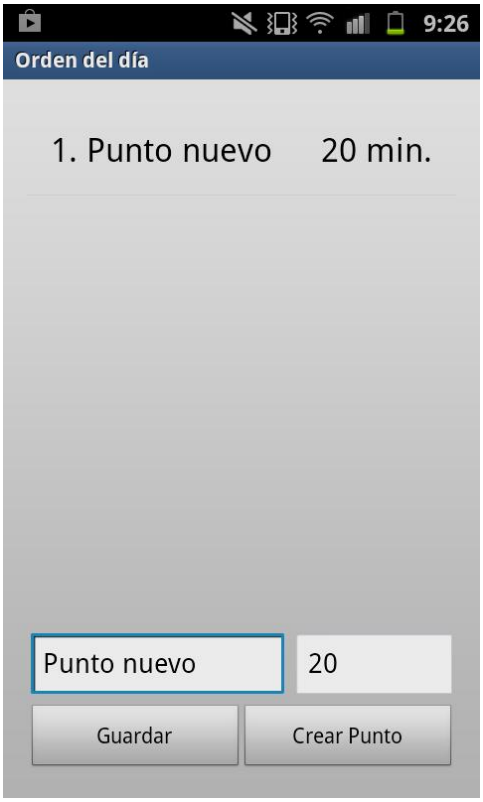
Y ya tenemos a nuestro participante creado.

10.2 Crear reunión

Vamos a crear reunión.

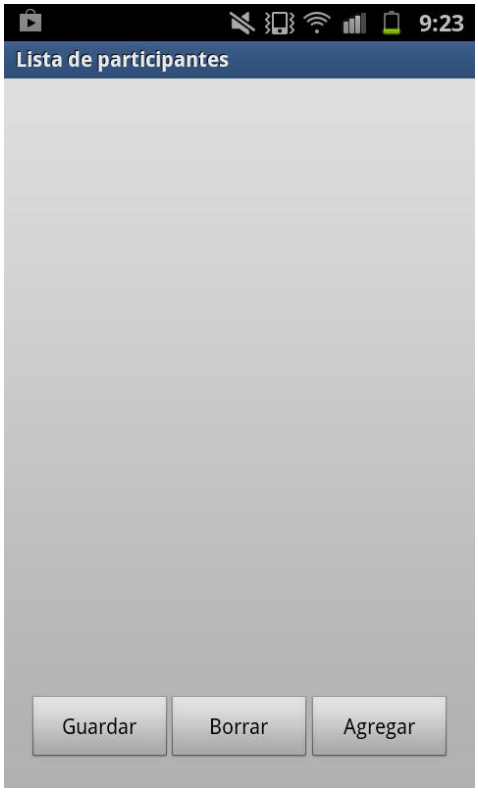


Rellenamos los datos y creamos el Orden del día.



Creamos los puntos y guardamos.

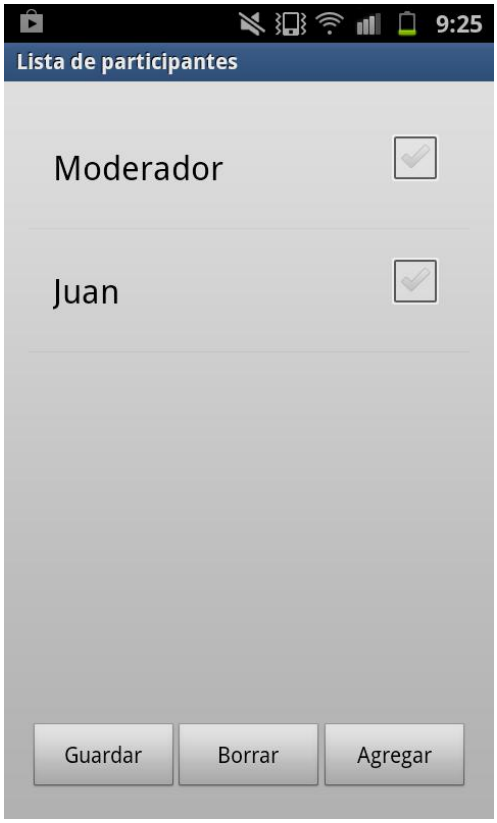
Creamos la lista de participantes a la reunión.



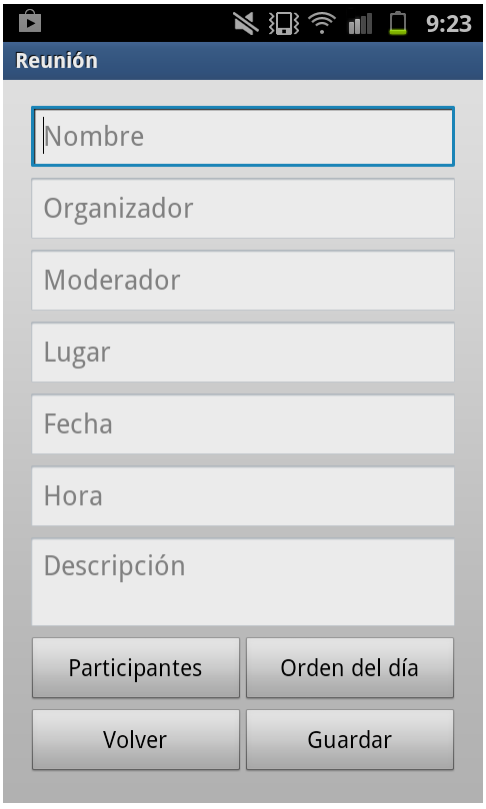
En un principio, estará vacía. Le damos a Agregar.

Seleccionamos participantes y les damos a agregar.

Volvemos a la pantalla de selección de participantes.



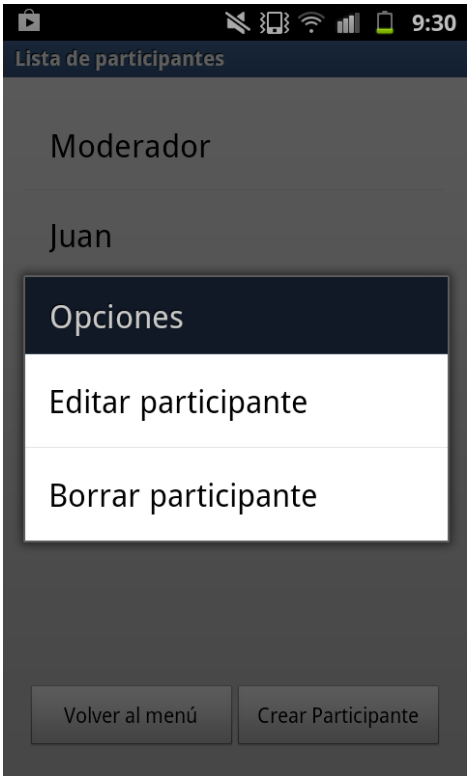
Si estamos de acuerdo, guardamos.



Cuando estén los datos necesarios, guardamos la reunión.

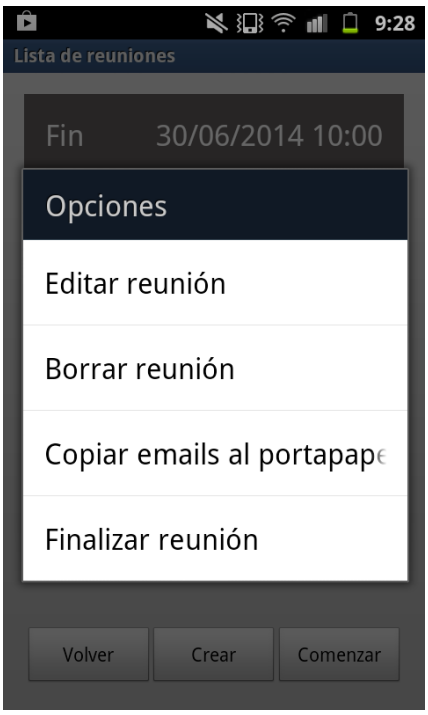
10.3 Opciones participante

En la lista de participantes, mantenemos pulsado una opción.



10.4 Opciones reunión

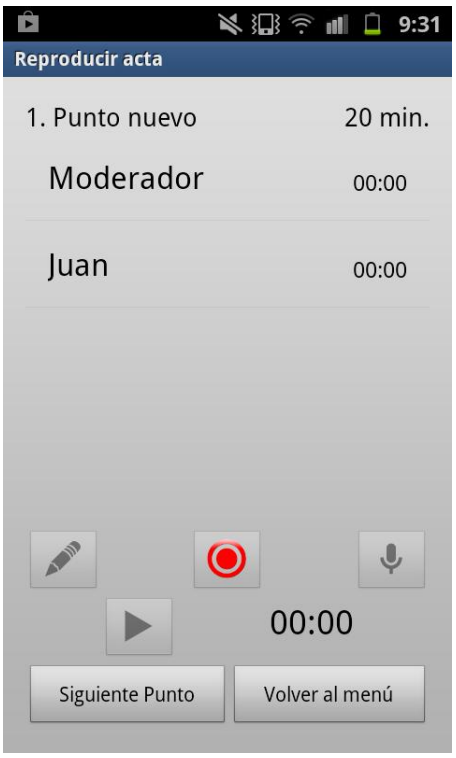
En la lista de reuniones, mantenemos pulsado una opción.



10.5 Comenzar reunión

Vamos a la lista de reuniones.

Seleccionamos la reunión y pulsamos el botón de comenzar.



10.6 Selección de participante

En la pantalla de desarrollo de reunión, seleccionamos a un participante.

Al dar al botón play empiezan los cronómetros a correr.

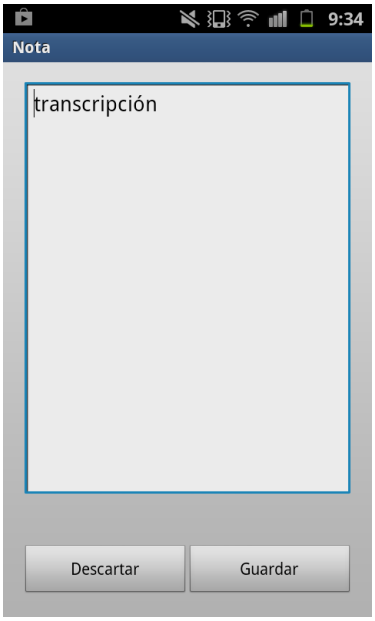
10.7 Grabar audio

Al pulsar el botón de grabar, comenzará la grabación de audio.

El botón cambia a detener grabación. Si se pulsa, se detiene la grabación y se guarda el archivo.

10.8 Transcribir

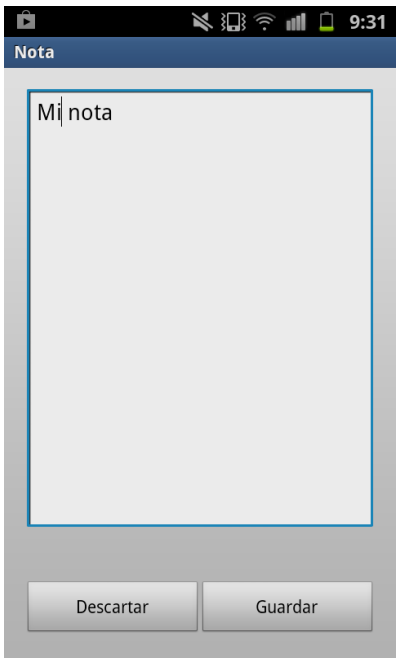
Al pulsar el botón de transcripción, comenzará la transcripción de audio



Devuelve el resultado de la transcripción.

10.9 Nota

Al pulsar el botón de nota, se abrirá la actividad de nota.



Podemos guardar la nota o descartarla.

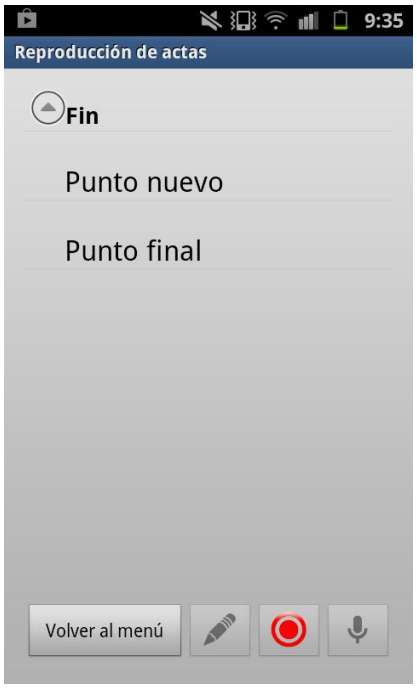
10.10 Finalizar reunión

En la pantalla de desarrollo de reunión

si es el último punto aparecerá la opción de fin de reunión.

10.11 Consultar nota

Vamos a la sección de actas



Donde seleccionamos una reunión y un punto y le damos al botón de notas.

Y aparecerán las notas de cada uno de los participantes.

Si seleccionamos una podremos leerla con el botón leer nota.

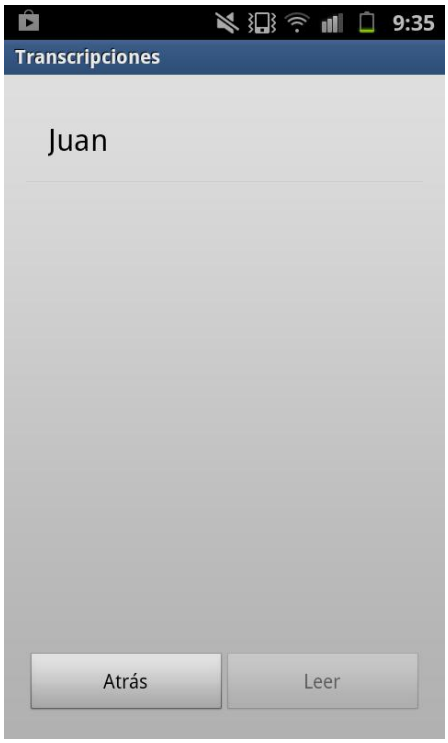
10.12 Reproducir grabación de audio

En la sección de actas seleccionamos el botón de grabaciones.

Nos aparecen los archivos guardados que podremos reproducir dándole al botón.

10.13 Consultar transcripciones

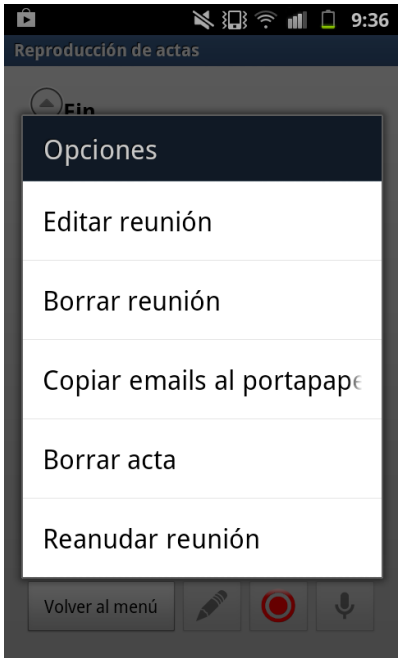
En la sección de actas seleccionamos el botón de transcripciones.



Nos aparecen los archivos guardados que podremos leer dándole al botón.

10.14 Opciones de reunión finalizada

En la sección de actas mantenemos pulsado sobre una reunión.



Y nos muestra el menú contextual.